

---

# **Annize**

***Release 6.0.6471***

**Author name not set**

**Jul 26, 2025**



# CONTENTS

<b>1</b>	<b>License</b>	<b>3</b>
<b>2</b>	<b>Up-To-Date?</b>	<b>5</b>
<b>3</b>	<b>Dependencies</b>	<b>7</b>
<b>4</b>	<b>User Manual</b>	<b>9</b>
4.1	Installation . . . . .	9
4.2	First Steps . . . . .	9
4.3	Project Configuration . . . . .	9
4.4	Graphical User Interface . . . . .	9
4.5	The Annize Programming Interface . . . . .	9
<b>5</b>	<b>Command Line Interface Reference</b>	<b>11</b>
5.1	Positional Arguments . . . . .	11
5.2	Named Arguments . . . . .	11
5.3	Sub-commands . . . . .	11
<b>6</b>	<b>API Reference</b>	<b>13</b>
6.1	annize namespace . . . . .	13
	<b>Python Module Index</b>	<b>127</b>
	<b>Index</b>	<b>129</b>



TODO Anise is a Python-based execution engine for automation tasks. Automation tasks exist in software development and probably all kinds of other sectors. They typically require the execution of different smaller and larger tools. Complex tasks often need a sequence of many steps to execute, with some steps having dependencies to each other. Manually triggering all these steps in the graphical interfaces of all the involved tools is possible in theory, but will generate errors and frustration after some cycles. The automation interfaces of those tools are sometimes easier, but sometimes they are error-prone. Some tasks may also need to ask the user for some information in an interactive way. Some smaller parts might also be machine-specific (e.g. filesystem paths or the code how to access a password vault), while the entire task must be runnable on some different machines. In some situations, this can lead to a rather intransparent forest of different tools, with unique oddnesses and special conventions. As the number of different project increases, you will see more and more different tools, often doing a similar job, but for different platforms or frameworks and, of course, with different usage conventions. Spontaneously written glue scripts help in the beginning, but will explode as the complexity exceeds some threshold. Typical tasks in software development could be:

- Generating documentation- Testing- Automatic code generation- Creating packages- Creating a homepage, automatically built from the available version information, the packages, the documentation and so on- Deploying this homepage to a web server- Handling version information - e.g. print it in the manual- and many moreThe Anise framework allows you to implement all those tasks in a structured but generic way in a combination of XML and Python code. Once you have created this stuff at a defined place in your project, Anise lets you easily execute your tasks from command line (or from any editor if you embed it somehow). This gives you a common and easy interface to all your 'tool glue' code.The Anise engine executes arbitrary Python sourcecode and provides some additional services like logging, parameter passing from command line, basic graphical userinterface support, a plugin interface, a flexible event system, injecting code and data from other place,dependencies between code fragments, and more.On top of this engine, Anise comes with a bunch of implementations that fulfill tasks (or parts of them) of software development. There is a testing module, a documentation- and homepage-generator, some package building methods and a lot more. The implementations use the event system in many places in order to allow customization in a somewhat technical but very flexible way. Even so, those implementations are rather specific and it depends on the particular case, if, and how many of those implementations are useful.



## **LICENSE**

Annize is distributed under the terms of the AGPL 3 license. This also affects all included files without a license header (non-source files like images), unless they are explicitly mentioned as third-party content. Read the ‘Dependencies’ section for included third-party stuff.





## **UP-TO-DATE?**

Are you currently reading from another source than the homepage? If you are in doubt whether your package is up-to-date, you should visit the project homepage and check that. You are currently reading the documentation for version 6.0.6471.



## DEPENDENCIES

Annize makes use of some third-party parts.



Required: **Python 3.13**



Required: **Python package PyGObject** `~= 3.52`



Required: **Python package hallyd** `~= 0.90`



Required: **Python package klovve** `~= 1.6`



Required: **Python package lxml** `~= 6.0`



Required: **Python package pycountry** `~= 24.6`

Required: **GTK 4**



Recommended: **GNU/Linux**



Included: **background image** (License: <http://creativecommons.org/licenses/by-sa/3.0>; from [here](#))



Included: **font ‘Inconsolata’** (for websites; by Raph Levien; License: OFL; [from here](#))



Included: **font ‘Khula’** (for websites; by Erin McLaughlin; License: OFL; [from here](#))



Included: **font ‘Symbola’** (for logo symbol; License: free for use; [from here](#))



Included: **icon set ‘Oxygen’** (some icons; [from here](#))



Included: **third-party project logos** ([from here](#))

## 4.1 Installation

TODO zz

## 4.2 First Steps

TODO zz TODO zz cli TODO zz file format TODO zz gui

## 4.3 Project Configuration

TODO zz TODO zz classes TODO zz features/xml namespaces TODO zz names, references and append\_to

### 4.3.1 Model

TODO zz

### 4.3.2 File Format

TODO zz

## 4.4 Graphical User Interface

TODO zz

## 4.5 The Annize Programming Interface

TODO zz TODO zz i18n TODO zz fs

### 4.5.1 Features

TODO zz



## COMMAND LINE INTERFACE REFERENCE

```
usage: annize [-h] [--project PROJECT]
              [--with-answers-from-json-file WITH_ANSWERS_FROM_JSON_FILE]
              [--with-answers-from-json-string WITH_ANSWERS_FROM_JSON_STRING]
              [--with-answer WITH_ANSWER WITH_ANSWER]
              [command] ...
```

### 5.1 Positional Arguments

**[command]**            Possible choices: do  
                      The command to execute.

### 5.2 Named Arguments

**--project**            Project directory or Annize configuration file (otherwise: the current working directory). Annize will automatically try to find it in parent directories as well.

**--with-answers-from-json-file** TODO  
                      Default: []

**--with-answers-from-json-string** TODO  
                      Default: []

**--with-answer**        TODO  
                      Default: []

### 5.3 Sub-commands

#### 5.3.1 do

Execute a task.

```
annize do [-h] [task_name]
```

### Positional Arguments

<b>task_name</b>	Task name.
	Default: ''



## API REFERENCE

## 6.1 annize namespace

### 6.1.1 Subpackages

**annize.asset** package

**Submodules**

**annize.asset.data** module

`annize.asset.data.readme_pdf(culture)`

**Parameters**

**culture** (*str*)

**Return type**

*Path*

**annize.asset.project\_info** module

**annize.data** package

**Submodules**

**annize.data.color** module

**class** `annize.data.color.Color`(\* (*Keyword-only parameters separator (PEP 3102)*), *red*, *green*, *blue*)

Bases: `object`

**Parameters**

- **red** (*float*)
- **green** (*float*)
- **blue** (*float*)

**property** `red`: `float`

**property** `green`: `float`

**property** `blue`: `float`

**property** `hue`: `float`

**property lightness:** float

**property saturation:** float

**with\_modified**(\*, *red=None, green=None, blue=None, hue=None, lightness=None, saturation=None*)

**Parameters**

- **red** (*float | None*)
- **green** (*float | None*)
- **blue** (*float | None*)
- **hue** (*float | None*)
- **lightness** (*float | None*)
- **saturation** (*float | None*)

**Return type**

[Color](#)

**property html\_color\_spec:** str

**scalehue**(\*, *brightness, saturation=None*)

**Parameters**

- **brightness** (*float*)
- **saturation** (*float | None*)

**Return type**

[Color](#)

**transformed**(\*, *brightness=None, saturation=None*)

**Parameters**

- **brightness** (*float | None*)
- **saturation** (*float | None*)

**Return type**

[Color](#)

## **annize.data.container module**

**class** annize.data.container.**Basket**(*\*args, \*\*kwargs*)

Bases: list

## **annize.data.unique module**

**class** annize.data.unique.**UniqueId**(*localid=None*)

Bases: object

**Parameters**

**localid** (*str | None*)

**\_\_uniqueid\_counter** = 0

**\_\_uniqueid\_lock** = <unlocked \_thread.lock object>

```

property long_str: str
property short_str: str
property processonly_long_str: str
property processonly_short_str: str
__long_str(txt)

```

**Parameters**  
*txt* (*str*)

**Return type**  
*str*

### annize.data.version module

```
class annize.data.version.AbstractVersionPatternSegment
```

Bases: ABC

```
property segment_names: list[str]
```

```
abstractmethod regexp_string()
```

**Return type**  
*str*

```
abstractmethod segments_tuples_to_text(segments_tuples)
```

**Parameters**  
*segments\_tuples* (*list*[*Tuple*[*str*, *str*]])

**Return type**  
*str*

```
_name_anon(namep)
```

**Parameters**  
*namep* (*str*)

**Return type**  
*None*

```
_str_to_val(txt)
```

**Parameters**  
*txt* (*str*)

**Return type**  
*object*

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.data.version.NumericVersionPatternSegment(*, partname=None)
```

Bases: [AbstractVersionPatternSegment](#)

**Parameters**  
*partname* (*str* | *None*)

```
property partname: str | None
```

```
property segment_names: list[str]
```

```
regexp_string()
```

Return type

str

```
segments_tuples_to_text(segments_tuples)
```

Parameters

`segments_tuples` (*list*[*Tuple*[*str*, *str*]])

Return type

str

```
_name_anon(namep)
```

```
_str_to_val(txt)
```

Parameters

`txt` (*str*)

Return type

object

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.data.version.SeparatorVersionPatternSegment(*, text)
```

Bases: *AbstractVersionPatternSegment*

Parameters

`text` (*str*)

```
regexp_string()
```

Return type

str

```
segments_tuples_to_text(segments_tuples)
```

Parameters

`segments_tuples` (*list*[*Tuple*[*str*, *str*]])

Return type

str

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.data.version.OptionalVersionPatternSegment(*, segments)
```

Bases: *AbstractVersionPatternSegment*

Parameters

`segments` (*Iterable*[*AbstractVersionPatternSegment*])

```
property segment_names
```

```
regexp_string()
```

Return type

str

```

segments_tuples_to_text(segments_tuples)

    Parameters
        segments_tuples (list[Tuple[str, str]])

    Return type
        str

_name_anon(namep)

_abc_impl = <_abc._abc_data object>

class annize.data.version.ConcatenatedVersionPatternSegment(*, segments)
    Bases: AbstractVersionPatternSegment

    Parameters
        segments (Iterable[AbstractVersionPatternSegment])

    property segment_names

    regexp_string()

    Return type
        str

    segments_tuples_to_text(segments_tuples)

        Parameters
            segments_tuples (list[Tuple[str, str]])

        Return type
            str

        _name_anon(namep)

        _abc_impl = <_abc._abc_data object>

class annize.data.version.VersionPattern(*, segments)
    Bases: object

    Parameters
        segments (Iterable[AbstractVersionPatternSegment])

    property segments: list[AbstractVersionPatternSegment]

    property segment_names

    text_to_segments_tuples(text)

        Parameters
            text (str)

        Return type
            list[Tuple[str, str]]

    segments_tuples_to_text(segments_tuples)

        Parameters
            segments_tuples (list[Tuple[str, str]])

        Return type
            str

```

```
class annize.data.version.Version(*, text=None, pattern=None, **segment_values)
```

Bases: object

**Parameters**

- **text** (*str*)
- **pattern** (*VersionPattern*)

property **segments\_tuples**

property **segments\_values**: *dict[str, Any]*

property **text**: *str*

property **pattern**: *VersionPattern*

**annize.features namespace**

**Subpackages**

**annize.features.changelog namespace**

**Submodules**

**annize.features.changelog.common module**

Changelogs.

```
class annize.features.changelog.common.Item(*, text)
```

Bases: object

**Parameters**

**text** (*TrStr*)

property **text**: *TrStr*

```
class annize.features.changelog.common.Entry(*, version, time, items)
```

Bases: object

**Parameters**

- **version** (*Version | None*)
- **time** (*datetime | None*)
- **items** (*Iterable[TrStr | Item]*)

property **items**: *list[Item]*

property **time**: *datetime | None*

property **version**: *Version | None*

```
class annize.features.changelog.common.Changelog(*, entries)
```

Bases: object

**Parameters**

**entries** (*Iterable[Entry]*)

property **entries**: *list[Entry]*

---

```
class annize.features.changelog.common.ByVersionControlSystemCommitMessagesChangelog(*, en-
                                                                    tries)
```

Bases: *Changelog*

Parameters

**entries** (*Iterable*[*Entry*])

**\_S\_CHANGE** = '##CHANGE:'

**\_S\_LABEL** = '##LABEL:'

property **entries**

`annize.features.changelog.common.default_changelog()`

Return type

*Changelog*

**annize.features.dependencies namespace**

**Submodules**

**annize.features.dependencies.common module**

Project dependency handling.

```
class annize.features.dependencies.common.Kind(*, label, importance)
```

Bases: object

Parameters

- **label** (*TrStr*)

- **importance** (*int*)

property **label**: *TrStr*

property **importance**: *int*

```
class annize.features.dependencies.common.Dependency(*, kind, label, comment, icon, importance=0)
```

Bases: object

Parameters

- **kind** (*Kind* / *None*)

- **label** (*TrStr* / *None*)

- **comment** (*TrStr* / *None*)

- **icon** (*str* / *None*)

- **importance** (*int*)

property **kind**: *Kind*

property **label**: *TrStr*

property **comment**: *TrStr*

property **icon**: *str*

**property importance:** int

**class** annize.features.dependencies.common.**Required**(\*\*b)

Bases: [Kind](#)

**class** annize.features.dependencies.common.**Recommended**(\*\*b)

Bases: [Kind](#)

**class** annize.features.dependencies.common.**Included**(\*\*b)

Bases: [Kind](#)

**class** annize.features.dependencies.common.**GnuLinux**(\*, kind=None, comment)

Bases: [Dependency](#)

**class** annize.features.dependencies.common.**Artwork**(\*, kind=None, label, origin, comment, license)

Bases: [Dependency](#)

**Parameters**

- **origin** (str)
- **license** (str | None)

annize.features.dependencies.common.**dependencies\_to\_rst\_text**(dependencies)

**Parameters**

**dependencies** (list[[Dependency](#)])

**Return type**

str

## annize.features.dependencies.python module

Python project dependency handling.

**class** annize.features.dependencies.python.**Python**(\*, version, kind, comment)

Bases: [Dependency](#)

**Parameters**

**version** (str)

**class** annize.features.dependencies.python.**PythonPackage**(\*, name, version, kind, comment)

Bases: [Dependency](#)

**Parameters**

- **name** (str)
- **version** (str)
- **kind** ([Kind](#) | None)
- **comment** (TrStr | None)

**property name:** str

**property version:** str

**class** annize.features.dependencies.python.**FromRequirementsFile**(\*, requirements\_file, kind=None, comment=None)

Bases: [Basket](#)



**Parameters****requirements\_file** (*str* | *Path*)**annize.features.distributables** namespace**Subpackages****annize.features.distributables.store** namespace**Submodules****annize.features.distributables.store.pypi** module

PyPI store for Python packages.

**class** annize.features.distributables.store.pypi.**Connection**(\*, *token*)

Bases: object

**Parameters****token** (*str*)**property token:** *str***class** annize.features.distributables.store.pypi.**Upload**(\*, *source*, *connection*)

Bases: object

**Parameters**

- **source** (*FilesystemContent*)
- **connection** (*Connection*)

**Submodules****annize.features.distributables.common** module

Distributables.

This defines *Group* of packages. Package groups can be provided for download on the project homepage or similar.There is also *PackageStore* for keeping a version history of packages (e.g. somewhere on disk).**class** annize.features.distributables.common.**PackageStore**

Bases: ABC

Base class for a package store.

**abstractmethod** **store\_package**(*items*, \*, *name*)

Store package items.

**Parameters**

- **items** (*Sequence*[*FilesystemContent*]) – The items to store.
- **name** (*str*) – The item name.

**Return type**

None

**abstractmethod** **package**(\*, *name*)

Return package items.

**Parameters**

**name** (*str*) – The item name.

**Return type**

*Sequence*[*FilesystemContent*] | None

**abstractmethod package\_history**(\*, *name*, *limit*=3)

Return the package history.

**Parameters**

- **name** (*str*) – The item name.
- **limit** (*int*) – The maximum number of history items to return.

**Return type**

*Sequence*[*Sequence*[*FilesystemContent*]]

**\_abc\_impl** = <\_abc.\_abc\_data object>

**class** annize.features.distributables.common.**Group**(\*, *title*, *files*, *description*, *package\_store*)

Bases: object

A distributable group of package files.

Often this contains only a single file, but for some kinds of packages, there can be multiple files related to each other somehow.

**Parameters**

- **title** (*TrStr*) – The title of this group of package files.
- **files** (*Iterable*[*FilesystemContent*]) – The package files.
- **description** (*TrStr* / *None*) – Additional description text.
- **package\_store** (*PackageStore* / *None*) – An optional package store.

**property title:** *TrStr*

The title of this group of package files.

**files**()

Return the files of this group.

**Return type**

*Sequence*[*FilesystemContent*]

**property description:** *TrStr*

Additional description text.

**property package\_store:** *PackageStore* | *None*

An optional package store.

**\_\_files\_from\_package\_store**()

**Return type**

*Iterable*[*FilesystemContent*] | None

**\_\_store\_files\_to\_package\_store**()

**Return type**

None

```
__package_store_name()
```

**Return type**

str

## annize.features.distributables.debian module

Debian (.deb) packages.

```
class annize.features.distributables.debian.Category(*, debian_name, freedesktop_name)
```

Bases: object

### Parameters

- **debian\_name** (str)
- **freedesktop\_name** (str)

property **debian\_name**: str

property **freedesktop\_name**: str

```
class annize.features.distributables.debian.MenuEntry(*, name, title, category, command, is_gui, icon)
```

Bases: object

### Parameters

- **name** (str)
- **title** (TrStr)
- **category** (Category)
- **command** (str)
- **is\_gui** (bool)
- **icon** (str | Path | FilesystemContent | None)

property **name**: str

property **title**: TrStr

property **category**: Category | None

property **command**: str

property **is\_gui**: bool

property **icon**: FilesystemContent | None

```
class annize.features.distributables.debian.ExecutableLink(*, path, name)
```

Bases: object

### Parameters

- **path** (str)
- **name** (str | None)

property **path**: str

**property name:** `str`

`annize.features.distributables.debian._debian_category(debian_name, freedesktop_name)`

**Parameters**

- **`debian_name`** (*str*)
- **`freedesktop_name`** (*str*)

**Return type**

`type[Category]`

`annize.features.distributables.debian.ApplicationsAccessibilityCategory`

alias of `ACategory`

`annize.features.distributables.debian.ApplicationsAmateurradioCategory`

alias of `ACategory`

`annize.features.distributables.debian.ApplicationsDatamanagementCategory`

alias of `ACategory`

`annize.features.distributables.debian.ApplicationsEditorsCategory`

alias of `ACategory`

`annize.features.distributables.debian.ApplicationsEducationCategory`

alias of `ACategory`

`annize.features.distributables.debian.ApplicationsEmulatorsCategory`

alias of `ACategory`

`annize.features.distributables.debian.ApplicationsFilemanagementCategory`

alias of `ACategory`

`annize.features.distributables.debian.ApplicationsGraphicsCategory`

alias of `ACategory`

`annize.features.distributables.debian.ApplicationsMobiledevicesCategory`

alias of `ACategory`

`annize.features.distributables.debian.ApplicationsNetworkCategory`

alias of `ACategory`

`annize.features.distributables.debian.ApplicationsNetworkCommunicationCategory`

alias of `ACategory`

`annize.features.distributables.debian.ApplicationsNetworkFiletransferCategory`

alias of `ACategory`

`annize.features.distributables.debian.ApplicationsNetworkMonitoringCategory`

alias of `ACategory`

`annize.features.distributables.debian.ApplicationsNetworkWebbrowsingCategory`

alias of `ACategory`

`annize.features.distributables.debian.ApplicationsNetworkWebnewsCategory`

alias of `ACategory`

`annize.features.distributables.debian.ApplicationsOfficeCategory`

alias of `ACategory`

`annize.features.distributables.debian.ApplicationsProgrammingCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsProjectmanagementCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceAstronomyCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceBiologyCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceChemistryCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceDataanalysisCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceElectronicsCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceEngineeringCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceGeoscienceCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceMathematicsCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceMedicineCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsSciencePhysicsCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceSocialCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsShellsCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsSoundsCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsSystemCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsSystemAdministrationCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsSystemHardwareCategory`  
alias of ACategory

`annize.features.distributables.debian.ApplicationsSystemLanguageenvironmentCategory`  
alias of `ACategory`

`annize.features.distributables.debian.ApplicationsSystemMonitoringCategory`  
alias of `ACategory`

`annize.features.distributables.debian.ApplicationsSystemPackagemanagementCategory`  
alias of `ACategory`

`annize.features.distributables.debian.ApplicationsSystemSecurityCategory`  
alias of `ACategory`

`annize.features.distributables.debian.ApplicationsTerminalemulatorsCategory`  
alias of `ACategory`

`annize.features.distributables.debian.ApplicationsTextCategory`  
alias of `ACategory`

`annize.features.distributables.debian.ApplicationsTvandradiocategory`  
alias of `ACategory`

`annize.features.distributables.debian.ApplicationsViewersCategory`  
alias of `ACategory`

`annize.features.distributables.debian.ApplicationsVideoCategory`  
alias of `ACategory`

`annize.features.distributables.debian.ApplicationsWebdevelopmentCategory`  
alias of `ACategory`

`annize.features.distributables.debian.GamesActionCategory`  
alias of `ACategory`

`annize.features.distributables.debian.GamesAdventureCategory`  
alias of `ACategory`

`annize.features.distributables.debian.GamesBlocksCategory`  
alias of `ACategory`

`annize.features.distributables.debian.GamesBoardCategory`  
alias of `ACategory`

`annize.features.distributables.debian.GamesCardCategory`  
alias of `ACategory`

`annize.features.distributables.debian.GamesPuzzlesCategory`  
alias of `ACategory`

`annize.features.distributables.debian.GamesSimulationCategory`  
alias of `ACategory`

`annize.features.distributables.debian.GamesStrategyCategory`  
alias of `ACategory`

`annize.features.distributables.debian.GamesToolsCategory`  
alias of `ACategory`

`annize.features.distributables.debian.GamesToysCategory`  
alias of `ACategory`

`annize.features.distributables.debian.HelpCategory`  
alias of `ACategory`

`annize.features.distributables.debian.ScreenSavingCategory`  
alias of `ACategory`

`annize.features.distributables.debian.ScreenLockingCategory`  
alias of `ACategory`

**class** `annize.features.distributables.debian.Section(*, name)`  
Bases: `object`  
**property** `name`: `str`

`annize.features.distributables.debian._debian_section(name)`

**Parameters**  
**name** (*str*)

**Return type**  
*Type[Section]*

`annize.features.distributables.debian.AdministrationUtilitiesSection`  
alias of `ASection`

`annize.features.distributables.debian.MonoCliSection`  
alias of `ASection`

`annize.features.distributables.debian.CommunicationProgramsSection`  
alias of `ASection`

`annize.features.distributables.debian.DatabasesSection`  
alias of `ASection`

`annize.features.distributables.debian.DebianInstallerUdebPackagesSection`  
alias of `ASection`

`annize.features.distributables.debian.DebugPackagesSection`  
alias of `ASection`

`annize.features.distributables.debian.DevelopmentSection`  
alias of `ASection`

`annize.features.distributables.debian.DocumentationSection`  
alias of `ASection`

`annize.features.distributables.debian.EditorsSection`  
alias of `ASection`

`annize.features.distributables.debian.EducationSection`  
alias of `ASection`

`annize.features.distributables.debian.ElectronicsSection`  
alias of `ASection`

`annize.features.distributables.debian.EmbeddedSoftwareSection`

alias of ASection

`annize.features.distributables.debian.FontsSection`

alias of ASection

`annize.features.distributables.debian.GamesSection`

alias of ASection

`annize.features.distributables.debian.GnomeSection`

alias of ASection

`annize.features.distributables.debian.GnuRSection`

alias of ASection

`annize.features.distributables.debian.GnustepSection`

alias of ASection

`annize.features.distributables.debian.GraphicsSection`

alias of ASection

`annize.features.distributables.debian.HamRadioSection`

alias of ASection

`annize.features.distributables.debian.HaskellSection`

alias of ASection

`annize.features.distributables.debian.WebServersSection`

alias of ASection

`annize.features.distributables.debian.InterpretersSection`

alias of ASection

`annize.features.distributables.debian.IntrospectionSection`

alias of ASection

`annize.features.distributables.debian.JavaSection`

alias of ASection

`annize.features.distributables.debian.JavascriptSection`

alias of ASection

`annize.features.distributables.debian.KdeSection`

alias of ASection

`annize.features.distributables.debian.KernelsSection`

alias of ASection

`annize.features.distributables.debian.LibraryDevelopmentSection`

alias of ASection

`annize.features.distributables.debian.LibrariesSection`

alias of ASection

`annize.features.distributables.debian.LispSection`

alias of ASection

`annize.features.distributables.debian.LanguagePacksSection`

alias of ASection



`annize.features.distributables.debian.MailSection`  
alias of ASection

`annize.features.distributables.debian.MathematicsSection`  
alias of ASection

`annize.features.distributables.debian.MetaPackagesSection`  
alias of ASection

`annize.features.distributables.debian.MiscellaneousSection`  
alias of ASection

`annize.features.distributables.debian.NetworkSection`  
alias of ASection

`annize.features.distributables.debian.NewsgroupsSection`  
alias of ASection

`annize.features.distributables.debian.OcamlSection`  
alias of ASection

`annize.features.distributables.debian.OldLibrariesSection`  
alias of ASection

`annize.features.distributables.debian.OtherOSsAndFSsSection`  
alias of ASection

`annize.features.distributables.debian.PperlSection`  
alias of ASection

`annize.features.distributables.debian.PhpSection`  
alias of ASection

`annize.features.distributables.debian.PythonSection`  
alias of ASection

`annize.features.distributables.debian.RubySection`  
alias of ASection

`annize.features.distributables.debian.RustSection`  
alias of ASection

`annize.features.distributables.debian.ScienceSection`  
alias of ASection

`annize.features.distributables.debian.ShellsSection`  
alias of ASection

`annize.features.distributables.debian.SoundSection`  
alias of ASection

`annize.features.distributables.debian.TasksSection`  
alias of ASection

`annize.features.distributables.debian.TexSection`  
alias of ASection

`annize.features.distributables.debian.TextProcessingSection`  
alias of ASection

`annize.features.distributables.debian.UtilitiesSection`

alias of `ASection`

`annize.features.distributables.debian.VersionControlSystemsSection`

alias of `ASection`

`annize.features.distributables.debian.VideoSection`

alias of `ASection`

`annize.features.distributables.debian.VirtualPackagesSection`

alias of `ASection`

`annize.features.distributables.debian.WebSoftwareSection`

alias of `ASection`

`annize.features.distributables.debian.XWindowSystemSoftwareSection`

alias of `ASection`

`annize.features.distributables.debian.XfceSection`

alias of `ASection`

`annize.features.distributables.debian.ZopePloneFrameworkSection`

alias of `ASection`

**class** `annize.features.distributables.debian.ServiceDescription`(*name, command*)

Bases: `object`

Description for Debian services to be included in a package.

#### Parameters

- **name** (*str*) – The display name.
- **command** (*str*) – The command to be executed.

**class** `annize.features.distributables.debian.Package`(\*, *source, menuentries, executable\_links, packagename, description, summary, section, homepage\_url, version, documentation, authors, prerm="", postinst="", architecture='all'*)

Bases: `FilesystemContent`

#### Parameters

- **generate\_func** – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).
- **source** (`FilesystemContent`)
- **menuentries** (*list*[`MenuEntry`])
- **executable\_links** (*list*[`ExecutableLink`])
- **packagename** (*str* | *None*)
- **description** (`TrStr` | *None*)
- **summary** (`TrStr` | *None*)
- **section** (`Section` | *None*)
- **homepage\_url** (*str* | *None*)
- **version** (`Version` | *None*)

- **documentation** (`FileSystemContent` | `None`)
- **authors** (`list[Author]`)
- **prerm** (`str`)
- **postinst** (`str`)
- **architecture** (`str`)

**\_path()**

```
class _BuildInfo(source: annize.fs.FileSystemContent, executable_links:
    list[annize.features.distributables.debian.ExecutableLink], menuentries:
    list[annize.features.distributables.debian.MenuEntry], services:
    list[annize.features.distributables.debian.ServiceDescription], description: str, name:
    str, version: annize.data.version.Version, homepage: str, author:
    annize.features.authors.Author, licensename: str, section:
    annize.features.distributables.debian.Section | None, summary: str,
    documentation_source: annize.fs.FileSystemContent | None, prerm: str, postinst: str,
    architecture: str, authorstring: str = None, pkgrootpath: str = None, pkgpath_debian:
    str = None, pkgpath_documentation: str = None, pkgpath_pixmap: str = None,
    pkgpath_usrbin: str = None, config_files: list[str] = None, pkgsize: int = None, result:
    annize.fs.FileSystemContent = None)
```

Bases: `object`

#### Parameters

- **source** (`FileSystemContent`)
- **executable\_links** (`list[ExecutableLink]`)
- **menuentries** (`list[MenuEntry]`)
- **services** (`list[ServiceDescription]`)
- **description** (`str`)
- **name** (`str`)
- **version** (`Version`)
- **homepage** (`str`)
- **author** (`Author`)
- **licensename** (`str`)
- **section** (`Section` | `None`)
- **summary** (`str`)
- **documentation\_source** (`FileSystemContent` | `None`)
- **prerm** (`str`)
- **postinst** (`str`)
- **architecture** (`str`)
- **authorstring** (`str`)
- **pkgrootpath** (`str`)
- **pkgpath\_debian** (`str`)
- **pkgpath\_documentation** (`str`)

- `pkgpath_pixmap`(*str*)
- `pkgpath_usrbin`(*str*)
- `config_files`(*list[str]*)
- `pkgsize`(*int*)
- `result`(`FilesystemContent`)

`source`: `FilesystemContent`

`executable_links`: `list[ExecutableLink]`

`menuentries`: `list[MenuEntry]`

`services`: `list[ServiceDescription]`

`description`: `str`

`name`: `str`

`version`: `Version`

`homepage`: `str`

`author`: `Author`

`licensename`: `str`

`section`: `Section` | `None`

`summary`: `str`

`documentation_source`: `FilesystemContent` | `None`

`prerm`: `str`

`postinst`: `str`

`architecture`: `str`

`authorstring`: `str` = `None`

`pkgrootpath`: `str` = `None`

`pkgpath_debian`: `str` = `None`

`pkgpath_documentation`: `str` = `None`

`pkgpath_pixmap`: `str` = `None`

`pkgpath_usrbin`: `str` = `None`

`config_files`: `list[str]` = `None`

`pkgsize`: `int` = `None`

`result`: `FilesystemContent` = `None`

**classmethod** `_mkpackage(build)`

**Parameters**

`build` (`_BuildInfo`)

**Return type**

`FileSystemContent`

**classmethod** `_mkpackage_prepareinfos(build, tmpdir)`

**Parameters**

- `build` (`_BuildInfo`)
- `tmpdir` (`str` | `Path`)

**Return type**

`None`

**classmethod** `_mkpackage_mkcopyright(build)`

**Parameters**

`build` (`_BuildInfo`)

**Return type**

`None`

**classmethod** `_mkpackage_mkchangelog(build)`

**Parameters**

`build` (`_BuildInfo`)

**Return type**

`None`

**classmethod** `_mkpackage_mkexeclinks(build)`

**Parameters**

`build` (`_BuildInfo`)

**Return type**

`None`

**classmethod** `_mkpackage_mkmenuentries(build)`

**Parameters**

`build` (`_BuildInfo`)

**Return type**

`None`

**classmethod** `_mkpackage_mkservices(build)`

**Parameters**

`build` (`_BuildInfo`)

**Return type**

`None`

**classmethod** `_mkpackage_determinesize(build)`

**Parameters**

`build` (`_BuildInfo`)

**Return type**

None

**classmethod** `_mkpackage_mkprepostcmds(build)`**Parameters**`build` (`_BuildInfo`)**Return type**

None

**classmethod** `_mkpackage_mkdebiancontrolfile(build)`**Parameters**`build` (`_BuildInfo`)**Return type**

None

**classmethod** `_mkpackage_mkdebianconffilesfile(build)`**Parameters**`build` (`_BuildInfo`)**Return type**

None

**classmethod** `_mkpackage_correctbuildsourcepermissions(build)`**Parameters**`build` (`_BuildInfo`)**Return type**

None

**classmethod** `_mkpackage_dpkg(build)`**Parameters**`build` (`_BuildInfo`)**Return type**

None

**annize.features.distributables.flatpak module**

Flatpaks.

**class** `annize.features.distributables.flatpak.MenuEntry(*, name, title, category, command, is_gui, icon)`Bases: `object`**Parameters**

- **name** (`str`)
- **title** (`TrStr`)
- **category** (`Tuple`)
- **command** (`str`)
- **is\_gui** (`bool`)
- **icon** (`FileSystemContent` / `None`)

```

    property name: str
    property title: TrStr
    property category: Tuple
    property command: str
    property is_gui: bool
    property icon: FilesystemContent | None

class annize.features.distributables.flatpak.Filesystem(**b)
    Bases: object

class annize.features.distributables.flatpak.Share(**b)
    Bases: object

class annize.features.distributables.flatpak.EnvironmentVariable(**b)
    Bases: object

class annize.features.distributables.flatpak.Repository(*, public_url, friendly_name_suggestion)
    Bases: object
        Parameters
            • public_url (str)
            • friendly_name_suggestion (str | None)

    upload(source)

        Parameters
            source (FilesystemContent)

    property public_url: str
    property friendly_name_suggestion: str

class annize.features.distributables.flatpak.LocalRepository(*, public_url,
                                                             friendly_name_suggestion,
                                                             upload_path, upload_fsentry)
    Bases: Repository
        Parameters
            • public_url (str)
            • friendly_name_suggestion (str | None)
            • upload_path (str | None)
            • upload_fsentry (FilesystemContent | None)

    upload(source)

        Parameters
            source (FilesystemContent)

```

```
class annize.features.distributables.flatpak.Group(*, source, title, description, repository,
                                                    package_name, project_short_hint_name,
                                                    package_short_name)
```

Bases: [Group](#)

#### Parameters

- **title** (*str*) – The title of this group of package files.
- **files** – The package files.
- **description** ([TrStr](#) / *None*) – Additional description text.
- **package\_store** – An optional package store.
- **source** ([FilesystemContent](#))
- **repository** ([Repository](#))
- **package\_name** (*str*)
- **project\_short\_hint\_name** (*str* / *None*)
- **package\_short\_name** (*str* / *None*)

#### files()

Return the files of this group.

#### property description

Additional description text.

```
class annize.features.distributables.flatpak.FlatpakrefFile(*, refname, package_name, title,
                                                            branch='master',
                                                            runtime_repository_url, gpgkey,
                                                            repository)
```

Bases: [FilesystemContent](#)

#### Parameters

- **generate\_func** – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).
- **refname** (*str* / *None*)
- **package\_name** (*str*)
- **title** (*str* / *None*)
- **branch** (*str*)
- **runtime\_repository\_url** (*str* / *None*)
- **gpgkey** (*bytes* / *None*)
- **repository** ([Repository](#))

#### \_path()

```
class annize.features.distributables.flatpak.GpgFile(*, refname=None, gpgkey=None)
```

Bases: [FilesystemContent](#)

#### Parameters

- **generate\_func** – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).



- **refname** (*str*)
- **gpgkey** (*bytes*)

**\_path()**

```
class annize.features.distributables.flatpak.FlatpakImage(*, source, package_name,
                                                         sockets=('x11'), filesystems=('home'),
                                                         shares=('network'))
```

Bases: [FilesystemContent](#)

#### Parameters

- **generate\_func** – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).
- **source** ([FilesystemContent](#))
- **package\_name** (*str*)
- **sockets** (*Iterable[str]*)
- **filesystems** (*Iterable[str]*)
- **shares** (*Iterable[str]*)

**\_path()**

```
class _BuildInfo(source: annize.fs.FilesystemContent, name: str, sdk: str, platform: str, kitversion: str |
                 None, command: str | None, sockets: Iterable[str], filesystems: Iterable[str], shares:
                 Iterable[str], menu_entries: Iterable[annize.features.distributables.flatpak.MenuEntry],
                 environment: dict[str, str], pkgrootpath: str = None, pkgpath_share: str = None,
                 pkgpath_share_applications: str = None, pkgpath_share_icons: str = None, result:
                 annize.fs.FilesystemContent = None)
```

Bases: `object`

#### Parameters

- **source** ([FilesystemContent](#))
- **name** (*str*)
- **sdk** (*str*)
- **platform** (*str*)
- **kitversion** (*str* | *None*)
- **command** (*str* | *None*)
- **sockets** (*Iterable[str]*)
- **filesystems** (*Iterable[str]*)
- **shares** (*Iterable[str]*)
- **menu\_entries** (*Iterable[MenuEntry]*)
- **environment** (*dict[str, str]*)
- **pkgrootpath** (*str*)
- **pkgpath\_share** (*str*)
- **pkgpath\_share\_applications** (*str*)
- **pkgpath\_share\_icons** (*str*)

```
    • result(FilesystemContent)
source: FilesystemContent
name: str
sdk: str
platform: str
kitversion: str | None
command: str | None
sockets: Iterable[str]
filesystems: Iterable[str]
shares: Iterable[str]
menu_entries: Iterable[MenuEntry]
environment: dict[str, str]
pkgrootpath: str = None
pkgpath_share: str = None
pkgpath_share_applications: str = None
pkgpath_share_icons: str = None
result: FilesystemContent = None
classmethod _mkpackage_prepareinfos(build, tmpdir)

    Parameters
        • build(_BuildInfo)
        • tmpdir(Path)

    Return type
        None
classmethod _mkpackage_flatpak_build_init(build)

    Parameters
        build(_BuildInfo)

    Return type
        None
classmethod _mkpackage_applysource(build)

    Parameters
        build(_BuildInfo)

    Return type
        None
```

**classmethod** `_mkpackage_flatpak_build_finish(build)`

**Parameters**

`build` (`_BuildInfo`)

**Return type**

None

**classmethod** `_mkpackage_share(build)`

**Parameters**

`build` (`_BuildInfo`)

**Return type**

None

**classmethod** `_mkpackage_flatpak_build_export(build)`

**Parameters**

`build` (`_BuildInfo`)

**Return type**

None

**classmethod** `_mkpackage(build)`

**Parameters**

`build` (`_BuildInfo`)

**Return type**

`Path`

## annize.features.distributables.python\_wheel module

Python Wheels.

**class** `annize.features.distributables.python_wheel.ExecutableLink(*, linkname, modulename, methodname, is_gui)`

Bases: `object`

**Parameters**

- `linkname` (`str`)
- `modulename` (`str`)
- `methodname` (`str`)
- `is_gui` (`bool`)

**property** `linkname`: `str`

**property** `modulename`: `str`

**property** `methodname`: `str`

**property** `is_gui`: `bool`

**class** `annize.features.distributables.python_wheel.Package(*, source, executable_links, packagename, description, homepage_url, long_description, version, keywords, dependencies, license, authors)`

Bases: [FilesystemContent](#)

#### Parameters

- **generate\_func** – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).
- **source** ([FilesystemContent](#))
- **executable\_links** ([Iterable](#)[[ExecutableLink](#)])
- **packagename** ([str](#) | [None](#))
- **description** ([TrStr](#) | [None](#))
- **homepage\_url** ([str](#) | [None](#))
- **long\_description** ([TrStr](#) | [None](#))
- **version** ([Version](#) | [None](#))
- **keywords** ([Keywords](#) | [None](#))
- **dependencies** ([Iterable](#)[[PythonPackage](#)])
- **license** ([License](#) | [None](#))
- **authors** ([Iterable](#)[[Author](#)])

[\\_path\(\)](#)

```
class _BuildInfo(source: annize.fs.FilesystemContent, description: str, long_description: str, keywords:
    annize.features.base.Keywords, name: str, version: annize.data.version.Version,
    homepage: str, author: annize.features.authors.Author, license:
    annize.features.licensing.License, executable_links:
    list[annize.features.distributables.python_wheel.ExecutableLink], dependencies: list,
    pkgrootpath: str = None, pkgpath_setuppy: str = None, setuppy_conf: dict[str, Any |
    None] = None, result: annize.fs.FilesystemContent = None)
```

Bases: [object](#)

#### Parameters

- **source** ([FilesystemContent](#))
- **description** ([str](#))
- **long\_description** ([str](#))
- **keywords** ([Keywords](#))
- **name** ([str](#))
- **version** ([Version](#))
- **homepage** ([str](#))
- **author** ([Author](#))
- **license** ([License](#))
- **executable\_links** ([list](#)[[ExecutableLink](#)])
- **dependencies** ([list](#))
- **pkgrootpath** ([str](#))
- **pkgpath\_setuppy** ([str](#))

```

        • setuppy_conf(dict[str, Any | None])
        • result(FilesystemContent)
source: FilesystemContent
description: str
long_description: str
keywords: Keywords
name: str
version: Version
homepage: str
author: Author
license: License
executable_links: list[ExecutableLink]
dependencies: list
pkgrootpath: str = None
pkgpath_setuppy: str = None
setuppy_conf: dict[str, Any | None] = None
result: FilesystemContent = None

classmethod _mkpackage(build)

    Parameters
        build (_BuildInfo)

    Return type
        FilesystemContent

classmethod _mkpackage_prepareinfos(build, tmpdir)

    Parameters
        • build (_BuildInfo)
        • tmpdir (Path)

    Return type
        None

classmethod _mkpackage_setuppyconf_prepare(build)

    Parameters
        build (_BuildInfo)

    Return type
        None

```

**classmethod** `_mkpackage_setuptoolsconf_install_requires`(*build*)

**Parameters**

`build` (`_BuildInfo`)

**Return type**

None

**classmethod** `_mkpackage_setuptoolsconf_classifiers`(*build*)

**Parameters**

`build` (`_BuildInfo`)

**Return type**

None

**classmethod** `_mkpackage_mkexeclinks`(*build*)

**Parameters**

`build` (`_BuildInfo`)

**Return type**

None

**classmethod** `_mkpackage_mksetuptoolsconf`(*build*)

**Parameters**

`build` (`_BuildInfo`)

**Return type**

None

**classmethod** `_mkpackage_bdist_wheel`(*build*)

**Parameters**

`build` (`_BuildInfo`)

**Return type**

None

**classmethod** `_mkpackage_mkmanifestin`(*build*)

**Parameters**

`build` (`_BuildInfo`)

**Return type**

None

## **annize.features.distributables.tar module**

Tarballs.

**class** `annize.features.distributables.tar.Package`(\*, *packagename*, *packagenamepostfix*, *source*, *version*, *license*, *documentation*)

Bases: `FilesystemContent`

**Parameters**

- **generate\_func** – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).
- **packagename** (*str* | *None*)

- **packagenamepostfix** (*str* / *None*)
- **source** (*FileSystemContent*)
- **version** (*Version* / *None*)
- **license** (*License* / *None*)
- **documentation** (*FileSystemContent* / *None*)

**\_path()**

**annize.features.documentation namespace**

**Subpackages**

**annize.features.documentation.sphinx namespace**

**Subpackages**

**annize.features.documentation.sphinx.output namespace**

**Submodules**

**annize.features.documentation.sphinx.output.common module**

Sphinx-based documentation output.

**annize.features.documentation.sphinx.output.common.register\_output\_generator**(*outputgenerator*)

**annize.features.documentation.sphinx.output.common.find\_output\_generator\_for\_outputspec**(*outputspec*)

**class annize.features.documentation.sphinx.output.common.OutputGenerator**(*outputspec*)

Bases: ABC

Base class for documentation output specifications. See **render()**.

**property outputspec:** *OutputSpec*

**abstractmethod classmethod is\_compatible\_for**(*outputspec*)

**Parameters**

**outputspec** (*OutputSpec*)

**Return type**

bool

**abstractmethod formatname**()

Returns the Sphinx format name.

**Return type**

str

**prepare\_generate**(*geninfo*)

**Parameters**

**geninfo** (*documentationsphinx.Document.GenerateInfo*)

**Return type**

None

**postproc**(*preresult*)

Parameters

**preresult** ([FilesystemContent](#))

Return type

[FilesystemContent](#)

**multilanguage\_frame**(*document*)

Parameters

**document** ([documentationsphinx.Document](#))

Return type

[DocumentGenerateAllCulturesResult](#)

**\_abc\_impl** = <**\_abc.\_abc\_data** object>

## **annize.features.documentation.sphinx.output.html module**

Sphinx-based HTML documentation output.

```
class annize.features.documentation.sphinx.output.html.HtmlOutputSpec(* , is_homepage=False,
                                                                    short_title=None,
                                                                    short_desc=None,
                                                                    theme=None,
                                                                    masterlink=None,
                                                                    logo_image=None, back-
                                                                    ground_image=None)
```

Bases: [HtmlOutputSpec](#)

Parameters

- **theme** (*str* / *None*) – The sphinx html theme name.
- **short\_title** ([TrStr](#) / *None*) – The short html title.
- **short\_desc** ([TrStr](#) / *None*) – The short description. Ignored by most themes.
- **masterlink** (*str* / *None*) – Url that overrides the target of the main heading (which is also a link).
- **is\_homepage** (*bool*)
- **logo\_image** (*str* / *Path* / [FilesystemContent](#) / *None*)
- **background\_image** (*str* / *Path* / [FilesystemContent](#) / *None*)

property **short\_title**: [TrStr](#) | *None*

property **short\_desc**: [TrStr](#) | *None*

property **theme**: *str* | *None*

property **masterlink**: *str* | *None*

property **logo\_image**: [FilesystemContent](#) | *None*

property **background\_image**: [FilesystemContent](#) | *None*

**\_abc\_impl** = <**\_abc.\_abc\_data** object>



```

class annize.features.documentation.sphinx.output.html.HtmlOutputGenerator(outputspec)
    Bases: OutputGenerator
    HTML documentation output.
    classmethod is_compatible_for(outputspec)
    formatname()
        Returns the Sphinx format name.
    prepare_generate(geninfo)
    multilanguage_frame(document)
    _abc_impl = <_abc._abc_data object>

```

### annize.features.documentation.sphinx.output.pdf module

Sphinx-based PDF documentation output.

```

class annize.features.documentation.sphinx.output.pdf.PdfOutputGenerator(outputspec)
    Bases: OutputGenerator
    PDF documentation output.
    classmethod is_compatible_for(outputspec)
    formatname()
        Returns the Sphinx format name.
    postproc(preresult)
        Parameters
            preresult (str | Path)
        Return type
            Path
    _abc_impl = <_abc._abc_data object>

```

### annize.features.documentation.sphinx.output.plaintext module

Sphinx-based plaintext documentation output.

```

class annize.features.documentation.sphinx.output.plaintext.PlaintextOutputGenerator(outputspec)
    Bases: OutputGenerator
    Plaintext documentation output.
    classmethod is_compatible_for(outputspec)
    formatname()
        Returns the Sphinx format name.
    _abc_impl = <_abc._abc_data object>

```

## Submodules

### annize.features.documentation.sphinx.\_utils module

Internal Sphinx utilities.

`annize.features.documentation.sphinx._utils.serialize_for_confpy(obj)`

#### Parameters

`obj` (*Any*)

#### Return type

`str`

### annize.features.documentation.sphinx.common module

Sphinx-based documentation.

`class annize.features.documentation.sphinx.common.Document(*, project_name, title, authors, version, release)`

Bases: *Document*, ABC

#### Parameters

- `project_name` (*TrStr* / *None*)
- `title` (*TrStr* / *None*)
- `authors` (*list*[*Author*])
- `version` (*Version* / *None*)
- `release` (*TrStr* / *None*)

`class GenerateInfo(main_document: 'Document', intstruct: annize.fs.Path, outdir: annize.fs.FilesystemContent, confdir: annize.fs.FilesystemContent, culture: annize.i18n.Culture, configvalues: dict, configlines: list, entry_path: str = "")`

Bases: *object*

#### Parameters

- `main_document` (*Document*)
- `intstruct` (*Path*)
- `outdir` (*FilesystemContent*)
- `confdir` (*FilesystemContent*)
- `culture` (*Culture*)
- `configvalues` (*dict*)
- `configlines` (*list*)
- `entry_path` (*str*)

`main_document:` *Document*

`intstruct:` *Path*

`outdir:` *FilesystemContent*

`confdir:` *FilesystemContent*

```

    culture: Culture
    configvalues: dict
    configlines: list
    entry_path: str = ''
    _to_dict()

abstractmethod _generate_sources(geninfo)

    Parameters
        geninfo (GenerateInfo)

    Return type
        str

property projectname__original: TrStr | None
property project_name: TrStr
property title__original: TrStr | None
property title: TrStr
property authors: list[Author]
property version: Version | None
property release: TrStr | None
generate_all_cultures(outputspec)
generate(outputspec, *, culture=None)
__generate_set_misc(geninfo)
__generate_prepare_shortsnippets()
__generate_prepare_annizeicons()
__generate_set_culture()
__generate_set_version_and_release(geninfo)
__generate_geninfo_to_confpy()
_abc_impl = <_abc._abc_data object>

class annize.features.documentation.sphinx.common.CompositeDocument(*, documents, **kwargs)
    Bases: Document

    Parameters
        documents (Iterable[Document])

    __get_inner_generateinfo(innerdoc)

    Parameters
        • geninfo (GenerateInfo)
        • innerdoc (Document)

```

```
_generate_sources(geninfo)
```

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.features.documentation.sphinx.common.ApiReferenceLanguage
```

Bases: `ABC`

Base class for a programming language in api references. See `ApiReferencePiece`.

```
class ApiReferenceGenerateInfo(geninfo, source, heading)
```

Bases: `GenerateInfo`

#### Parameters

- **geninfo** (`GenerateInfo`)
- **source** (`FilesystemContent`)
- **heading** (`TrStr`)

property **source**: `FilesystemContent`

property **heading**: `TrStr`

```
abstractmethod generate_sources(rgeninfo)
```

#### Parameters

**rgeninfo** (`ApiReferenceGenerateInfo`)

#### Return type

`str`

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.features.documentation.sphinx.common.ApiReferenceDocument(*, language, heading,  
                                                                    source, cultures,  
                                                                    **kwargs)
```

Bases: `Document`

An api reference.

#### Parameters

- **language** (`ApiReferenceLanguage`)
- **heading** (`TrStr` | `None`)
- **source** (`str` | `Path` | `FilesystemContent`)
- **cultures** (`list[Culture]`)

```
available_cultures()
```

```
__get_refgeninfo(geninfo)
```

#### Parameters

**geninfo** (`GenerateInfo`)

```
_generate_sources(geninfo)
```

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.features.documentation.sphinx.common.ArgparseCommandLineInterfaceDocument(*,
                                                                                       parser_factory,
                                                                                       pro-
                                                                                       gram_name,
                                                                                       head-
                                                                                       ing,
                                                                                       source,
                                                                                       cul-
                                                                                       tures,
                                                                                       **kwargs)
```

Bases: *Document*

#### Parameters

- **parser\_factory** (*str*)
- **program\_name** (*str*)
- **heading** (*str* | *None*)
- **source** (*str* | *Path* | *FilesystemContent*)
- **cultures** (*list*[*Culture*])

**available\_cultures**()

**\_generate\_sources**(*geninfo*)

**\_abc\_impl** = *<\_abc.\_abc\_data object>*

```
class annize.features.documentation.sphinx.common.RstDocumentVariant(*, cul-
                                                                                       ture=<annize.i18n.UnspecifiedCulture
                                                                                       object>, source)
```

Bases: *object*

#### Parameters

- **culture** (*Culture*)
- **source** (*str* | *Path* | *FilesystemContent*)

**property culture**: *Culture*

**property source**: *FilesystemContent*

```
class annize.features.documentation.sphinx.common.RstDocument(*, variants, **kwargs)
```

Bases: *Document*

A reStructuredText formatted file or a directory of such files.

#### Parameters

**variants** (*list*[*RstDocumentVariant*])

**available\_cultures**()

**\_\_get\_variant**(*culture*)

#### Parameters

**culture** (*Culture*)

#### Return type

*RstDocumentVariant* | *None*

```
_generate_sources(geninfo)
```

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.features.documentation.sphinx.common.AboutProjectDocument(*, dependencies,  
                                                                    cultures, **kwargs)
```

Bases: *Document*

**Parameters**

- **dependencies** (*list*[*Dependency*])
- **cultures** (*list*[*Culture*])

```
available_cultures()
```

```
_generate_sources(geninfo)
```

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.features.documentation.sphinx.common.ReadmeDocument(*, project_name, title, authors,  
                                                                    version, release, documents,  
                                                                    dependencies, cultures)
```

Bases: *CompositeDocument*

A reStructuredText formatted file or a directory of such files.

**Parameters**

- **documents** (*Iterable*[*Document*])
- **dependencies** (*list*[*Dependency*])
- **cultures** (*list*[*Culture*])

```
_ABOUT_NAME = 'annize..about'
```

```
_abc_impl = <_abc._abc_data object>
```

```
available_cultures()
```

```
property title
```

## annize.features.documentation.sphinx.cpp module

Sphinx-based C/C++ documentation.

```
class annize.features.documentation.sphinx.cpp.CppApiReferenceLanguage(**kwargs)
```

Bases: *DoxygenSupportedApiReferenceLanguage*

C++ language support for api references.

```
_abc_impl = <_abc._abc_data object>
```

## annize.features.documentation.sphinx.doxygen\_compat module

Sphinx-based documentation with Doxygen support.

```

class annize.features.documentation.sphinx.doxygen_compat.DoxygenSupportedApiReferenceLanguage(*,
    _doxygenopts=dict[str, str] | None,
    extract_all=bool,
    extract_private=bool,
    extract_package=bool,
    extract_static=bool,
    extract_local_classes=bool,
    extract_local_methods=bool,
    extract_anon_nspaces=bool,
    extract_priv_virtual=bool,
    file_patterns=str,
    inline_inherited_members=bool,
    inherit_docs=bool,
    hide_undocumented_members=bool,
    exclude_patterns=str,
    pre_declarations=None,
    )

```

Bases: [ApiReferenceLanguage](#)

Language support for api references powered by Doxygen.

#### Parameters

- **\_doxygenopts** (*dict[str, str] | None*)
- **extract\_all** (*bool*)
- **extract\_private** (*bool*)
- **extract\_package** (*bool*)
- **extract\_static** (*bool*)
- **extract\_local\_classes** (*bool*)
- **extract\_local\_methods** (*bool*)
- **extract\_anon\_nspaces** (*bool*)
- **extract\_priv\_virtual** (*bool*)
- **file\_patterns** (*str*)
- **inline\_inherited\_members** (*bool*)
- **inherit\_docs** (*bool*)
- **hide\_undocumented\_members** (*bool*)

- **hide\_undoc\_classes** (*bool*)
- **exclude\_patterns** (*str*)
- **predefined** (*list[str] | None*)

**\_\_info**(*outpath*)

**\_\_prepare\_generate**(*rootpath, outpath*)

**generate\_sources**(*srcpath, intstructpath, confdirpath, heading*)

**\_abc\_impl** = <\_abc.\_abc\_data object>

### annize.features.documentation.sphinx.javascript module

Sphinx-based Javascript documentation.

**class** annize.features.documentation.sphinx.javascript.**JavaScriptApiReferenceLanguage**

Bases: [ApiReferenceLanguage](#)

JavaScript language support for api references.

**\_\_jsfiles**(*dirf*)

**Parameters**

**dirf** (*str*)

**Return type**

*list[str]*

**\_\_scanjsfile**(*f*)

**Parameters**

**f** (*str*)

**Return type**

*str*

**generate\_sources**(*rgeinfo*)

**\_abc\_impl** = <\_abc.\_abc\_data object>

### annize.features.documentation.sphinx.python module

Sphinx-based Python documentation.

**class** annize.features.documentation.sphinx.python.**Python3ApiReferenceLanguage**(\*,  
*show\_undoc\_members=True,*  
*show\_protected\_members=True*)

Bases: [ApiReferenceLanguage](#)

Python 3 language support for api references.

**Parameters**

- **show\_undoc\_members** (*bool*)
- **show\_protected\_members** (*bool*)



```
__patch_property_types_in_docstrings(pdir)
```

**Parameters**

**pdir** (*str*)

**Return type**

None

```
generate_sources(rgeinfo)
```

```
_abc_impl = <_abc._abc_data object>
```

## annize.features.documentation.sphinx.rst module

Sphinx-based reStructuredText documentation.

```
class annize.features.documentation.sphinx.rst.RstGenerator
```

Bases: object

Generator for some documentation source parts.

```
static heading(text, *, variant='=', sub=False, anchor=None)
```

Generates documentation source for a section heading.

**Parameters**

- **text** (*TrStr* | *str*)
- **variant** (*str*)
- **sub** (*bool*)
- **anchor** (*str* | *None*)

**Return type**

str

## Submodules

### annize.features.documentation.common module

Documentation.

```
class annize.features.documentation.common.DocumentGenerateResult(file, entry_path)
```

Bases: object

**Parameters**

- **file** (*FilesystemContent*)
- **entry\_path** (*str*)

**property file:** *FilesystemContent*

**property entry\_path:** *str*

```
_set_entry_path(entry_path)
```

**Parameters**

**entry\_path** (*str*)

**Return type**

None

```
class annize.features.documentation.common.DocumentGenerateAllCulturesResult(file, entry_path,
                                                                              en-
                                                                              try_paths_for_languages)
```

Bases: [DocumentGenerateResult](#)

**Parameters**

- **file** ([FilesystemContent](#))
- **entry\_path** (*str*)
- **entry\_paths\_for\_languages** (*dict[str, str]*)

**entry\_path\_for\_language** (*language*)

**Parameters**

**language** (*str*)

**Return type**

*str*

**property languages:** *list[str]*

```
class annize.features.documentation.common.Document
```

Bases: [ABC](#)

**abstractmethod available\_cultures()**

**Return type**

*list[[Culture](#)]*

**abstractmethod generate**(*outputspec, \*, culture=<annize.i18n.UnspecifiedCulture object>*)

**Parameters**

**culture** ([Culture](#))

**Return type**

[DocumentGenerateResult](#)

**abstractmethod generate\_all\_cultures**(*outputspec*)

**Return type**

[DocumentGenerateAllCulturesResult](#)

**\_abc\_impl** = *<\_abc.\_abc\_data object>*

```
class annize.features.documentation.common.OutputSpec
```

Bases: [ABC](#)

Base class for documentation output specifications. See `render()`.

**\_abc\_impl** = *<\_abc.\_abc\_data object>*

```
class annize.features.documentation.common.HtmlOutputSpec(*, is_homepage=False)
```

Bases: [OutputSpec](#)

HTML documentation output.

**Parameters**

**is\_homepage** (*bool*) – If to render output for a homepage with slight different stylings and behavior.

**property** `is_homepage`

`_abc_impl = <_abc._abc_data object>`

**class** `annize.features.documentation.common.PdfOutputSpec`

Bases: `OutputSpec`

PDF documentation output.

`_abc_impl = <_abc._abc_data object>`

**class** `annize.features.documentation.common.PlaintextOutputSpec`

Bases: `OutputSpec`

Plaintext documentation output.

`_abc_impl = <_abc._abc_data object>`

**class** `annize.features.documentation.common.GeneratedDocument(*, document, outputspec, culture, filename)`

Bases: `FilesystemContent`

#### Parameters

- **generate\_func** – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).
- **document** (`Document`)
- **outputspec** (`OutputSpec`)
- **culture** (`Culture` | `None`)
- **filename** (`str` | `None`)

`_path()`

**annize.features.files namespace**

**Subpackages**

**annize.features.files.transfer namespace**

**Submodules**

**annize.features.files.transfer.common module**

File transfers.

**class** `annize.features.files.transfer.common.Endpoint`

Bases: `ABC`

**abstractmethod** `access_filesystem(rootpath)`

#### Parameters

**rootpath** (`str`)

#### Return type

`ContextManager[Path, bool | None]`

`_abc_impl = <_abc._abc_data object>`

```
class annize.features.files.transfer.common.FsEndpoint(*, fsentry, path)
```

Bases: [Endpoint](#)

**Parameters**

- **fsentry** ([Path](#) | *None*)
- **path** (*str* | *None*)

```
access_filesystem(rootpath)
```

**Parameters**

**rootpath** (*str*)

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.features.files.transfer.common.Upload(*, source, destination_endpoint,  
                                                    destination_path)
```

Bases: *object*

**Parameters**

- **source** ([FilesystemContent](#))
- **destination\_endpoint** ([Endpoint](#))
- **destination\_path** (*str* | *Path* | *None*)

## **annize.features.files.transfer.ssh module**

ssh-based file transfers.

```
class annize.features.files.transfer.ssh.Endpoint(*, host, port=22, username, identity_file,  
                                                  has_shell_access=False)
```

Bases: [Endpoint](#)

**Parameters**

- **host** (*str*)
- **port** (*int*)
- **username** (*str*)
- **identity\_file** (*str* | *None*)
- **has\_shell\_access** (*bool*)

```
property host: str
```

```
property port: int
```

```
property username: str
```

```
property identity_file: str | None
```

```
property has_shell_access: bool
```

```
access_filesystem(rootpath)
```

**locationstring**(*path=None*)

**Parameters**

**path** (*str* | *None*)

**Return type**

*str*

**exec**(*cmdline*)

**Parameters**

**cmdline** (*str*)

**Return type**

TODO

**\_abc\_impl** = <\_abc.\_abc\_data object>

## Submodules

### annize.features.files.common module

Files and directories.

**class** annize.features.files.common.**FsEntry**(\*, *path*, *root*)

Bases: *FilesystemContent*

A filesystem location, either relative to the Annize project root directory or another root.

If it is already known whether the entry is a file or a directory, consider using *File* or *Directory* instead. Special files (e.g. symlinks) can also be represented by a *File*.

**Parameters**

- **path** (*str* | *Path* | *None*) – The path that points to the referenced content (relative to *root*).
- **root** (*str* | *Path* | *FilesystemContent* | *None*) – The root directory. If unset, it is the Annize project root directory.

**property root:** *FilesystemContent*

The root directory.

*relative\_path* is considered to be relative to this one.

**property relative\_path:** *Path*

The path that points to the referenced content (relative to *root*).

**\_path**()

**class** annize.features.files.common.**File**(\*, *path*, *root*)

Bases: *FsEntry*

A file location, either relative to the Annize project root directory or another root.

**Parameters**

- **path** (*str* | *Path* | *None*) – The path that points to the referenced content (relative to *root*).
- **root** (*str* | *Path* | *FilesystemContent* | *None*) – The root directory. If unset, it is the Annize project root directory.

**class** annize.features.files.common.**Exclude**(\*, *by\_path\_pattern*, *by\_path*, *by\_name\_pattern*, *by\_name*)

Bases: object

An exclusion definition. Usually used with *Directory* and *DirectoryPart*.

**Parameters**

- **by\_path\_pattern** (*str* / *None*) – Exclude by this regexp pattern on the full path.
- **by\_path** (*str* / *None*) – Exclude this path.
- **by\_name\_pattern** (*str* / *None*) – Exclude by this regexp pattern on the file name.
- **by\_name** (*str* / *None*) – Exclude this file name.

**\_\_does\_exclude**(*by\_text*, *by\_pattern*)

**Parameters**

- **text** (*str*)
- **by\_text** (*str*)
- **by\_pattern** (*Pattern*)

**Return type**

bool

**does\_exclude**(*relative\_path*, *source*, *destination*)

Return whether a given location is excluded by this exclusion definition.

**Parameters**

- **relative\_path** (*Path*) – The relative path to check for exclusion.
- **source** (*Path*) – The absolute source path.
- **destination** (*Path*) – The absolute destination path.

**Return type**

bool

**class** annize.features.files.common.**ExcludeAllBut**(\*, *excludes*)

Bases: *Exclude*

A negative exclusion definition.

It will exclude an item whenever *\_none\_* of the given inner exclude definitions match.

**Parameters**

**excludes** (*list* [*Exclude*]) – List of inner exclude definitions.

**does\_exclude**(*relative\_path*, *source*, *destination*)

Return whether a given location is excluded by this exclusion definition.

**Parameters**

- **relative\_path** – The relative path to check for exclusion.
- **source** – The absolute source path.
- **destination** – The absolute destination path.

```
class annize.features.files.common.DirectoryPart(*, excludes, root, source_path=None,
                                                destination_path=None, path=None,
                                                destination_is_parent=False)
```

Bases: object

A part of a directory. Used in [Directory](#).

#### Parameters

- **excludes** (*Iterable*[[Exclude](#)]) – List of exclusion definitions.
- **root** (*str* | *Path* | [FilesystemContent](#) | *None*) – The root directory. If unset, it is the root directory specified for the owning [Directory](#).
- **source\_path** (*str* | *Path* | *None*) – The path that points to the referenced content (relative to root).
- **destination\_path** (*str* | *Path* | *None*) – The relative destination path inside the owning [Directory](#).
- **path** (*str* | *Path* | *None*) – Shorter way to set `source_path` and `destination_path` to the same path.
- **destination\_is\_parent** (*bool*) – Whether to consider the destination path as the parent of the new destination (instead of the new destination itself). The actual destination will have the same basename as the source then.

**property excludes:** [Sequence](#)[[Exclude](#)]

Exclusion definitions.

**property root:** [FilesystemContent](#) | *None*

The root directory (or *None* for the owning [Directory](#).root).

[source\\_path](#) is considered to be relative to this one.

**property source\_path:** [Path](#)

The path that points to the referenced content (relative to [root](#)).

**property destination\_path:** [Path](#)

The relative destination path inside the owning [Directory](#).

See also [destination\\_is\\_parent](#).

**property destination\_is\_parent:** *bool*

Whether to consider the destination path as the parent of the new destination (instead of the new destination itself).

```
class annize.features.files.common.Directory(*, path, root, excludes, parts, name)
```

Bases: [FsEntry](#)

A directory location, either relative to the Annize project root directory or another root.

Depending on how it is configured, this might point to a dynamically generated temporary location (e.g. if it is composed of parts or excludes are specified).

#### Parameters

- **path** (*str* | *None*) – The path that points to the referenced directory (relative to root). If set, do not set `parts`!
- **root** (*str* | *Path* | [FilesystemContent](#) | *None*) – The root directory. If unset, it is the Annize project root directory.

- **excludes** (*Iterable*[[Exclude](#)]) – Exclusion specifications. If some are specified, this directory will be dynamically generated. If set, do not set **parts**!
- **parts** (*Iterable*[[DirectoryPart](#)]) – Directory parts. If some are specified, this directory will be dynamically generated. Also, do not set **path** or **excludes**!
- **name** (*str* / *None*) – The name that this directory shall have (instead of its original name). If specified, this directory will be dynamically generated. It must not contain directory separator characters (mostly `"/"`).

**property parts:** [Sequence](#)[[DirectoryPart](#)]

**property excludes:** [Sequence](#)[[Exclude](#)]

**\_path()**

**\_\_transfer\_filter\_for\_exclude()**

**Parameters**

**exclude** ([Exclude](#))

**Return type**

`annize.fs.Path.TTransferFilter`

**class** `annize.features.files.common.ProjectDirectory`

Bases: [FilesystemContent](#)

The Annize project root directory.

**Parameters**

**generate\_func** – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).

**\_path()**

**class** `annize.features.files.common.MachineRootDirectory`

Bases: [FilesystemContent](#)

The machine root directory, i.e. `/`.

**Parameters**

**generate\_func** – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).

**\_path()**

[annize.features.homepage namespace](#)

[Subpackages](#)

[annize.features.homepage.sections namespace](#)

[Submodules](#)

[annize.features.homepage.sections.about module](#)

Homepage about section.

**class** `annize.features.homepage.sections.about.Section`(*\**, *head*=<*annize.i18n.ProvidedTrStr object*>, *sort\_index*=10000)

Bases: [HomepageSection](#)



```
generate_content(info)

_abc_impl = <_abc._abc_data object>
```

### annize.features.homepage.sections.changelog module

Homepage changelog section.

```
class annize.features.homepage.sections.changelog.Section(*, changelog,
                                                         head=<annize.i18n.ProvidedTrStr
                                                         object>, sort_index=60000)
```

Bases: *HomepageSection*

#### Parameters

**changelog** (*Changelog* / *None*)

```
generate_content(info)
```

```
_abc_impl = <_abc._abc_data object>
```

### annize.features.homepage.sections.documentation module

Homepage documentation section.

```
class annize.features.homepage.sections.documentation.Section(*, documentation,
                                                             head=<annize.i18n.ProvidedTrStr
                                                             object>, sort_index=30000)
```

Bases: *HomepageSection*

#### Parameters

**documentation** (*list* [*Document*])

```
pre_process_generate(info)
```

```
generate_content(info)
```

```
_abc_impl = <_abc._abc_data object>
```

### annize.features.homepage.sections.download module

Homepage download section.

```
class annize.features.homepage.sections.download.Section(*, distributables, dependencies,
                                                         head=<annize.i18n.ProvidedTrStr
                                                         object>, sort_index=40000)
```

Bases: *HomepageSection*

#### Parameters

- **distributables** (*list* [*Group*])
- **dependencies** (*list* [*Dependency*])

```
__generate_packagelist(info)
```

#### Parameters

**info** (*\_GenerateInfo*)

```
pre_process_generate(info)
```

```
post_process_generate(info)
```

```
generate_content(info)
```

```
_abc_impl = <_abc._abc_data object>
```

```
annize.features.homepage.sections.download.friendly_filesize(isize)
```

**Parameters**

**isize** (*int*)

**Return type**

str

```
annize.features.homepage.sections.download.filehash(filepath)
```

**Parameters**

**filepath** (*str*)

**Return type**

str

## annize.features.homepage.sections.gallery module

Homepage media gallery section.

```
class annize.features.homepage.sections.gallery.Section(*, head=<annize.i18n.ProvidedTrStr
                                                         object>, sort_index=50000,
                                                         media_galleries)
```

Bases: [HomepageSection](#)

**Parameters**

**media\_galleries** (*list[Gallery]*)

```
pre_process_generate(info)
```

```
generate_content(info)
```

```
_abc_impl = <_abc._abc_data object>
```

## annize.features.homepage.sections.imprint module

Homepage imprint section.

```
class annize.features.homepage.sections.imprint.Section(*, imprint,
                                                         head=<annize.i18n.ProvidedTrStr object>,
                                                         sort_index=70000)
```

Bases: [HomepageSection](#)

**Parameters**

**imprint** (*TrStr*)

```
generate_content(info)
```

```
_abc_impl = <_abc._abc_data object>
```

## annize.features.homepage.sections.license module

Homepage license section.

```
class annize.features.homepage.sections.license.Section(*, head=<annize.i18n.ProvidedTrStr
object>, sort_index=20000)
```

Bases: [HomepageSection](#)

`generate_content` (*info*)

`_abc_impl` = `<_abc._abc_data object>`

## Submodules

### annize.features.homepage.common module

Homepage.

```
class annize.features.homepage.common.HomepageSection(*, head, sort_index=0)
```

Bases: `ABC`

#### Parameters

- `head` ([TrStr](#))
- `sort_index` (*int*)

```
class _GenerateInfo(culture: annize.i18n.Culture, custom_arg: object | None, document_root_url: str,
document_variant_directory: annize.fs.Path, document_variant_url: str)
```

Bases: `object`

#### Parameters

- `culture` ([Culture](#))
- `custom_arg` (*object | None*)
- `document_root_url` (*str*)
- `document_variant_directory` ([Path](#))
- `document_variant_url` (*str*)

`culture`: [Culture](#)

`custom_arg`: *object | None*

`document_root_url`: *str*

`document_variant_directory`: [Path](#)

`document_variant_url`: *str*

```
class _PrePostProcGenerateInfo(document_root_directory: annize.fs.Path, document_root_url: str,
custom_arg: object | None)
```

Bases: `object`

#### Parameters

- `document_root_directory` ([Path](#))
- `document_root_url` (*str*)

- `custom_arg` (*object* | *None*)

`document_root_directory:` *Path*

`document_root_url:` *str*

`custom_arg:` *object* | *None*

**class** `Content`(\*, *rst\_text=""*, *media\_files=()*)

Bases: *object*

**Parameters**

- `rst_text` (*str*)
- `media_files` (*list*[*FilesystemContent*])

**property** `rst_text:` *str*

**property** `media_files:` *list*[*FilesystemContent*]

**append\_rst**(*rst\_text*)

**Return type**  
*None*

**attach\_media\_file**(*file*)

**Parameters**  
*file* (*FilesystemContent*)

**Return type**  
*None*

**property** `head:` *TrStr*

**property** `sort_index:` *int*

**pre\_process\_generate**(*info*)

**Parameters**  
*info* (*\_PrePostProcGenerateInfo*)

**Return type**  
*None*

**abstractmethod** `generate_content`(*info*)

**Parameters**  
*info* (*\_GenerateInfo*)

**Return type**  
*Content*

**post\_process\_generate**(*info*)

**Parameters**  
*info* (*\_PrePostProcGenerateInfo*)

**Return type**  
*None*

`_abc_impl = <_abc._abc_data object>`

```
class annize.features.homepage.common.Homepage(*, title, short_desc, sections, cultures)
```

Bases: object

#### Parameters

- **title** ([TrStr](#) / *None*)
- **short\_desc** ([TrStr](#) / *None*)
- **sections** (*list* [[HomepageSection](#)])
- **cultures** (*list* [[Culture](#)])

property cultures: *list* [[Culture](#)]

property sections: *list* [[HomepageSection](#)]

property title: [TrStr](#)

property short\_desc: [TrStr](#)

**\_append\_section**(*section*)

#### Parameters

**section** ([HomepageSection](#))

#### Return type

*None*

**generate**()

**\_\_generate\_pre\_post\_proc**(\*, *is\_pre*, *document\_root\_directory*, *document\_root\_url*)

#### Parameters

- **custom\_args** (*dict*)
- **is\_pre** (*bool*)
- **document\_root\_directory** ([Path](#))
- **document\_root\_url** (*str*)

**\_\_generate\_section**(\*, *custom\_args*, *culture*, *document\_root\_directory*, *document\_root\_url*, *document\_variant\_directory*)

#### Parameters

- **custom\_args** (*dict*)
- **culture** ([Culture](#))
- **document\_root\_directory** ([Path](#))
- **document\_root\_url** (*str*)
- **document\_variant\_directory** ([Path](#))

```
class annize.features.homepage.common.SimpleProjectHomepage(*, changelog, dependencies,
                                                             distributables, documentation,
                                                             imprint, media_galleries, **kwargs)
```

Bases: [Homepage](#)

#### Parameters

- **changelog** ([Changelog](#) / *None*)

- **dependencies** (*list*[[Dependency](#)])
- **distributables** (*list*[[Group](#)])
- **documentation** (*list*[[Document](#)])
- **imprint** ([TrStr](#) | *None*)
- **media\_galleries** (*list*[[Gallery](#)])

**class** `annize.features.homepage.common.GeneratedHomepage`(\*, *homepage*)

Bases: [FilesystemContent](#)

#### Parameters

- **generate\_func** – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).
- **homepage** ([Homepage](#))

`_path()`

## `annize.features.i18n` namespace

### Submodules

#### `annize.features.i18n.common` module

Internationalization, i.e. translation and similar tasks.

**class** `annize.features.i18n.common.ProjectDefinedTranslationProvider`

Bases: [TranslationProvider](#)

`__translations_for_stringname`(*stringname*)

#### Parameters

**stringname** (*str*)

#### Return type

`dict[str, str]`

`translate`(*stringname*, \*, *culture*)

`add_translations`(*stringname*, *variants*)

#### Parameters

- **stringname** (*str*)
- **variants** (*dict*[*str*, *str*])

#### Return type

*None*

`_abc_impl` = `<_abc._abc_data object>`

**class** `annize.features.i18n.common.String`(\*, *stringname*, *stringtr*, *\*\*variants*)

Bases: [ProvidedTrStr](#)

#### Parameters

- **stringname** (*str* | *None*)
- **stringtr** (*str* | *None*)

```

        • variants (str)
    _abc_impl = <_abc._abc_data object>

class annize.features.i18n.common.Culture(*, iso_639_1_lang_code, subcode, fallback_cultures)
    Bases: Culture

    Parameters
        • iso_639_1_lang_code (str)
        • subcode (str | None)
        • fallback_cultures (list[Culture])

class annize.features.i18n.common.UnspecifiedCulture
    Bases: UnspecifiedCulture

class annize.features.i18n.common.IdCulture
    Bases: IdCulture

class annize.features.i18n.common.ProjectCultures(*, cultures)
    Bases: list

    Parameters
        cultures (list[Culture])

annize.features.i18n.common.project_cultures()

    Return type
        list[Culture]

```

## annize.features.i18n.gettext module

gettext-based internationalization.

```

class annize.features.i18n.gettext.UpdatePOs(*, po_directory)
    Bases: object

    Parameters
        po_directory (str | Path)

class annize.features.i18n.gettext.GenerateMOs(*, po_directory, mo_directory)
    Bases: object

    Parameters
        • po_directory (str | Path | FileSystemContent)
        • mo_directory (str | Path)

```

## annize.features.injections namespace

### Submodules

#### annize.features.injections.common module

Injects.

```

class annize.features.injections.common.Injection
    Bases: ABC

```

```
abstractmethod inject(destination)
```

Parameters

**destination** ([Path](#))

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.features.injections.common.FilesystemContentInjection(*, content)
```

Bases: [Injection](#)

Parameters

**content** ([str](#) | [Path](#) | [FilesystemContent](#))

property content: [FilesystemContent](#)

```
inject(destination)
```

Parameters

**destination** ([Path](#))

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.features.injections.common.Inject(*, injection, destination)
```

Bases: [object](#)

Parameters

- **injection** ([Injection](#))
- **destination** ([Path](#))

## [annize.features.injections.python module](#)

Python injections.

```
class annize.features.injections.python.ProjectInfoInjection(*, filename='project_info.py',  
                                                             version)
```

Bases: [Injection](#)

Parameters

- **filename** ([str](#))
- **version** ([Version](#) | [None](#))

```
inject(destination)
```

Parameters

**destination** ([Path](#))

```
_abc_impl = <_abc._abc_data object>
```

## [annize.features.testing namespace](#)

### [Submodules](#)

### [annize.features.testing.common module](#)

Testing.



```

class annize.features.testing.common.RunTests(**b)
    Bases: object

class annize.features.testing.common.Test
    Bases: ABC
    abstractmethod run()
    _abc_impl = <_abc._abc_data object>

class annize.features.testing.common.TestGroup(*, tests)
    Bases: Test
    Parameters
        tests (list[Test])
    run()
    _abc_impl = <_abc._abc_data object>

```

### annize.features.testing.pylint module

pylint testing.

```

class annize.features.testing.pylint.Test(*, source)
    Bases: Test
    Parameters
        source (FilesystemContent)
    run()
    _abc_impl = <_abc._abc_data object>

```

### annize.features.testing.pytest module

pytest testing.

```

class annize.features.testing.pytest.Test(*, test_directory, source_directory)
    Bases: Test
    Parameters
        • test_directory (str | Path)
        • source_directory (str | Path)
    run()
    _abc_impl = <_abc._abc_data object>

```

### annize.features.testing.pyunit module

pyunit testing.

```

class annize.features.testing.pyunit.Test(*, src)
    Bases: Test
    Parameters
        src (str)

```

```
run()
```

```
_abc_impl = <_abc._abc_data object>
```

`annize.features.version_control` namespace

Submodules

`annize.features.version_control.common` module

Version control.

```
class annize.features.version_control.common.VersionControlSystem
```

Bases: `ABC`

```
abstractmethod get_current_revision()
```

Return type

`str`

```
abstractmethod get_revision_list()
```

Return type

`list[str]`

```
abstractmethod get_commit_message(revision)
```

Parameters

**revision** (`str`)

Return type

`str`

```
abstractmethod get_commit_time(revision)
```

Parameters

**revision** (`str`)

Return type

`datetime`

```
abstractmethod get_revision_number(revision)
```

Parameters

**revision** (`str`)

Return type

`int`

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.features.version_control.common.BuildVersion(*, base_version, vcs)
```

Bases: `Version`

Parameters

- **base\_version** (`Version`)
- **vcs** (`VersionControlSystem`)

```
__effversion()
```

**property** `segments_tuples`

**property** `text`

`annize.features.version_control.common.default_version_control_system()`

**Return type**

`VersionControlSystem` | `None`

## **annize.features.version\_control.git module**

git-based version control.

**class** `annize.features.version_control.git.VersionControlSystem(*, path)`

Bases: `VersionControlSystem`

**Parameters**

`path` (`str` | `None`)

**property** `path`: `Path`

`__call_git(cmdline)`

**Parameters**

`cmdline` (`list[str]`)

**Return type**

`str`

`get_current_revision()`

`get_revision_list()`

`get_commit_message(revision)`

`get_commit_time(revision)`

`get_revision_number(revision)`

`_abc_impl = <_abc._abc_data object>`

**class** `annize.features.version_control.git.ExcludeByGitIgnores`

Bases: `Exclude`

**Parameters**

- **by\_path\_pattern** – Exclude by this regexp pattern on the full path.
- **by\_path** – Exclude this path.
- **by\_name\_pattern** – Exclude by this regexp pattern on the file name.
- **by\_name** – Exclude this file name.

`does_exclude(relative_path, source, destination)`

Return whether a given location is excluded by this exclusion definition.

**Parameters**

- **relative\_path** – The relative path to check for exclusion.
- **source** – The absolute source path.
- **destination** – The absolute destination path.

## Submodules

### annize.features.authors module

Project author information.

```
class annize.features.authors.Author(*, fullname, email)
```

Bases: object

#### Parameters

- **fullname** (*TrStr*)
- **email** (*str* | *None*)

**property** **fullname**: *TrStr*

**property** **email**: *str* | *None*

```
annize.features.authors.project_authors()
```

#### Return type

*list*[*Author*]

```
annize.features.authors.join_authors(authors)
```

#### Parameters

**authors** (*list*[*Author*])

#### Return type

*Author*

### annize.features.base module

Project base information.

```
class annize.features.base.Data(*, project_name=None, pretty_project_name=None, summary=None,  
                                long_description=None, homepage_url=None, imprint=None,  
                                project_directory=None)
```

Bases: object

#### Parameters

- **project\_name** (*str*)
- **pretty\_project\_name** (*TrStr* | *None*)
- **summary** (*TrStr* | *None*)
- **long\_description** (*TrStr* | *None*)
- **homepage\_url** (*str* | *None*)
- **imprint** (*TrStr* | *None*)
- **project\_directory** (*str* | *None*)

**property** **project\_name**: *str*

**property** **pretty\_project\_name**: *TrStr*

**property** **summary**: *TrStr*

```

    property long_description: TrStr

    property homepage_url: str

    property imprint: TrStr

    property project_directory: str

class annize.features.base.BrandColor(*, red, green, blue)
    Bases: Color

        Parameters
            • red (float)
            • green (float)
            • blue (float)

class annize.features.base.DateTime(*, iso)
    Bases: datetime

        Parameters
            iso (str)

class annize.features.base.Keywords(*, from_string='', split_by=' ', keywords=())
    Bases: object

        Parameters
            • from_string (str)
            • split_by (str)
            • keywords (list[str])

    property keywords: list[str]

class annize.features.base.Keyword(text)
    Bases: Keywords

        Parameters
            text (str)

annize.features.base.project_keywords()

    Return type
        Keywords

class annize.features.base.Basket(*, items)
    Bases: Basket

        Parameters
            items (list[object])

class annize.features.base.List(*, items)
    Bases: list

        Parameters
            items (list[object])

```

**class** annize.features.base.FirstOf(\*, objects)

Bases: [Basket](#)

**Parameters**

**objects** (*list[object]*)

annize.features.base.brand\_color(\*, none\_on\_undefined=False)

**Parameters**

**none\_on\_undefined** (*bool*)

**Return type**

[Color](#)

annize.features.base.\_get\_data(key, default)

**Parameters**

- **key** (*str*)
- **default** (*Any*)

**Return type**

*Any*

annize.features.base.project\_name()

**Return type**

*str*

annize.features.base.pretty\_project\_name()

**Return type**

[TrStr](#)

annize.features.base.summary()

**Return type**

[TrStr](#)

annize.features.base.long\_description()

**Return type**

[TrStr](#)

annize.features.base.homepage\_url()

**Return type**

*str*

annize.features.base.imprint()

**Return type**

[TrStr](#)

annize.features.base.project\_directory()

**Return type**

[Path](#)

**annize.features.licensing module**

Project licensing information.

**class** annize.features.licensing.**License**(\*, *name*, *text*, *\*\*additional\_info*)

Bases: object

**Parameters**

- **name** (*TrStr*)
- **text** (*TrStr* / *None*)

**property name:** *TrStr*

**property text:** *TrStr* | *None*

**additional\_info**(*key*, \*, *default=None*)

**Parameters**

- **key** (*str*)
- **default** (*Any*)

**Return type**

*Any*

annize.features.licensing.**\_license**(*name*)

**Parameters**

**name** (*str*)

**Return type**

*Type*[*License*]

annize.features.licensing.**AFLv3**

alias of *ALicense*

annize.features.licensing.**AGPLv3**

alias of *ALicense*

annize.features.licensing.**Apache2**

alias of *ALicense*

annize.features.licensing.**Artistic1**

alias of *ALicense*

annize.features.licensing.**BSD2clause**

alias of *ALicense*

annize.features.licensing.**BSD3clause**

alias of *ALicense*

annize.features.licensing.**Cc0v1**

alias of *ALicense*

annize.features.licensing.**CcBy3**

alias of *ALicense*

annize.features.licensing.**CcByNc3**

alias of *ALicense*

`annize.features.licensing.CcByNcNd3`  
alias of `ALicense`

`annize.features.licensing.CcByNcSa3`  
alias of `ALicense`

`annize.features.licensing.CcByNd3`  
alias of `ALicense`

`annize.features.licensing.CcBySa3`  
alias of `ALicense`

`annize.features.licensing.GPLv3`  
alias of `ALicense`

`annize.features.licensing.LGPLv3`  
alias of `ALicense`

`annize.features.licensing.MIT`  
alias of `ALicense`

`annize.features.licensing.MPLv11`  
alias of `ALicense`

`annize.features.licensing.MPLv2`  
alias of `ALicense`

`annize.features.licensing.PublicDomain`  
alias of `ALicense`

`annize.features.licensing.project_licenses()`

**Return type**list[[License](#)]**`annize.features.media_galleries` module**

Media galleries.

**class** `annize.features.media_galleries.MediaType(*values)`Bases: `Enum`**IMAGE** = `'image'`**VIDEO** = `'video'`**class** `annize.features.media_galleries.Gallery(*, source, title)`Bases: `object`**Parameters**

- **source** ([FilesystemContent](#))
- **title** ([TrStr](#) | `None`)

**class** `Item(file, description, mediatype)`Bases: `object`**Parameters**

- **file** ([FilesystemContent](#))



- **description** (`TrStr` | `None`)
- **mediatype** (`MediaType`)

property file: `FilesystemContent`

property description: `TrStr` | `None`

property mediatype: `MediaType`

property items: `list[Item]`

property title: `TrStr`

`_description_for_mediafile(itemfile)`

**Parameters**

`itemfile` (`FilesystemContent`)

**Return type**

`TrStr`

### annize.features.task module

Tasks.

**class** `annize.features.task.Task`(\*, `innertasks`, `is_advanced=False`)

Bases: `object`

**Parameters**

- **innertasks** (`list[object]`)
- **is\_advanced** (`bool`)

property `is_advanced`: `bool`

### annize.features.version module

Project versioning.

**class** `annize.features.version.Line`(\*, `version`)

Bases: `object`

**Parameters**

`version` (`Version`)

property `version`: `Version`

**class** `annize.features.version.Version`(\*, `text`, `pattern`, `**segment_values`)

Bases: `Version`

**Parameters**

- **text** (`str` | `None`)
- **pattern** (`VersionPattern` | `None`)

`annize.features.version.default_version_pattern()`

**Return type**

`VersionPattern`

`annize.features.version.project_versions()`

**Return type**

`list[Version]`

**class** `annize.features.version.CommonVersionPattern`

Bases: `VersionPattern`

## **annize.flow package**

Execution of Annize projects.

See e.g. `annize.flow.runner.Runner` and `annize.flow.run_context.RunContext`.

## **Submodules**

### **annize.flow.run\_context module**

Run contexts. Typically used by the infrastructure in the course of the execution of an Annize project.

See also `RunContext` and `current()`.

**class** `annize.flow.run_context.RunContext`

Bases: `object`

Holds data for a single execution of an Annize project (usually happening in a `annize.flow.runner.Runner`).

Beyond a few fixed data, like the root path of the Annize project configuration files, it stores every object that was created by definition in the Annize project configuration. Many parts of this API (e.g. many method names) use the term ‘object’ for all data items stored in a run context.

Each of those objects has at least one name. This can be a “friendly name”, i.e. a name that was explicitly specified in the project. If no name was specified, there will at least be an automatically generated one, so every object is uniquely addressable by name.

There are further ways to access stored data, which do not involve names. See this class’ methods.

For each object in the store, arbitrary metadata can be stored as well.

See also `current()`.

The owner of a run context (i.e. parts of Annize infrastructure!) needs to enter the context (`with-block`) during execution. TODO xx why?

Do not use directly.

`_IS_TOPLEVEL_OBJECT__METADATA_KEY = 'annize..is_toplevel_object'`

`_ANNIZE_CONFIG_ROOTPATH__NAME = 'annize..annize_config_rootpath'`

**prepare**(`*`, `annize_config_rootpath`)

Prepare the execution.

Needs to be called once, before the actual execution begins, in order to make some basic data available.

**Parameters**

**annize\_config\_rootpath** (*Path*) – The Annize project configuration root path.

**Return type**

`None`

**object\_by\_name**(*name*, *default=None*, \*, *create\_nonexistent=False*)

Return an object by one of its names (or a default value).

See also [set\\_object\\_name\(\)](#).

#### Parameters

- **name** (*str*) – An object name.
- **default** (*Any*) – The default value to return when no object exists with the given name.
- **create\_nonexistent** (*bool*) – (If the default value is going to be returned because no object existed with the given name) Whether to store the default value in the data store with the given name, so it can be found later.

#### Return type

*Any*

**object\_names**(*obj*)

Return all object names for a given object (with friendly names first).

This method always returns a non-empty list. Even if the given object was not stored at all yet, it automatically gets added to the store implicitly.

See also [set\\_object\\_name\(\)](#).

#### Parameters

**obj** (*Any*) – The object.

#### Return type

list[str]

**object\_name**(*obj*)

Return one object name for a given object (preferably a friendly one).

This method always returns a valid name. Even if the given object was not stored at all yet, it automatically gets added to the store implicitly.

See also [set\\_object\\_name\(\)](#).

#### Parameters

**obj** (*Any*) – The object.

#### Return type

str

**set\_object\_name**(*obj*, *name*)

Assign a name to an object.

All names assigned earlier remain valid as well.

See also [object\\_by\\_name\(\)](#), [object\\_names\(\)](#) and others.

#### Parameters

- **obj** (*Any*) – The object.
- **name** (*str*) – The new name.

#### Return type

None

**objects\_by\_type**(*obj\_type*, *toplevel\_only*=True)

Return all stored objects that are instance of a given type.

See also [add\\_object\(\)](#) and others.

**Parameters**

- **obj\_type** (*type[T]*) – The type.
- **toplevel\_only** (*bool*) – Whether to return only objects that are defined on project level.

**Return type**

list[T]

**add\_object**(*obj*)

Add an object to the store and return its name.

If the object already is the store, and already has a friendly name, this one is returned. So, in fact, this method has the same effect as [object\\_name\(\)](#). It might just express your intent better than that one in some cases.

See also [object\\_by\\_name\(\)](#), [objects\\_by\\_type\(\)](#) and others.

**Parameters**

**obj** (*Any*) – The object to add.

**Return type**

str

**is\_friendly\_name**(*name*)

Returns whether the given name is a friendly one.

**Parameters**

**name** (*str*) – The name to check.

**Return type**

bool

**is\_toplevel\_object**(*obj*)

Return whether a given object represents the definition of an object on the Annize project file root level (e.g. by the top level tags in .xml configuration files).

**Parameters**

**obj** (*Any*) – The object to check.

**Return type**

bool

**mark\_object\_as\_toplevel**(*obj*)

Mark an object as a toplevel one. See [is\\_toplevel\\_object\(\)](#).

**Parameters**

**obj** (*Any*) – The object.

**Return type**

None

**object\_metadata**(*obj*, *key*, *default*=None)

Return a piece of metadata for a given object (or a default value if there is no value by the given key).

See also [set\\_object\\_metadata\(\)](#).

**Parameters**

- **obj** (*Any*) – The object.
- **key** (*str*) – The metadata key.
- **default** (*Any*) – The default value.

**Return type***Any***set\_object\_metadata**(*obj*, *key*, *value=None*)

Store a piece of metadata for a given object.

See also [\*object\\_metadata\(\)\*](#).**Parameters**

- **obj** (*Any*) – The object.
- **key** (*str*) – The metadata key.
- **value** (*Any*) – The metadata value to store for this object and key.

**Return type**

None

**\_\_put\_object**(*name*, *obj*)**Parameters**

- **name** (*str*)
- **obj** (*Any*)

**Return type**

None

**\_\_object\_raw\_name**(*obj*)**Parameters****obj** (*Any*)**Return type**

str

**\_\_object\_metadata\_dict**(*obj*)**Parameters****obj** (*Any*)**Return type**dict[str, *Any*]**annize.flow.run\_context.current()**

Return the current run context.

Note: In most cases you do not need to use this function directly. See the other functions defined on module level.

If there is no current run context (i.e. this function is called outside the execution of an Annize project), [\*OutOfContextError\*](#) will be raised.

**Return type**[\*RunContext\*](#)

**exception** `annize.flow.run_context.OutOfContextError`

Bases: `TypeError`

`annize.flow.run_context.object_by_name(name, default=None, *, create_nonexistent=False)`

Same as `RunContext.object_by_name()` on the `_current_` run context (`current()`).

**Parameters**

- **name** (*str*)
- **default** (*Any*)
- **create\_nonexistent** (*bool*)

**Return type**

*Any*

`annize.flow.run_context.object_names(obj)`

Same as `RunContext.object_names()` on the `_current_` run context (`current()`).

**Parameters**

**obj** (*Any*)

**Return type**

`list[str]`

`annize.flow.run_context.object_name(obj)`

Same as `RunContext.object_name()` on the `_current_` run context (`current()`).

**Parameters**

**obj** (*Any*)

**Return type**

`str`

`annize.flow.run_context.set_object_name(obj, name)`

Same as `RunContext.set_object_name()` on the `_current_` run context (`current()`).

**Parameters**

- **obj** (*Any*)
- **name** (*str*)

**Return type**

`None`

`annize.flow.run_context.objects_by_type(obj_type, toplevel_only=True)`

Same as `RunContext.objects_by_type()` on the `_current_` run context (`current()`).

**Parameters**

- **obj\_type** (*type[T]*)
- **toplevel\_only** (*bool*)

**Return type**

`list[T]`

`annize.flow.run_context.add_object(obj)`

Same as `RunContext.add_object()` on the `_current_` run context (`current()`).

**Parameters**

**obj** (*Any*)

**Return type**

str

`annize.flow.run_context.is_friendly_name(name)`Same as `RunContext.is_friendly_name()` on the `_current_` run context (`current()`).**Parameters****name** (str)**Return type**

bool

`annize.flow.run_context.is_toplevel_object(obj)`Same as `RunContext.is_toplevel_object()` on the `_current_` run context (`current()`).**Parameters****obj** (Any)**Return type**

bool

`annize.flow.run_context.object_metadata(obj, key, default=None)`Same as `RunContext.object_metadata()` on the `_current_` run context (`current()`).**Parameters**

- **obj** (Any)
- **key** (str)
- **default** (Any)

**Return type**

Any

`annize.flow.run_context.set_object_metadata(obj, key, value=None)`Same as `RunContext.set_object_metadata()` on the `_current_` run context (`current()`).**Parameters**

- **obj** (Any)
- **key** (str)
- **value** (Any)

**Return type**

None

**annize.flow.runner module**

The Annize project runner.

See [Runner](#).**class** `annize.flow.runner.Runner(*, project, selected_task=None, user_feedback=None)`

Bases: ABC

TODO.

**Parameters**

- **project** (Project)
- **selected\_task** (str | None)

- `user_feedback` (*TODO*)

`run_runner()`

**Return type**

None

`_dispatch(func)`

`__do_run(project)`

**Parameters**

**project** (*Project*)

`abstractmethod show_task_chooser()`

**Return type**

None

`abstractmethod show_task_execution()`

**Return type**

None

`abstractmethod show_task_execution_success()`

**Return type**

None

`get_tasks()`

**Return type**

list[str]

`__set_tasks(tasks)`

**Parameters**

**tasks** (*list[str]*)

**Return type**

None

`get_selected_task()`

**Return type**

str

`set_selected_task(task_name)`

**Parameters**

**task\_name** (*str*)

**Return type**

None

`is_finished()`

**Return type**

bool



**get\_success\_state()**

**Return type**

*Tuple*[bool, str]

**\_\_set\_success\_state**(*success*, *message*)

**Parameters**

- **success** (*bool*)
- **message** (*str*)

**Return type**

None

**wait\_finished()**

**Return type**

None

**\_abc\_impl** = <\_abc.\_abc\_data object>

## annize.fs package

Annize filesystem API.

Used by Annize Features.

See [Path](#), [FilesystemContent](#) and others.

**class** `annize.fs.FilesystemContent`(*generate\_func*)

Bases: object

Base class for a source of arbitrary filesystem content.

It provides access to that content by [path\(\)](#). Some implementations will return a static path to already existing content, while other implementations will return a temporary path to ad-hoc generated content.

This content can be a file, a complete directory, or anything else. It could even return a path that points to nothing. It depends on the actual implementation what kind of content it provides.

This type is used instead of plain paths in situations where dynamic filesystem content might be exchanged (usually via some temporary files) instead of already existing files or directories. So, a `FilesystemContent` usually provides a path for reading. There is no strict rule against writing at that path, but that might lead to expected behavior (e.g. when the path points to a temporary copy of something, so changes do not take the desired effect, or when it has undesired side effects on other consumers of the same instance).

**Parameters**

**generate\_func** (*Callable*[[], *TInputPath*]) – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).

**path()**

Return the path that points to the content.

It always returns the same path and does not do any further processing when called more than once (so it is safe and cheap to call that multiple times).

**Return type**

[Path](#)

**class** annize.fs.Path(\*args, \*\*kwargs)

Bases: Path, [FilesystemContent](#)

A path.

This is compatible to `pathlib.Path`, but provides some convenience methods that can save a few lines of code for typical operations.

Each path is also a [FilesystemContent](#). However, since `FilesystemContent` only allows absolute paths, using a relative path as a `FilesystemContent` will fail at runtime! See also [content\(\)](#).

**Parameters**

**args** (*str* / *Path* / *FilesystemContent*) – Path parts. Often this is one string, one `pathlib.Path` or one [FilesystemContent](#). The latter one is only allowed as the first part.

**\_path()**

Return itself (in order to implement [FilesystemContent](#)).

**Return type**

[Path](#)

**path()**

Return itself (in order to implement [FilesystemContent](#)).

**Return type**

[Path](#)

**children()**

Like `iterdir()`, but sorted by name.

**Return type**

*Sequence*[[Path](#)]

**ctime()**

Return the ctime for this path.

**Return type**

*datetime*

**mtime()**

Return the mtime for this path.

**Return type**

*datetime*

**write\_file(data)**

Write data to a file at this path (like `write_text` or `write_bytes`).

**Parameters**

**data** (*bytes* / *TrStr* / *str*) – The data to write.

**Return type**

None

**remove(\*, missing\_ok=True)**

Remove the file, directory, symlink, ... at this path.

**Parameters**

**missing\_ok** (*bool*) – Whether it is okay if there is nothing at this path.

**Return type**

None

**file\_size()**

Return the file size in bytes.

**Return type**

int

**temp\_clone(\*, temp\_root\_path=None, basename=None)**

Return a temporary clone of the content at this path.

**Parameters**

- **temp\_root\_path** (*str* / *Path* / *None*) – Optional root directory for temporary files. If unset, an OS-default will be used.
- **basename** (*str* / *None*) – Optional new basename. If unset, the original one will be used.

**Return type**

*Path*

**TTransferFilter**

alias of Callable[[*Path*, *Path*, *Path*], bool]

**copy\_to(destination, \*, destination\_as\_parent=False, merge=False, overwrite=False, transfer\_filter=None)**

Copy the file, directory, symlink, ... at this path to a given destination. All missing parent directories in the destination path get created automatically.

**Parameters**

- **destination** (*str* / *Path*) – The destination.
- **destination\_as\_parent** (*bool*) – Whether to consider the destination as the parent of the new destination (instead of the new destination itself). The actual destination will have the same basename as the source then.
- **merge** (*bool*) – Whether to merge the source content into the destination. If not, each new destination directory will replace the existing one or even fail.
- **overwrite** (*bool*) – Whether to allow overwriting of the destination.
- **transfer\_filter** (Callable[[*Path*, *Path*, *Path*], bool] / *None*) – The optional transfer filter to use. It can exclude particular parts from the transfer. It is a function with three *Path* parameters: The relative path of an item, the absolute source path and the absolute destination path. It returns *False* to skip that item.

**Return type**

*Path*

**move\_to(destination, \*, destination\_as\_parent=False, merge=False, overwrite=False, transfer\_filter=None)**

Move the file, directory, symlink, ... at this path to a given destination. All missing parent directories in the destination path get created automatically.

**Parameters**

- **destination** (*str* / *Path*) – The destination.
- **destination\_as\_parent** (*bool*) – Whether to consider the destination as the parent of the new destination (instead of the new destination itself). The actual destination will have the same basename as the source then.
- **merge** (*bool*) – Whether to merge the source content into the destination. If not, each new destination directory will replace the existing one or even fail.

- **overwrite** (*bool*) – Whether to allow overwriting of the destination.
- **transfer\_filter** (*Callable*[[*Path*, *Path*, *Path*], *bool*] | *None*) – The optional transfer filter to use. It can exclude particular parts from the transfer. It is a function with three *Path* parameters: The relative path of an item, the absolute source path and the absolute destination path. It returns *False* to skip that item.

**Return type**

*Path*

**class TransferFilters**

Bases: *object*

**class And**(\**inner\_filters*)

Bases: *object*

**Parameters**

**inner\_filters** (*Path.TTransferFilter*)

**class \_TransferHelper**

Bases: *object*

**static transfer\_to**(*source*, *destination*, \*, *merge*, *overwrite*, *destination\_as\_parent*, *action*, *transfer\_filter*=*None*)

**Parameters**

- **source** (*Path*)
- **destination** (*Path*)
- **merge** (*bool*)
- **overwrite** (*bool*)
- **destination\_as\_parent** (*bool*)
- **action** (*Callable*)
- **transfer\_filter** (*Callable*[[*Path*, *Path*, *Path*], *bool*] | *None*)

**Return type**

*Path*

**static transfer\_action\_copy**(*source*, *destination*)

**Parameters**

- **source** (*Path*)
- **destination** (*Path*)

**Return type**

*None*

**static transfer\_action\_move**(*source*, *destination*)

**Parameters**

- **source** (*Path*)
- **destination** (*Path*)

**Return type**

*None*

**static \_TransferHelper\_\_transfer\_piece**(*action*, *transfer\_filter*, *source*, *destination*, *merge*, *overwrite*, *relative\_path*=*"*)

**Parameters**

- **action** (*Callable*)
- **transfer\_filter** (*Callable*[[*Path*, *Path*, *Path*], *bool*] | *None*)
- **source** (*Path*)
- **destination** (*Path*)
- **merge** (*bool*)
- **overwrite** (*bool*)

- **relative\_path** (*str*)

**Return type**

None

`annize.fs.content(f, *, root=None)`

Return a [FilesystemContent](#) for an arbitrary given path or [FilesystemContent](#).

If the input already is a valid [FilesystemContent](#), it gets returned as-is. If the input is a string, it automatically gets interpreted as a path (like [Path](#)). If it is a relative path, this function will return a [FilesystemContent](#) that interprets it relative to the current Annize project root directory (which only makes sense when used inside an Annize project execution) or another root location.

**Parameters**

- **f** (*str* | *Path* | [FilesystemContent](#)) – The input path or filesystem content.
- **root** (*str* | *Path* | [FilesystemContent](#) | *None*) – The path or filesystem content to be used as root directory for relative paths in *f*.

**Return type**[FilesystemContent](#)

`annize.fs.fresh_temp_directory(name=None, *, temp_root_path=None)`

Return a fresh empty temporary directory for arbitrary usage.

This directory will automatically be removed after the Annize project run has been finished. It can only be used for a `with`-block, which removes it directly after this block. Each instance can only be used once in the latter way.

For usage without a `with`-block, see [annize.fs.ext.FreshTempDirectory.path](#).

**Parameters**

- **name** (*str* | *Path* | *None*) – The optional directory name. Otherwise, the implementation will choose a name.
- **temp\_root\_path** (*str* | *Path* | *None*) – Optional root directory for temporary files. If unset, an OS-default will be used.

**Return type**[FreshTempDirectory](#)

`annize.fs.dynamic_file(*, content, file_name=None, temp_root_path=None)`

Return a ‘filesystem content’ that provides a file with some given content.

**Parameters**

- **content** (*str* | *bytes* | *Callable[[], str | bytes]*) – The content of this dynamic file. This may be either direct content (*str* or *bytes*) or a function that returns content.
- **file\_name** (*str* | *None*) – The optional file name. Otherwise, the implementation will choose a name.
- **temp\_root\_path** (*str* | *Path* | *None*) – Optional root directory for temporary files. If unset, an OS-default will be used.

**Return type**[FilesystemContent](#)

## Submodules

### annize.fs.ext module

Annize filesystem API extensions.

Note: Commonly used functionality is also available in simpler ways (e.g. somehow in [annize.fs](#)).

**class** `annize.fs.ext.FreshTempDirectory`(*name=None, \*, temp\_root\_path=None*)

Bases: `object`

A fresh empty temp directory for arbitrary usage.

See [annize.fs.fresh\\_temp\\_directory\(\)](#).

Do not use directly.

#### Parameters

- **name** (*str* | *Path* | *None*)
- **temp\_root\_path** (*str* | *Path* | *None*)

**property** `path`: [Path](#)

The path of this temp directory.

It is empty after creation and will be removed automatically after usage.

`__cleanup()`

**class** `annize.fs.ext.DynamicFile`(*\*, content, file\_name=None, temp\_root\_path=None*)

Bases: [FilesystemContent](#)

A filesystem content that provides a file with some given content.

See [annize.fs.dynamic\\_file\(\)](#).

Do not use directly.

#### Parameters

- **content** (*str* | *bytes* | *Callable*[[*str* | *bytes*]], *str* | *bytes*)
- **file\_name** (*str* | *None*)
- **temp\_root\_path** (*str* | *Path* | *None*)

`_TStaticContent` = *str* | *bytes*

`_TContent`

alias of *str* | *bytes* | *Callable*[[*str* | *bytes*], *str* | *bytes*]

`_path()`

**class** `annize.fs.ext.Mount`(*src, dst, \*, options=(), mount\_command=('mount',),  
umount\_command=('umount',))*)

Bases: `object`

Mounting of a filesystem.

This mounts a filesystem as long as its context is entered (with-block).

#### Parameters

- **src** (*str* | *Path*) – The filesystem to mount. Often a device file.

- **dst** (*str* | *Path*) – The mount-point.
- **options** (*Iterable[str]*) – Additional mount options.
- **mount\_command** (*Sequence[str]*) – The mount command to use.
- **umount\_command** (*Sequence[str]*) – The umount command to use.

**property destination:** *Path*

The mount-point.

## annize.i18n package

TODO.

See *tr()*.

**class** *annize.i18n.TranslationProvider*

Bases: *ABC*

**abstractmethod** *translate(stringname, \*, culture)*

### Parameters

- **stringname** (*str*)
- **culture** (*Culture*)

### Return type

*str* | *None*

*\_abc\_impl* = <*\_abc.\_abc\_data* object>

**class** *annize.i18n.GettextTranslationProvider(mopath)*

Bases: *TranslationProvider*

**translate**(*stringname, \*, culture*)

### Parameters

- **stringname** (*str*)
- **culture** (*Culture*)

*\_abc\_impl* = <*\_abc.\_abc\_data* object>

*annize.i18n.add\_translation\_provider(provider, \*, priority=0)*

### Parameters

- **provider** (*TranslationProvider*)
- **priority** (*int*)

### Return type

*None*

*annize.i18n.translation\_providers()*

### Return type

*list[TranslationProvider]*

```
annize.i18n.tr(stringname, *, culture=None)
```

**Parameters**

- **stringname** (*str*)
- **culture** (*Culture* | *str* | *None*)

**Return type**

*str*

```
class annize.i18n.TrStr
```

Bases: *ABC*

```
static tr(stringname)
```

**Parameters**

**stringname** (*str*)

**Return type**

*TrStr*

```
translate(culture=None)
```

**Parameters**

**culture** (*Culture* | *str* | *None*)

**Return type**

*str*

```
abstractmethod get_variant(culture)
```

**Parameters**

**culture** (*Culture*)

**Return type**

*str*

```
property stringname: str
```

```
format(*args, **kwargs)
```

**Return type**

*TrStr*

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.i18n.ProvidedTrStr(stringname)
```

Bases: *TrStr*

**Parameters**

**stringname** (*str*)

```
property stringname
```

```
get_variant(culture)
```

```
_abc_impl = <_abc._abc_data object>
```

```
annize.i18n.TrStrOrStr = annize.i18n.TrStr | str
```

Type annotation for something that can be either a *str* or a *TrStr*.



```
annize.i18n.tr_if_trstr(txt, culture=None)
```

**Parameters**

- **txt** ([TrStr](#) | *str*)
- **culture** ([Culture](#) | *str* | *None*)

**Return type**

*str*

```
annize.i18n.to_trstr(txt)
```

**Parameters**

**txt** ([TrStr](#) | *str*)

**Return type**

[TrStr](#)

```
class annize.i18n.Culture(english_lang_name, iso_639_1_lang_code, subcode, fallback_cultures)
```

Bases: *object*

**Parameters**

- **english\_lang\_name** (*str*)
- **iso\_639\_1\_lang\_code** (*str*)
- **subcode** (*str* | *None*)
- **fallback\_cultures** (*Iterable*[[Culture](#)])

```
static get_from_iso_639_1_lang_code(iso_639_1_lang_code, subcode=None, fallback_cultures=())
```

**Parameters**

- **iso\_639\_1\_lang\_code** (*str*)
- **subcode** (*str* | *None*)
- **fallback\_cultures** (*Iterable*[[Culture](#)])

```
property english_lang_name: str
```

```
property iso_639_1_lang_code: str
```

```
property subcode: str
```

```
property fullname: str
```

```
property fallback_cultures: Iterable[Culture]
```

```
__find_lcall()
```

```
__get_env()
```

```
__set_env(language, locale_lc_all)
```

```
class annize.i18n.UnspecifiedCulture
```

Bases: [Culture](#)

```
class annize.i18n.IdCulture
```

Bases: [Culture](#)

**class** annize.i18n.\_CultureFence

Bases: [Culture](#)

annize.i18n.get\_culture(*culture*, \*, *fallback\_cultures*=())

Parameters

- **culture** ([Culture](#) | *str* | *None*)
- **fallback\_cultures** (*Iterable*[[Culture](#)])

Return type

[Culture](#) | *None*

annize.i18n.current\_culture()

Return type

[Culture](#)

annize.i18n.annize\_user\_interaction\_culture()

Return type

[Culture](#)

annize.i18n.friendly\_join\_string\_list(*strlist*)

Parameters

**strlist** (*list*[[TrStr](#) | *str*])

Return type

[TrStr](#)

**exception** annize.i18n.NoCurrentCultureError

Bases: *TypeError*

**exception** annize.i18n.TranslationUnavailableError(*stringname*, *language*)

Bases: *TypeError*

Parameters

- **stringname** (*str*)
- **language** (*str*)

## annize.object package

Handling of Annize objects.

There is no particular subclass that all Annize objects inherit from! Annize objects can be of arbitrary types.

There are some decorators for optional configuration and finetuning of Annize objects' methods and attributes here.

In submodules, there are routines for object and object type handling, internally used by the infrastructure.

annize.object.explicit\_only(*arg\_name*)

Parameters

**arg\_name** (*str*)

## Submodules

### annize.object.controller module

TODO.

**class** annize.object.controller.\_CreateObjectHelper

Bases: object

**class** ParameterConfig(*parameter\_name: str, explicit\_only: bool | None = None*)

Bases: object

#### Parameters

- **parameter\_name** (*str*)
- **explicit\_only** (*bool | None*)

**parameter\_name:** *str*

**explicit\_only:** *bool | None = None*

**with\_updates**(*oconfig*)

#### Parameters

**oconfig** (*ParameterConfig*)

#### Return type

*ParameterConfig*

**static** create\_object(*call\_type, args, kwargs*)

#### Parameters

- **args** (*Iterable*)
- **kwargs** (*dict*)

**static** \_CreateObjectHelper\_\_convert\_kwargs\_from\_string(*argument\_infos, args, kwargs*)

**static** \_CreateObjectHelper\_\_determine\_matching\_keywords\_for\_arg(*arg, call\_type, argument\_infos*)

**static** \_CreateObjectHelper\_\_fill\_empty\_lists(*argument\_infos, args, kwargs*)

**static** \_CreateObjectHelper\_\_fill\_unspecified\_optionals(*argument\_infos, args, kwargs*)

**static** \_CreateObjectHelper\_\_get\_parameter\_config(*call\_type, arg\_name*)

#### Parameters

- **call\_type** (*type*)
- **arg\_name** (*str*)

#### Return type

*ParameterConfig*

**static** \_CreateObjectHelper\_\_put\_item\_into\_kwargs(*arg, kwargs, kwarg\_name, param\_type\_info*)

**static** \_CreateObjectHelper\_\_shift\_args\_to\_kwargs(*call\_type, argument\_infos, args, kwargs*)

`annize.object.controller.create_object(call_type, args, kwargs)`

**Parameters**

- **args** (*Iterable*)
- **kwargs** (*dict*)

**exception** `annize.object.controller.MultipleValuesForSingleArgumentError(argname)`

Bases: `TypeError`

**Parameters**

**argname** (*str*)

## `annize.object.parameter_info` module

TODO.

**class** `annize.object.parameter_info.ParameterInfo(name, resolved_type, is_optional, construct_from_string_func)`

Bases: `object`

**Parameters**

- **name** (*str*)
- **resolved\_type** (*type* | *None*)
- **is\_optional** (*bool*)
- **construct\_from\_string\_func** (*Callable[[str], Any]* | *None*)

**property** `name`: *str*

**property** `resolved_type`: *type* | *None*

**matches\_object**(*obj*)

**Parameters**

**obj** (*object*)

**Return type**

*bool*

**matches\_type**(*type\_*)

**Parameters**

**type\_** (*type*)

**Return type**

*bool*

**matches\_inner\_type**(*type\_*)

**Parameters**

**type\_** (*type*)

**Return type**

*bool*

**property** `is_optional`: *bool*

**property** `inner_type_info`: [\*ParameterInfo\*](#) | *None*

```

property allows_multiple_args: bool
property is_constructable_from_string: bool
construct_from_string(s)

```

**Parameters**  
*s* (*str*)

**Return type**  
*Any*

```
class annize.object.parameter_info.ListParameterInfo(name, is_optional, innertypeinfo)
```

Bases: [ParameterInfo](#)

**Parameters**

- **name** (*str*)
- **is\_optional** (*bool*)

```
property allows_multiple_args
```

```
property inner_type_info
```

```
class annize.object.parameter_info.UnionParameterInfo(name, is_optional,  

                                                    union_member_type_infos)
```

Bases: [ParameterInfo](#)

**Parameters**

- **name** (*str*)
- **is\_optional** (*bool*)

```
property resolved_type
```

```
matches_object(obj)
```

```
annize.object.parameter_info._get_type_info(for_type, as_optional=False)
```

**Parameters**  
**as\_optional** (*bool*)

**Return type**  
[ParameterInfo](#)

```
annize.object.parameter_info.type_parameter_infos(*, for_callable)
```

**Parameters**  
**for\_callable** (*Callable*)

**Return type**  
dict[str, [ParameterInfo](#)]

## annize.project package

Annize projects.

See [Project](#), [Node](#) and also the submodules.

**class** `annize.project.Project`(*node*, *annize\_config\_rootpath*)

Bases: `object`

An Annize project.

The configuration structure is available in *node*.

Do not use directly.

Load a project with *annize.project.loader*. Create a fresh project with `TODO`.

**Parameters**

- **node** (*ProjectNode*)
- **annize\_config\_rootpath** (*str* / *Path*)

**property node:** *ProjectNode*

The project node.

This contains the entire configuration structure of this project.

**property annize\_config\_rootpath:** *Path*

The “config root path” of this Annize project.

This is usually not the same as the project’s “root path”, but a subdirectory like ‘-meta’ inside it.

**static load**(*project\_path*)

Load a project from disk. Return `None` if the given path does not point into an Annize project.

**Parameters**

**project\_path** (*str* / *Path*) – A path somewhere inside the project to be opened.

**Return type**

*Project* | `None`

**save**()

Save the current state of the project back to disk.

**static create\_new**(*project\_root\_path*, *subdirectory\_name*='meta')

Create a new Annize project.

This will create an initial version of the Annize project configuration on disk as well.

**Parameters**

- **project\_root\_path** (*str* / *Path*) – The project root path.
- **subdirectory\_name** (*str*) – The subdirectory name where to store the Annize configuration files inside the project root directory. This is not arbitrary but must be one of the well known ones!

**Return type**

*Project*

**class** `annize.project.Node`

Bases: `ABC`

Nodes are the building blocks of a project.

They exist in a serialized way in the project files (usually xml), and when the project is loaded to memory (see *annize.project.loader*) they are represented by a structure of `Node` instances.

Each Node has various features (see methods and properties of this class), e.g. it can be observed for changes. Each node can also have children. This is just a base class for more specific node types, though. See also its subclasses in the same module.

The most relevant subclass in many regards is *ObjectNode*.

**class ChangeEvent**(*target\_node*)

Bases: object

Base class for events on a *Node*. See subclasses and *Node.add\_change\_handler()*.

**Parameters**

**target\_node** (*Node*)

**property target\_node:** *Node*

The target node this event is about.

**class ChildrenListChangeEvent**(*target\_node*, *child\_node*, *child\_position*)

Bases: *ChangeEvent*

Base class for events on a *Node* that are about changes on the list of children. See subclasses.

**Parameters**

- **target\_node** (*Node*)
- **child\_node** (*Node*)
- **child\_position** (*int*)

**property child\_node:** *Node*

The child node this event is about.

**property child\_position:** *int*

The position of the child node in the list of children.

**class ChildAddedEvent**(*target\_node*, *child\_node*, *child\_position*)

Bases: *ChildrenListChangeEvent*

Node event that occurs when a child node was added.

**Parameters**

- **target\_node** (*Node*)
- **child\_node** (*Node*)
- **child\_position** (*int*)

**class ChildRemovedEvent**(*target\_node*, *child\_node*, *child\_position*)

Bases: *ChildrenListChangeEvent*

Node event that occurs when a child node was removed.

**Parameters**

- **target\_node** (*Node*)
- **child\_node** (*Node*)
- **child\_position** (*int*)

**class** `PropertyChangedEvent`(*target\_node*, *property\_name*, *old\_value*, *new\_value*)

Bases: [ChangeEvent](#)

Node event that occurs when a property of a node was changed.

**Parameters**

- **target\_node** ([Node](#))
- **property\_name** (*str*)
- **old\_value** (*Any*)
- **new\_value** (*Any*)

**property property\_name:** *str*

The property name.

**property old\_value:** *Any*

The old property value.

**property new\_value:** *Any*

The new property value.

**add\_change\_handler**(*handler*, \*, *also\_watch\_children*)

Add a function that handles changes on this node.

See also [remove\\_change\\_handler\(\)](#).

**Parameters**

- **handler** (*Callable*[[[ChangeEvent](#)], *None*]) – The handler function to add.
- **also\_watch\_children** (*bool*) – Whether this function shall also observe this node's children.

**remove\_change\_handler**(*handler*)

Remove a change handler function that was added by [add\\_change\\_handler\(\)](#) earlier.

If that function was added multiple times, it will remove all of them. If the function was not added, this will do nothing.

**Parameters**

**handler** (*Callable*[[[ChangeEvent](#)], *None*]) – The handler function to remove.

**\_\_changed\_\_helpers**(*event*)

**Parameters**

**event** ([ChangeEvent](#))

**\_changed\_\_child\_added**(*child\_node*, *child\_position*)

**Parameters**

- **child\_node** ([Node](#))
- **child\_position** (*int*)

**\_changed\_\_child\_removed**(*child\_node*, *child\_position*)

**Parameters**

- **child\_node** ([Node](#))
- **child\_position** (*int*)



**`_changed__property_changed`**(*node*, *property\_name*, *old\_value*, *new\_value*)

**Parameters**

- **`node`** (*Node*)
- **`property_name`** (*str*)
- **`old_value`** (*Any*)
- **`new_value`** (*Any*)

**`property parent`**: *Node* | *None*

This node's parent node.

**`property children`**: *Iterable*[*Node*]

This node's child nodes.

**`insert_child`**(*i*, *node*)

Insert a new child node.

**Parameters**

- **`i`** (*int*) – The position.
- **`node`** (*Node*) – The node to insert.

**Return type**

*None*

**`append_child`**(*node*)

Append a new child node.

**Parameters**

**`node`** (*Node*) – The node to append.

**Return type**

*None*

**`remove_child`**(*node*)

Remove a child node.

If that node is not a child node, it raises a *ValueError*.

**Parameters**

**`node`** (*Node*) – The node to remove.

**Return type**

*None*

**`abstractmethod classmethod _allowed_child_types`**()

Return a list of node types that this node type allows to have as child nodes.

**Return type**

*Iterable*[*type*[*Node*]]

**`clone`**(*with\_children=True*, *with\_marshallers=False*)

**Parameters**

- **`with_children`** (*bool*)
- **`with_marshallers`** (*bool*)

**Return type**[Node](#)**description**(\*, *with\_children=True, multiline=True*)**Parameters**

- **with\_children** (*bool*)
- **multiline** (*bool*)

**Return type**

str

**\_\_description**(*indent, with\_children, multiline*)**Parameters**

- **indent** (*int*)
- **with\_children** (*bool*)
- **multiline** (*bool*)

**Return type**

str

**abstractmethod** **\_str\_helper**()**Return type**[Iterable](#)[str]**\_abc\_impl** = <[\\_abc.\\_abc\\_data](#) object>**class** [annize.project.ProjectNode](#)Bases: [Node](#)

An Annize project root node.

Each project has exactly one root node. It has no parent. Its children are the Annize project configuration files. It has no direct serialized representation (or, one could argue, it is the directory that contains these files).

**save**()

Store the current state to the Annize project configuration files.

**insert\_child**(*i, node*)

Insert a new child node.

**Parameters**

- **i** – The position.
- **node** – The node to insert.

**\_\_changed\_handler**(*event*)**Parameters****event** ([ChangeEvent](#))**get\_changes**(\* , *since=0, until=9223372036854775807*)

Return all changes that happened to the project, since the moment of loading it or any later point in time, and until now or any earlier point in time.

All timestamp arguments are based on a virtual clock (which basically increases by 1 for each change). See [TODO](#).

**Parameters**

- **since** (*int*) – The timestamp where to start with returning changes (inclusive).
- **until** (*int*) – The timestamp where to stop with return changes (non-inclusive).

**Return type**list[[ChangeEvent](#)]**\_\_compacted\_changelist()****Parameters****events** (*list*[[ChangeEvent](#)])**Return type**list[[ChangeEvent](#) | None]**undo\_changes**(*since*)

Undo all changes that happened to the project since a given point in time.

**Parameters****since** (*int*) – The timestamp where to start with undoing changes (inclusive).**Return type**

None

**static load**(*path*)**Parameters****path** (*str* | *Path*)**Return type**[ProjectNode](#)**classmethod \_allowed\_child\_types()**

Return a list of node types that this node type allows to have as child nodes.

**\_str\_helper()****\_abc\_impl** = <[\\_abc.\\_abc\\_data](#) object>**class** annize.project.**FileNode**(*path*, *marshaller*)Bases: [Node](#)

An Annize project file node.

Each project has one file node per configuration file. They are the children of the [ProjectNode](#). The children of a file node are mostly of type [ObjectNode](#), but can also be different ones.

**Parameters**

- **path** (*str* | *Path*)
- **marshaller** (*TODO*)

**property path:** *Path*

The file path.

**property marshaller:** *TODO***\_str\_helper()**

**classmethod** `_allowed_child_types()`

Return a list of node types that this node type allows to have as child nodes.

`_abc_impl = <_abc._abc_data object>`

**class** `annize.project.ArgumentNode`

Bases: [Node](#), `ABC`

Base class for nodes that can be used as an argument, usually in an [ObjectNode](#).

See subclasses.

**property** `name: str | None`

The name of this argument node.

Names are used for a few purposes (the documentation will mention that where it is important), but primarily you can refer to a named argument with a [ReferenceNode](#) and you can use it for [append\\_to](#).

**property** `append_to: str | None`

The name of another argument node where this argument node gets appended to its children at runtime.

This essentially makes this argument node appear twice at runtime. It will also be in the place where it was defined; just a reference to that argument is created as a result.

**property** `arg_name: str | None`

The argument name where this argument is associated to in the parent object.

Valid argument names depend on the type of object that the parent is representing.

`_str_helper()`

`_abc_impl = <_abc._abc_data object>`

**class** `annize.project.ObjectNode`

Bases: [ArgumentNode](#)

An Annize project object node.

In a typical Annize project, most nodes are object nodes. Most structure in their project files represent them (usually the tags in xml files). All the other node types are basically related to containing object nodes (like file nodes or the project root node) or have other support purposes.

Children are mostly other object nodes, [ScalarValueNode](#) or [ReferenceNode](#). They are associated to a particular parameter name (of the object type) by their [ArgumentNode.arg\\_name](#).

**property** `type_name: str`

The name of the type of this object.

**property** `feature: str`

The Annize feature name that provides this object.

`_str_helper()`

**classmethod** `_allowed_child_types()`

Return a list of node types that this node type allows to have as child nodes.

`_abc_impl = <_abc._abc_data object>`

**class** annize.project.ScalarValueNode

Bases: [ArgumentNode](#)

An Annize project scalar value node.

It represents a fixed string value.

**property value:** str

The string that this node represents.

**\_str\_helper()**

**\_\_shorten(maxlen=100)**

**Parameters**

- **obj** (*Any*)
- **maxlen** (*int*)

**Return type**

str

**classmethod \_allowed\_child\_types()**

Return a list of node types that this node type allows to have as child nodes.

**\_abc\_impl** = <\_abc.\_abc\_data object>

**class** annize.project.ReferenceNode

Bases: [ArgumentNode](#)

A reference node.

This node represents a reference to another node (by its [ArgumentNode.name](#))

**class OnUnresolvableAction(\*values)**

Bases: Enum

**FAIL** = 'fail'

**SKIP** = 'skip'

**property reference\_key:** str

The name of the node this node references to.

**property on\_unresolvable:** [OnUnresolvableAction](#)

**\_str\_helper()**

**classmethod \_allowed\_child\_types()**

Return a list of node types that this node type allows to have as child nodes.

**\_abc\_impl** = <\_abc.\_abc\_data object>

**class** annize.project.BlockNode

Bases: [Node](#)

A block node without any own behavior.

The purpose of that block differs for particular subclass.

**\_str\_helper()**

```
classmethod _allowed_child_types()
```

Return a list of node types that this node type allows to have as child nodes.

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.project.BlockNodeWithScope
```

Bases: [BlockNode](#)

Base class for a block node with an additional scope definition.

The purpose of the block and the scope definition depend on the particular subclass.

```
class Scope(*values)
```

Bases: Enum

```
BLOCK = 'block'
```

```
FILE = 'file'
```

```
PROJECT = 'project'
```

```
_str_helper()
```

```
property scope: Scope
```

The scope of this block.

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.project.OnFeatureUnavailableNode
```

Bases: [BlockNodeWithScope](#)

An Annize project on-Feature-unavailable definition node.

They control how Annize behaves when the project configuration refers to a Feature that is not available.

The scope defines whether this rule applies only for the block, for the entire file that contains it, or for the entire project, while [do](#) defines if and how much gets ignored when the specified Feature is not available.

```
class Action(*values)
```

Bases: Enum

```
FAIL = 'fail'
```

```
SKIP_BLOCK = 'skip_block'
```

```
SKIP_NODE = 'skip_node'
```

```
_str_helper()
```

```
property feature: str
```

The name of the Feature that gets checked by this node. Empty string or \* (the default) means all features.

```
property do: Action
```

The action when the specified Feature is not available.

```
_abc_impl = <_abc._abc_data object>
```

```
exception annize.project.FeatureUnavailableError(feature_name)
```

Bases: ModuleNotFoundError

Parameters

**feature\_name** (*str*)

**exception** `annize.project.BadStructureError(message)`

Bases: `ValueError`

**Parameters**

**message** (*str*)

**exception** `annize.project.MaterializerError(message)`

Bases: `TypeError`

**Parameters**

**message** (*str*)

**exception** `annize.project.ParserError(message)`

Bases: `ValueError`

Parsing error like bad input xml.

**Parameters**

**message** (*str*)

**exception** `annize.project.UnresolvableReferenceError(reference_key)`

Bases: `MaterializerError`

**Parameters**

**reference\_key** (*str*)

## Subpackages

### `annize.project.file_formats` package

File formats for Annize configuration files.

See also the submodules.

**class** `annize.project.file_formats.FileFormat`

Bases: `ABC`

A file format for Annize configuration files.

**class** `Marshaler`

Bases: `ABC`

**abstractmethod** `add_change(change)`

**Parameters**

**change** (*TODO*)

**Return type**

`None`

`_abc_impl = <_abc._abc_data object>`

**classmethod** `parse_file(path)`

Read the given file and return a project file node for it.

**Parameters**

**path** (*str* / *Path*) – The file to parse.

**Return type**

`FileNode`

`_abc_impl = <_abc._abc_data object>`

`annize.project.file_formats.register_file_format(format_name)`

**Parameters**

**format\_name** (*str*)

`annize.project.file_formats.get_format(format_name)`

**Parameters**

**format\_name** (*str*)

**Return type**

[FileFormat](#) | `None`

`annize.project.file_formats.parse(path)`

**Parameters**

**path** (*str* | *Path*)

**Return type**

[ProjectNode](#)

## Submodules

### `annize.project.file_formats.xml` module

Support for Annize configuration XML files.

**class** `annize.project.file_formats.xml.XmlFileFormat`

Bases: [FileFormat](#)

**classmethod** `parse_file(path)`

Read the given file and return a project file node for it.

**Parameters**

**path** – The file to parse.

`_abc_impl` = `<_abc._abc_data object>`

**class** `annize.project.file_formats.xml._XmlParser`

Bases: `object`

**class** `_Context`

Bases: `object`

**property** `marshaller`

**in\_file**(*fpath*, *marshaller*)

**Parameters**

**fpath** (*str* | *Path*)

**in\_node**(*xnode*)

**static** `_Context.__nodeshort(xnode, maxlen=100)`

**Parameters**

**xnode** (*Element*)

**class** `_TagParts(name, namespace='')`

Bases: `object`

**Parameters**

**namespace** (*str*)



```

ATTRIBUTE_COMMAND_START = '~'

ATTRIBUTE_COMMAND_END = '~'

classmethod escape_attribute_string(txt)

    Parameters
        txt (str)

    Return type
        str

parse_file(fpath)

    Parameters
        fpath (str | Path)

    Return type
        FileNode

classmethod _XmlParser__interpret_attribute_string(txt)

    Parameters
        txt (str)

    Return type
        Tuple[bool, str]

classmethod _XmlParser__parse_attrib(key, value)

    Parameters
        • key (str)
        • value (str)

    Return type
        Node

 XmlParser__parse_child(node, xnode)

    Parameters
        • node (Node)
        • xnode (Element)

    Return type
        Tuple[Node, Element]

 XmlParser__parse_children(node, xparent)

    Parameters
        • node (Node)
        • xparent (Element)

 XmlParser__parse_tag(node, argname, callname, feature, xnode)

    Parameters
        node (Node)

class annize.project.file_formats.xml.Marshaler
    Bases: Marshaler

```

```
class XmlDocumentLocation(element: <cyfunction Element at 0x7fd179aa1220>, attr_name: str = '')
    Bases: object

    Parameters
        • element (Element)
        • attr_name (str)

    element: Element
    attr_name: str = ''

    add_change(change)

    add_element(node, xelem)

        Parameters
            • node (Node)
            • xelem (Element)

    add_element_attr(node, xelem, attrname)

        Parameters
            • node (Node)
            • xelem (Element)
            • attrname (str)

    _abc_impl = <_abc._abc_data object>

    add_element_tree(node, xtree)

        Parameters
            • node (Node)
            • xtree (ElementTree)

    save_filenode_to_file(node)

        Parameters
            • node (FileNode)
```

### annize.project.materializer package

Materializing of Annize projects into a working runtime structure (usually used by the Runner application).

See *materialize()*.

All submodules are only used internally by this one. There is a core part, some preprocessor functions, and some behaviors that implement what it does for different types of project nodes.

```
class annize.project.materializer.MaterializationResult(root_objects, node_association, problems)
    Bases: object

    Parameters
        • root_objects (list[Any])
        • node_association (dict[Node, list[Any]])
```

```

        • problems (dict[Node | None, list[Exception]])
property root_objects: list[Any]
objects_for_node(node)

    Parameters
        node (Node)

    Return type
        list[Any] | None

erroneous_nodes()

    Return type
        list[Node]

errors_for_node(node)

    Parameters
        node (Any)

    Return type
        list[Exception]

annize.project.materializer.materialize(project, *, feature_loader=None)

    Parameters
        • project (ProjectNode)
        • feature_loader (FeatureLoader | None)

    Return type
        MaterializationResult

annize.project.materializer._translate_from_clone(real_nodes_for_clones, node_association, errors)
annize.project.materializer._node_clone_link(original, clone)

```

## Subpackages

### [annize.project.materializer.behaviors](#) package

Behaviors.

See [Behavior](#).

**class** [annize.project.materializer.behaviors.Behavior](#)

Bases: [ABC](#)

A behavior implements what the materializer does for a given node. See subclasses in the submodules.

**abstractmethod** **node\_context**(*nodemat*)

For a node, the materializer will enter the context returned by this function for all behaviors.

The materializer itself does that for the root node. Behaviors itself are responsible for triggering that same process on children.

So, any node gets materialized in the context of all behaviors on all parent nodes. Actual materialization logic happens in this function, in the course of setting up and taking down all these contexts.

**Parameters**

**nodemat** ([NodeMaterialization](#)) – The node materialization for the current node.

**Return type***ContextManager*`_abc_impl = <_abc._abc_data object>`**Submodules****annize.project.materializer.behaviors.argument module**

See *ArgumentBehavior* and *AssociateArgumentNodeBehavior*.

```
class annize.project.materializer.behaviors.argument.ArgumentBehavior(callfct, *,  
                                                                    feature_loader)
```

Bases: *Behavior*

Behavior that handles argument nodes (incl. creation of an object for an object node).

**Parameters**

**feature\_loader** (*FeatureLoader*)

**node\_context**(*nodemat*)

For a node, the materializer will enter the context returned by this function for all behaviors.

The materializer itself does that for the root node. Behaviors itself are responsible for triggering that same process on children.

So, any node gets materialized in the context of all behaviors on all parent nodes. Actual materialization logic happens in this function, in the course of setting up and taking down all these contexts.

**Parameters**

**nodemat** – The node materialization for the current node.

`_abc_impl = <_abc._abc_data object>`

```
class annize.project.materializer.behaviors.argument.AssociateArgumentNodeBehavior(association)
```

Bases: *Behavior*

**Parameters**

**association** (*dict*[*ArgumentNode*, *list*[*Any*]])

**node\_context**(*nodemat*)

For a node, the materializer will enter the context returned by this function for all behaviors.

The materializer itself does that for the root node. Behaviors itself are responsible for triggering that same process on children.

So, any node gets materialized in the context of all behaviors on all parent nodes. Actual materialization logic happens in this function, in the course of setting up and taking down all these contexts.

**Parameters**

**nodemat** – The node materialization for the current node.

`_abc_impl = <_abc._abc_data object>`**annize.project.materializer.behaviors.basket module**

See *BasketBehavior*.

**class** annize.project.materializer.behaviors.basket.**BasketBehavior**

Bases: *Behavior*

Behavior that handles baskets.

**node\_context**(*nodemat*)

For a node, the materializer will enter the context returned by this function for all behaviors.

The materializer itself does that for the root node. Behaviors itself are responsible for triggering that same process on children.

So, any node gets materialized in the context of all behaviors on all parent nodes. Actual materialization logic happens in this function, in the course of setting up and taking down all these contexts.

**Parameters**

**nodemat** – The node materialization for the current node.

**\_abc\_impl** = <\_abc.\_abc\_data object>

## annize.project.materializer.behaviors.block module

See *BlockBehavior*.

**class** annize.project.materializer.behaviors.block.**BlockBehavior**

Bases: *Behavior*

Behavior that handles block.

**node\_context**(*nodemat*)

For a node, the materializer will enter the context returned by this function for all behaviors.

The materializer itself does that for the root node. Behaviors itself are responsible for triggering that same process on children.

So, any node gets materialized in the context of all behaviors on all parent nodes. Actual materialization logic happens in this function, in the course of setting up and taking down all these contexts.

**Parameters**

**nodemat** – The node materialization for the current node.

**\_abc\_impl** = <\_abc.\_abc\_data object>

## annize.project.materializer.behaviors.feature\_unavailable module

See *FeatureUnavailableBehavior*.

**class**

annize.project.materializer.behaviors.feature\_unavailable.**FeatureUnavailableBehavior**

Bases: *Behavior*

Behavior that handles on-feature-unavailable nodes.

**\_\_context\_skipnode\_featureignorelist**(*node*)

**\_\_context\_catchexceptions**(*nodemat*, *featureignorelist*)

**node\_context**(*nodemat*)

For a node, the materializer will enter the context returned by this function for all behaviors.

The materializer itself does that for the root node. Behaviors itself are responsible for triggering that same process on children.

So, any node gets materialized in the context of all behaviors on all parent nodes. Actual materialization logic happens in this function, in the course of setting up and taking down all these contexts.

**Parameters**

**nodemat** – The node materialization for the current node.

**\_abc\_impl** = <\_abc.\_abc\_data object>

**annize.project.materializer.behaviors.reference module**

See *ReferenceBehavior*.

**class** annize.project.materializer.behaviors.reference.ReferenceBehavior

Bases: *Behavior*

Behavior that handles reference nodes.

**node\_context**(*nodemat*)

For a node, the materializer will enter the context returned by this function for all behaviors.

The materializer itself does that for the root node. Behaviors itself are responsible for triggering that same process on children.

So, any node gets materialized in the context of all behaviors on all parent nodes. Actual materialization logic happens in this function, in the course of setting up and taking down all these contexts.

**Parameters**

**nodemat** – The node materialization for the current node.

**\_abc\_impl** = <\_abc.\_abc\_data object>

**Submodules****annize.project.materializer.core module**

Inner core parts of the project materializer. Only used internally by the parent package.

**class** annize.project.materializer.core.NodeMaterialization(*materializer, node, store*)

Bases: object

**Parameters**

- **materializer** (*ProjectMaterializer*)
- **node** (*Node*)
- **store** (*dict*)

**property node:** *Node*

**set\_materialized\_result**(*resultlist*)

**set\_problems**(*problems*)

**Parameters**

**problems** (*Iterable[Exception]*)

**get\_materialized\_children\_tuples**()

**get\_materialized\_children()**

**Return type**

*Iterable*[Any]

**try\_get\_materialization\_for\_node(*node*)**

**Parameters**

**node** (Node)

**property** has\_result

**property** result

**property** problems: list[Exception]

**class** annize.project.materializer.core.**ProjectMaterializer**(*node*, \*, *behaviors*)

Bases: object

**Parameters**

- **node** (Node)

- **behaviors** (*Iterable*[annize.project.materializer.behaviors.Behavior])

**\_\_materialization\_for\_node(*node*, *store*)**

**Parameters**

- **node** (Node)

- **store** (*dict*)

**Return type**

NodeMaterialization

**\_\_materialize(*node*, *store*)**

**Parameters**

- **node** (Node)

- **store** (*dict*)

**Return type**

None

**\_materialize\_hlp\_childobjs(*node*, *store*)**

**Parameters**

- **node** (Node)

- **store** (*dict*)

**Return type**

list[Tuple[Node, list[Any]]]

**\_\_get\_erroneous\_nodes(*materializationstore*, *old\_erroneous\_nodes*)**

**get\_materialized()**

**Return type**

Tuple[list[Any] | None, dict[Node, list[Exception]]]

**exception** `annize.project.materializer.core.InternalError`

Bases: `Exception`

**exception** `annize.project.materializer.core.ChildrenNotMaterializableError(node)`

Bases: [\*InternalError\*](#)

**Parameters**

**node** ([\*Node\*](#))

## **`annize.project.materializer.preprocessors` module**

Some preprocessor functions used by the materializer.

Only used internally by the parent package.

`annize.project.materializer.preprocessors.resolve_appendtonodes(topnode)`

**Parameters**

**topnode** ([\*Node\*](#))

**Return type**

[\*Node\*](#)

`annize.project.materializer.preprocessors.normalize_blockscopes(topnode)`

**Parameters**

**topnode** ([\*Node\*](#))

**Return type**

[\*Node\*](#)

## **Submodules**

### **`annize.project.feature_loader` module**

Feature module loader.

See [\*FeatureLoader\*](#).

**class** `annize.project.feature_loader.FeatureLoader`

Bases: `ABC`

Base class for a feature module loader.

**abstractmethod** `load_feature(name)`

**Parameters**

**name** (*str*)

**Return type**

*Any* | `None`

**abstractmethod** `get_all_available_feature_names()`

**Return type**

`list[str]`

`_abc_impl = <_abc._abc_data object>`



```
class annize.project.feature_loader.DefaultFeatureLoader
```

Bases: [FeatureLoader](#)

Default feature module loader.

```
_FEATURES_NAMESPACE = 'annize.features'
```

```
_COMMON_NAMESPACE_POSTFIX = 'common'
```

```
load_feature(name)
```

```
get_all_available_feature_names()
```

```
__find_feature_modules_in_package(package_name)
```

**Parameters**

**package\_name** (*str*)

**Return type**

list[str]

```
_abc_impl = <_abc._abc_data object>
```

## annize.project.inspector module

Project inspector.

See [Inspector](#).

```
class annize.project.inspector.Inspector(feature_loader)
```

Bases: object

Inspectors are used in order to get various additional metadata about parts of a project, which are useful e.g. for project configuration UIs.

**Parameters**

**feature\_loader** ([FeatureLoader](#))

```
class ArgumentMatchings(all_matchings)
```

Bases: object

**Parameters**

**all\_matchings** (list[[ArgumentMatching](#)])

```
class ArgumentMatching(arg_name, nodes, allows_multiple_args)
```

Bases: object

**Parameters**

- **arg\_name** (*str*)
- **nodes** (list[[ArgumentNode](#)])
- **allows\_multiple\_args** (*bool*)

**property argname:** str

**property allows\_multiple\_args:** bool

**property nodes:** list[[ArgumentNode](#)]

```
matching_by_argname(arg_name)
```

**Parameters**

**arg\_name** (*str*)

**Return type***ArgumentMatching* | None**all()****Return type**list[*ArgumentMatching*]**class TypeInfo**(*feature*, *typename*, *ctype*)

Bases: object

**Parameters**

- **feature** (*str* | None)
- **typename** (*str*)
- **ctype** (*type*)

**property feature:** *str* | None**property typename:** *str***property type:** *type***match\_arguments**(*node*)**Parameters****node** (*Node*)**Return type***ArgumentMatchings***match\_node**(*node*)**Parameters****node** (*Node*)**Return type***ArgumentMatching* | None**get\_all\_types**()**Return type**list[*TypeInfo*]**get\_types\_for\_argument**(*node*, *argname*)**Parameters**

- **node** (*Node*)
- **argname** (*str*)

**Return type**list[*TypeInfo*]**get\_project\_node**(*node*)**Parameters****node** (*Node*)**Return type***ProjectNode* | None

**get\_node\_by\_name**(*subtree, name*)

**Parameters**

- **subtree** ([ProjectNode](#))
- **name** (*str*)

**Return type**

[Node](#) | None

**resolve\_reference\_node**(*node, \*, deep=True*)

**Parameters**

- **node** ([Node](#))
- **deep** (*bool*)

**Return type**

[ArgumentNode](#) | None

**\_\_get\_node\_materialtype**(*node*)

**Parameters**

**node** ([Node](#))

**Return type**

*type* | None

## annize.project.loader module

Loading Annize projects from disk.

See also [load\\_project\(\)](#).

**annize.project.loader.load\_project**(*project\_path*)

Load a project from disk. Return None if the given path does not lead to a location inside an Annize project.

Do not use it directly. See [annize.project.Project.load\(\)](#).

**Parameters**

**project\_path** (*str* | *Path*) – A path to somewhere inside an Annize project.

**Return type**

[Project](#) | None

**annize.project.loader.find\_project\_annize\_config\_root\_file**(*project\_path*)

Return the main configuration file for an Annize project given by a path (the path may point to somewhere inside the project; not only inside the Annize configuration directory), or None if the given path does not lead to a location inside an Annize project.

**Parameters**

**project\_path** (*str* | *Path*) – A path into the Annize project.

**Return type**

*Path* | None

**annize.project.loader.project\_root\_directory**(*annize\_config\_rootpath*)

**Parameters**

**annize\_config\_rootpath** (*str* | *Path*)

**Return type**

*Path*

**annize.ui package**

`annize.ui.app(app_name, **kwargs)`

**Parameters**

`app_name (str)`

**Subpackages**

**annize.ui.apps package**

**Subpackages**

**annize.ui.apps.runner package**

**Subpackages**

**annize.ui.apps.runner.models package**

**Submodules**

**annize.ui.apps.runner.models.main module**

**annize.ui.apps.runner.models.task\_chooser module**

**annize.ui.apps.runner.models.task\_execution module**

**annize.ui.apps.runner.models.user\_feedback module**

**annize.ui.apps.runner.views package**

**Submodules**

**annize.ui.apps.runner.views.main module**

**annize.ui.apps.runner.views.task\_chooser module**

**annize.ui.apps.runner.views.task\_execution module**

**annize.ui.apps.runner.views.user\_feedback module**

**annize.ui.apps.studio package**

**Subpackages**

**annize.ui.apps.studio.models package**

**Submodules**

**annize.ui.apps.studio.models.add\_child module**

**annize.ui.apps.studio.models.main module**

**annize.ui.apps.studio.models.main\_tab\_panel module**

`annize.ui.apps.studio.models.object_editor` module

`annize.ui.apps.studio.models.problems_list` module

`annize.ui.apps.studio.models.project_config` module

`annize.ui.apps.studio.views` package

#### Submodules

`annize.ui.apps.studio.views.add_child` module

`annize.ui.apps.studio.views.main` module

`annize.ui.apps.studio.views.main_tab_panel` module

`annize.ui.apps.studio.views.object_editor` module

`annize.ui.apps.studio.views.problems_list` module

`annize.ui.apps.studio.views.project_config` module

`annize.user_feedback` package

**class** `annize.user_feedback.UserFeedbackController`

Bases: `ABC`

**abstractmethod** `message_dialog(message, answers, config_key)`

#### Parameters

- `message` (*str*)
- `answers` (*list[str]*)
- `config_key` (*str | None*)

#### Return type

*int*

**abstractmethod** `input_dialog(question, suggested_answer, config_key)`

#### Parameters

- `question` (*str*)
- `suggested_answer` (*str*)
- `config_key` (*str | None*)

#### Return type

*str | None*

**abstractmethod** `choice_dialog(question, choices, config_key)`

#### Parameters

- `question` (*str*)
- `choices` (*list[str]*)
- `config_key` (*str | None*)

**Return type**

int | None

`_abc_impl = <_abc._abc_data object>``class annize.user_feedback.NullUserFeedbackController`

Bases: object

`message_dialog(*_)``input_dialog(*_)``choice_dialog(*_)``exception annize.user_feedback.UnsatisfiableUserFeedbackAttemptError`

Bases: RuntimeError

`annize.user_feedback._controllers_tuples_for_context(context)`**Parameters**`context` (RunContext)**Return type**

list[Tuple[int, UserFeedbackController]]

`annize.user_feedback._controllers_for_context(context)`**Parameters**`context` (RunContext)**Return type**

list[UserFeedbackController]

`annize.user_feedback._add_controller_to_context(*, controller, context, priority_index=0)`**Parameters**

- `controller` (UserFeedbackController)
- `context` (RunContext)
- `priority_index` (int)

**Return type**

None

`annize.user_feedback.message_dialog(message, answers, *, config_key=None)`**Parameters**

- `message` (TrStr | str)
- `answers` (Iterable[TrStr | str])
- `config_key` (str | None)

**Return type**

int

`annize.user_feedback.input_dialog(message, *, suggested_answer, config_key=None)`**Parameters**

- `message` (TrStr | str)

- **suggested\_answer** (*TrStr* / *str*)
- **config\_key** (*str* / *None*)

**Return type***str* | *None*

`annize.user_feedback.choice_dialog(message, choices, *, config_key=None)`

**Parameters**

- **message** (*TrStr* / *str*)
- **choices** (*Iterable*[*TrStr* / *str*])
- **config\_key** (*str* / *None*)

**Return type***int* | *None***Submodules****annize.user\_feedback.static module**

**class** `annize.user_feedback.static.StaticUserFeedbackController`(*answers*)

Bases: *UserFeedbackController*

**Parameters**

**answers** (*dict*[*str*, *Any*])

**add\_answer**(*config\_key*, *value*)

**Parameters**

- **config\_key** (*str*)
- **value** (*Any*)

**Return type***None*

**\_\_get\_answer**(*config\_key*)

**Parameters**

**config\_key** (*str*)

**Return type***Any*

**message\_dialog**(*message*, *answers*, *config\_key*)

**input\_dialog**(*question*, *suggested\_answer*, *config\_key*)

**choice\_dialog**(*question*, *choices*, *config\_key*)

**\_abc\_impl** = *<\_abc.\_abc\_data object>*

**annize.user\_feedback.tty module**

**class** `annize.user_feedback.tty.TtyUserFeedbackController`

Bases: *UserFeedbackController*

**\_\_dialog\_frame\_message**(*message*)

**Parameters**

**message** (*str*)

**Return type**

*str*

**\_\_dialog\_frame\_configkey**(*config\_key*)

**Parameters**

**config\_key** (*str*)

**Return type**

*str*

**\_\_dialog\_frame\_actions**(*text*)

**Parameters**

**text** (*str*)

**Return type**

*str*

**\_\_action\_line**(*num*, *text*)

**Parameters**

- **num** (*Any*)
- **text** (*str*)

**Return type**

*str*

**\_\_dialog**(*message*, *config\_key*, *actionstext*)

**Parameters**

- **message** (*str*)
- **config\_key** (*str*)
- **actionstext** (*str*)

**Return type**

*str*

**message\_dialog**(*message*, *answers*, *config\_key*)

**input\_dialog**(*question*, *suggested\_answer*, *config\_key*)

**choice\_dialog**(*question*, *choices*, *config\_key*)

**\_abc\_impl** = <\_abc.\_abc\_data object>

## 6.1.2 Submodules

### 6.1.3 annize.annize\_cli module

The Annize CLI.

`annize.annize_cli.main()`



`annize.annize_cli.parser(*, only_documentation=True)`

**Parameters**

`only_documentation` (*bool*)

**Return type**

*ArgumentParser*

**class** `annize.annize_cli.Commands`(*project, with\_answers\_from\_json\_file, with\_answers\_from\_json\_string, with\_answer, \*\*\_*)

Bases: `object`

**Parameters**

- `project` (*str*)
- `with_answers_from_json_file` (*Iterable[str]*)
- `with_answers_from_json_string` (*Iterable[str]*)
- `with_answer` (*Iterable[Tuple[str, str]]*)

**class** `ConsoleRunner`(*\*, project, selected\_task=None, user\_feedback=None*)

Bases: *Runner*

**Parameters**

- `project` (*Project*)
- `selected_task` (*str | None*)
- `user_feedback` (*TODO*)

`show_task_chooser()`

`show_task_execution()`

`show_task_execution_success()`

`run_runner()`

`_abc_impl = <_abc._abc_data object>`

`project_default = '/home/pino/projects/annize'`

**classmethod** `__answers_from_json_files`(*destination, with\_answers\_from\_json\_files*)

**Parameters**

- `destination` (*dict*)
- `with_answers_from_json_files` (*Iterable[str]*)

**classmethod** `__answers_from_json_strings`(*destination, with\_answers\_from\_json\_strings*)

**Parameters**

- `destination` (*dict*)
- `with_answers_from_json_strings` (*Iterable[str]*)

**classmethod** `__answers_from_single_answers(destination, with_answers)`

**Parameters**

- **destination** (*dict*)
- **with\_answers** (*Iterable[Tuple[str, str]]*)

**do**(*task\_name*, *\*\*\_*)

**Parameters**

**task\_name** (*str*)

**studio**(*\*\*\_*)

## PYTHON MODULE INDEX

### a

- annize, 13
- annize.annize\_cli, 124
- annize.asset, 13
- annize.asset.data, 13
- annize.asset.project\_info, 13
- annize.data, 13
- annize.data.color, 13
- annize.data.container, 14
- annize.data.unique, 14
- annize.data.version, 15
- annize.features, 18
- annize.features.authors, 72
- annize.features.base, 72
- annize.features.changelog, 18
- annize.features.changelog.common, 18
- annize.features.dependencies, 19
- annize.features.dependencies.common, 19
- annize.features.dependencies.python, 20
- annize.features.distributables, 21
- annize.features.distributables.common, 21
- annize.features.distributables.debian, 23
- annize.features.distributables.flatpak, 34
- annize.features.distributables.python\_wheel, 39
- annize.features.distributables.store, 21
- annize.features.distributables.store.pypi, 21
- annize.features.distributables.tar, 42
- annize.features.documentation, 43
- annize.features.documentation.common, 53
- annize.features.documentation.sphinx, 43
- annize.features.documentation.sphinx.\_utils, 46
- annize.features.documentation.sphinx.common, 46
- annize.features.documentation.sphinx.cpp, 50
- annize.features.documentation.sphinx.doxygen\_compat, 50
- annize.features.documentation.sphinx.javascript, 52
- annize.features.documentation.sphinx.output, 43
- annize.features.documentation.sphinx.output.common, 43
- annize.features.documentation.sphinx.output.html, 44
- annize.features.documentation.sphinx.output.pdf, 45
- annize.features.documentation.sphinx.output.plaintext, 45
- annize.features.documentation.sphinx.python, 52
- annize.features.documentation.sphinx.rst, 53
- annize.features.files, 55
- annize.features.files.common, 57
- annize.features.files.transfer, 55
- annize.features.files.transfer.common, 55
- annize.features.files.transfer.ssh, 56
- annize.features.homepage, 60
- annize.features.homepage.common, 63
- annize.features.homepage.sections, 60
- annize.features.homepage.sections.about, 60
- annize.features.homepage.sections.changelog, 61
- annize.features.homepage.sections.documentation, 61
- annize.features.homepage.sections.download, 61
- annize.features.homepage.sections.gallery, 62
- annize.features.homepage.sections.imprint, 62
- annize.features.homepage.sections.license, 63
- annize.features.i18n, 66
- annize.features.i18n.common, 66
- annize.features.i18n.gettext, 67
- annize.features.injections, 67
- annize.features.injections.common, 67
- annize.features.injections.python, 68
- annize.features.licensing, 75
- annize.features.media\_galleries, 76
- annize.features.task, 77
- annize.features.testing, 68
- annize.features.testing.common, 68
- annize.features.testing.pylint, 69
- annize.features.testing.pytest, 69

- annize.features.testing.pyunit, 69
- annize.features.version, 77
- annize.features.version\_control, 70
- annize.features.version\_control.common, 70
- annize.features.version\_control.git, 71
- annize.flow, 78
- annize.flow.run\_context, 78
- annize.flow.runner, 83
- annize.fs, 85
- annize.fs.ext, 90
- annize.i18n, 91
- annize.object, 94
- annize.object.controller, 95
- annize.object.parameter\_info, 96
- annize.project, 97
- annize.project.feature\_loader, 116
- annize.project.file\_formats, 107
- annize.project.file\_formats.xml, 108
- annize.project.inspector, 117
- annize.project.loader, 119
- annize.project.materializer, 110
- annize.project.materializer.behaviors, 111
- annize.project.materializer.behaviors.argument, 112
- annize.project.materializer.behaviors.basket, 112
- annize.project.materializer.behaviors.block, 113
- annize.project.materializer.behaviors.feature\_unavailable, 113
- annize.project.materializer.behaviors.reference, 114
- annize.project.materializer.core, 114
- annize.project.materializer.preprocessors, 116
- annize.ui, 120
- annize.ui.apps, 120
- annize.user\_feedback, 121
- annize.user\_feedback.static, 123
- annize.user\_feedback.tty, 123

## Symbols

\_ABOUT\_NAME (annize.features.documentation.sphinx.common.ReadmeDocument attribute), 50  
 \_ANNIZE\_CONFIG\_ROOTPATH\_\_NAME (annize.flow.run\_context.RunContext attribute), 78  
 \_COMMON\_NAMESPACE\_POSTFIX (annize.project.feature\_loader.DefaultFeatureLoader attribute), 117  
 \_Context\_\_nodeshort() (annize.project.file\_formats.xml.\_XmlParser.\_Context static method), 108  
 \_CreateObjectHelper (class in annize.object.controller), 95  
 \_CreateObjectHelper.ParameterConfig (class in annize.object.controller), 95  
 \_CreateObjectHelper\_\_convert\_kwargs\_from\_string() (annize.object.controller.\_CreateObjectHelper static method), 95  
 \_CreateObjectHelper\_\_determine\_matching\_keywords\_for\_arg() (annize.object.controller.\_CreateObjectHelper static method), 95  
 \_CreateObjectHelper\_\_fill\_empty\_lists() (annize.object.controller.\_CreateObjectHelper static method), 95  
 \_CreateObjectHelper\_\_fill\_unspecified\_optionals() (annize.object.controller.\_CreateObjectHelper static method), 95  
 \_CreateObjectHelper\_\_get\_parameter\_config() (annize.object.controller.\_CreateObjectHelper static method), 95  
 \_CreateObjectHelper\_\_put\_item\_into\_kwargs() (annize.object.controller.\_CreateObjectHelper static method), 95  
 \_CreateObjectHelper\_\_shift\_args\_to\_kwargs() (annize.object.controller.\_CreateObjectHelper static method), 95  
 \_CultureFence (class in annize.i18n), 93  
 \_FEATURES\_NAMESPACE (annize.project.feature\_loader.DefaultFeatureLoader attribute), 117  
 \_IS\_TOPLEVEL\_OBJECT\_\_METADATA\_KEY (annize.flow.run\_context.RunContext attribute), 78  
 \_S\_CHANGE (annize.features.changelog.common.ByVersionControlSystemControl attribute), 19  
 \_S\_LABEL (annize.features.changelog.common.ByVersionControlSystemControl attribute), 19  
 \_TContent (annize.fs.ext.DynamicFile attribute), 90  
 \_TStaticContent (annize.fs.ext.DynamicFile attribute), 90  
 \_TransferHelper\_\_transfer\_piece() (annize.fs.Path.\_TransferHelper static method), 88  
 \_XmlParser (class in annize.project.file\_formats.xml), 108  
 \_XmlParser.\_Context (class in annize.project.file\_formats.xml), 108  
 \_XmlParser.\_TagParts (class in annize.project.file\_formats.xml), 108  
 \_XmlParser\_\_interpret\_attribute\_string() (annize.project.file\_formats.xml.\_XmlParser class method), 109  
 \_XmlParser\_\_parse\_attr() (annize.project.file\_formats.xml.\_XmlParser class method), 109  
 \_XmlParser\_\_parse\_child() (annize.project.file\_formats.xml.\_XmlParser method), 109  
 \_XmlParser\_\_parse\_children() (annize.project.file\_formats.xml.\_XmlParser method), 109  
 \_XmlParser\_\_parse\_tag() (annize.project.file\_formats.xml.\_XmlParser method), 109  
 \_\_action\_line() (annize.user\_feedback.tty.TtyUserFeedbackController method), 124  
 \_\_answers\_from\_json\_files() (annize.annize\_cli.Commands class method), 125  
 \_\_answers\_from\_json\_strings() (annize.annize\_cli.Commands class method), 125

<code>__answers_from_single_answers()</code> (annize.annize_cli.Commands class method), 125	<code>nize.features.documentation.sphinx.common.Document method), 47</code>
<code>__call_git()</code> (annize.features.version_control.git.VersionControlSystem method), 71	<code>__generate_prepare_shortsnippets()</code> (annize.features.documentation.sphinx.common.Document method), 47
<code>__changed_helpers()</code> (annize.project.Node method), 100	<code>__generate_section()</code> (annize.features.homepage.common.Homepage method), 65
<code>__changed_handler()</code> (annize.project.ProjectNode method), 102	<code>__generate_set_culture()</code> (annize.features.documentation.sphinx.common.Document method), 47
<code>__cleanup()</code> (annize.fs.ext.FreshTempDirectory method), 90	<code>__generate_set_misc()</code> (annize.features.documentation.sphinx.common.Document method), 47
<code>__compact_changelist()</code> (annize.project.ProjectNode method), 103	<code>__generate_set_misc_and_release()</code> (annize.features.documentation.sphinx.common.Document method), 47
<code>__context_catchexceptions()</code> (annize.project.materializer.behaviors.feature_unavailable_handler method), 113	<code>__generate_section_and_release()</code> (annize.features.documentation.sphinx.common.Document method), 47
<code>__context_skipnode_featureignorelist()</code> (annize.project.materializer.behaviors.feature_unavailable_handler method), 113	<code>__get_answers()</code> (annize.user_feedback.static.StaticUserFeedbackController method), 123
<code>__description()</code> (annize.project.Node method), 102	<code>__get_env()</code> (annize.i18n.Culture method), 93
<code>__dialog()</code> (annize.user_feedback.tty.TtyUserFeedbackController method), 124	<code>__get_erroneous_nodes()</code> (annize.project.materializer.core.ProjectMaterializer method), 115
<code>__dialog_frame_actions()</code> (annize.user_feedback.tty.TtyUserFeedbackController method), 124	<code>__get_inner_generateinfo()</code> (annize.features.documentation.sphinx.common.CompositeDocument method), 47
<code>__dialog_frame_configkey()</code> (annize.user_feedback.tty.TtyUserFeedbackController method), 124	<code>__get_node_materialtype()</code> (annize.project.inspector.Inspector method), 119
<code>__dialog_frame_message()</code> (annize.user_feedback.tty.TtyUserFeedbackController method), 123	<code>__get_refgeninfo()</code> (annize.features.documentation.sphinx.common.ApiReferenceDocument method), 48
<code>__do_run()</code> (annize.flow.runner.Runner method), 84	<code>__get_variant()</code> (annize.features.documentation.sphinx.common.RstDocument method), 49
<code>__does_exclude()</code> (annize.features.files.common.Exclude method), 58	<code>__init_py()</code> (annize.features.documentation.sphinx.doxygen_compat.Doxygen method), 52
<code>__effversion()</code> (annize.features.version_control.common.BriffPy method), 70	<code>__jsfiles()</code> (annize.features.documentation.sphinx.javascript.JavaScript method), 52
<code>__files_from_package_store()</code> (annize.features.distributables.common.Group method), 22	<code>__long_str()</code> (annize.data.unique.UniqueId method), 15
<code>__find_feature_modules_in_package()</code> (annize.project.feature_loader.DefaultFeatureLoader method), 117	<code>__materialization_for_node()</code> (annize.project.materializer.core.ProjectMaterializer method), 115
<code>__find_lcall()</code> (annize.i18n.Culture method), 93	<code>__materialize()</code> (annize.project.materializer.core.ProjectMaterializer method), 115
<code>__generate_geninfo_to_confpy()</code> (annize.features.documentation.sphinx.common.Document method), 47	<code>__object_metadata_dict()</code> (annize.flow.run_context.RunContext method), 81
<code>__generate_packagelist()</code> (annize.features.homepage.sections.download.Section method), 61	<code>__object_raw_name()</code> (annize.flow.run_context.RunContext method), 81
<code>__generate_pre_post_proc()</code> (annize.features.homepage.common.Homepage method), 65	<code>__package_store_name()</code> (annize.features.documentation.sphinx.common.Document method), 47
<code>__generate_prepare_annizeicons()</code> (annize.features.documentation.sphinx.common.Document method), 47	

`nize.features.distributables.common.Group`  
`method)`, 22  
`__patch_property_types_in_docstrings()` (an-  
`nize.features.documentation.sphinx.python.Python3ApiReferenceLanguage`  
`method)`, 52  
`__prepare_generate()` (an-  
`nize.features.documentation.sphinx.doxygen_compat.DoxygenSubtype`  
`method)`, 52  
`__put_object()` (`annize.flow.run_context.RunContext`  
`method)`, 81  
`__scanjsfile()` (`annize.features.documentation.sphinx.javascript.JavaScriptApiReferenceLanguage`  
`method)`, 52  
`__set_env()` (`annize.i18n.Culture` `method)`, 93  
`__set_success_state()` (`annize.flow.runner.Runner`  
`method)`, 85  
`__set_tasks()` (`annize.flow.runner.Runner` `method)`, 84  
`__shorten()` (`annize.project.ScalarValueNode` `method)`,  
105  
`__store_files_to_package_store()` (an-  
`nize.features.distributables.common.Group`  
`method)`, 22  
`__transfer_filter_for_exclude()` (an-  
`nize.features.files.common.Directory` `method)`,  
60  
`__translations_for_stringname()` (an-  
`nize.features.i18n.common.ProjectDefinedTranslations`  
`method)`, 66  
`__uniqueid_counter` (`annize.data.unique.UniqueId` at-  
`tribute)`, 14  
`__uniqueid_lock` (`annize.data.unique.UniqueId` at-  
`tribute)`, 14  
`_abc_impl` (`annize.annize_cli.Commands.ConsoleRunner`  
`attribute)`, 125  
`_abc_impl` (`annize.data.version.AbstractVersionPatternSegment`  
`attribute)`, 15  
`_abc_impl` (`annize.data.version.ConcatenatedVersionPatternSegment`  
`attribute)`, 17  
`_abc_impl` (`annize.data.version.NumericVersionPatternSegment`  
`attribute)`, 16  
`_abc_impl` (`annize.data.version.OptionalVersionPatternSegment`  
`attribute)`, 17  
`_abc_impl` (`annize.data.version.SeparatorVersionPatternSegment`  
`attribute)`, 16  
`_abc_impl` (`annize.features.distributables.common.PackageStore`  
`attribute)`, 22  
`_abc_impl` (`annize.features.documentation.common.Document`  
`attribute)`, 54  
`_abc_impl` (`annize.features.documentation.common.HtmlOutput`  
`attribute)`, 55  
`_abc_impl` (`annize.features.documentation.common.OutputSpec`  
`attribute)`, 54  
`_abc_impl` (`annize.features.documentation.common.PdfOutputSpec`  
`attribute)`, 55  
`_abc_impl` (`annize.features.documentation.common.PlaintextOutputSpec`  
`attribute)`, 55  
`_abc_impl` (`annize.features.documentation.sphinx.common.AboutProjectDocument`  
`attribute)`, 50  
`_abc_impl` (`annize.features.documentation.sphinx.common.ApiReferenceDocument`  
`attribute)`, 48  
`_abc_impl` (`annize.features.documentation.sphinx.common.ApiReferenceLanguage`  
`attribute)`, 48  
`_abc_impl` (`annize.features.documentation.sphinx.common.ArgparseCommand`  
`attribute)`, 49  
`_abc_impl` (`annize.features.documentation.sphinx.common.CompositeDocument`  
`attribute)`, 48  
`_abc_impl` (`annize.features.documentation.sphinx.common.Document`  
`attribute)`, 47  
`_abc_impl` (`annize.features.documentation.sphinx.common.ReadmeDocument`  
`attribute)`, 50  
`_abc_impl` (`annize.features.documentation.sphinx.common.RstDocument`  
`attribute)`, 50  
`_abc_impl` (`annize.features.documentation.sphinx.cpp.CppApiReferenceLanguage`  
`attribute)`, 50  
`_abc_impl` (`annize.features.documentation.sphinx.doxygen_compat.DoxygenSubtype`  
`attribute)`, 52  
`_abc_impl` (`annize.features.documentation.sphinx.javascript.JavaScriptApiReferenceLanguage`  
`attribute)`, 52  
`_abc_impl` (`annize.features.documentation.sphinx.output.common.OutputConfiguration`  
`attribute)`, 44  
`_abc_impl` (`annize.features.documentation.sphinx.output.html.HtmlOutputConfiguration`  
`attribute)`, 45  
`_abc_impl` (`annize.features.documentation.sphinx.output.html.HtmlOutputGenerator`  
`attribute)`, 44  
`_abc_impl` (`annize.features.documentation.sphinx.output.pdf.PdfOutputGenerator`  
`attribute)`, 45  
`_abc_impl` (`annize.features.documentation.sphinx.output.plaintext.PlaintextOutputGenerator`  
`attribute)`, 45  
`_abc_impl` (`annize.features.documentation.sphinx.python.Python3ApiReferenceDocument`  
`attribute)`, 53  
`_abc_impl` (`annize.features.files.transfer.common.Endpoint`  
`attribute)`, 55  
`_abc_impl` (`annize.features.files.transfer.common.FsEndpoint`  
`attribute)`, 56  
`_abc_impl` (`annize.features.files.transfer.ssh.Endpoint`  
`attribute)`, 57  
`_abc_impl` (`annize.features.homepage.common.HomepageSection`  
`attribute)`, 64  
`_abc_impl` (`annize.features.homepage.sections.about.Section`  
`attribute)`, 61  
`_abc_impl` (`annize.features.homepage.sections.changelog.Section`  
`attribute)`, 61  
`_abc_impl` (`annize.features.homepage.sections.documentation.Section`  
`attribute)`, 61  
`_abc_impl` (`annize.features.homepage.sections.download.Section`  
`attribute)`, 62  
`_abc_impl` (`annize.features.homepage.sections.gallery.Section`  
`attribute)`, 62  
`_abc_impl` (`annize.features.homepage.sections.imprint.Section`  
`attribute)`, 64



- attribute*), 62
- `_abc_impl` (*annize.features.homepage.sections.license.Section* *attribute*), 63
- `_abc_impl` (*annize.features.i18n.common.ProjectDefinedTranslations* *attribute*), 66
- `_abc_impl` (*annize.features.i18n.common.String* *attribute*), 67
- `_abc_impl` (*annize.features.injections.common.FilesystemConnector* *attribute*), 68
- `_abc_impl` (*annize.features.injections.common.Injection* *attribute*), 68
- `_abc_impl` (*annize.features.injections.python.ProjectInfoInjection* *attribute*), 68
- `_abc_impl` (*annize.features.testing.common.Test* *attribute*), 69
- `_abc_impl` (*annize.features.testing.common.TestGroup* *attribute*), 69
- `_abc_impl` (*annize.features.testing.pylint.Test* *attribute*), 69
- `_abc_impl` (*annize.features.testing.pytest.Test* *attribute*), 69
- `_abc_impl` (*annize.features.testing.pyunit.Test* *attribute*), 70
- `_abc_impl` (*annize.features.version\_control.common.VersionControl* *attribute*), 70
- `_abc_impl` (*annize.features.version\_control.git.VersionControl* *attribute*), 71
- `_abc_impl` (*annize.flow.runner.Runner* *attribute*), 85
- `_abc_impl` (*annize.i18n.GettextTranslationProvider* *attribute*), 91
- `_abc_impl` (*annize.i18n.ProvidedTrStr* *attribute*), 92
- `_abc_impl` (*annize.i18n.TrStr* *attribute*), 92
- `_abc_impl` (*annize.i18n.TranslationProvider* *attribute*), 91
- `_abc_impl` (*annize.project.ArgumentNode* *attribute*), 104
- `_abc_impl` (*annize.project.BlockNode* *attribute*), 106
- `_abc_impl` (*annize.project.BlockNodeWithScope* *attribute*), 106
- `_abc_impl` (*annize.project.FileNode* *attribute*), 104
- `_abc_impl` (*annize.project.Node* *attribute*), 102
- `_abc_impl` (*annize.project.ObjectNode* *attribute*), 104
- `_abc_impl` (*annize.project.OnFeatureUnavailableNode* *attribute*), 106
- `_abc_impl` (*annize.project.ProjectNode* *attribute*), 103
- `_abc_impl` (*annize.project.ReferenceNode* *attribute*), 105
- `_abc_impl` (*annize.project.ScalarValueNode* *attribute*), 105
- `_abc_impl` (*annize.project.feature\_loader.DefaultFeatureLoader* *attribute*), 117
- `_abc_impl` (*annize.project.feature\_loader.FeatureLoader* *attribute*), 116
- `_abc_impl` (*annize.project.file\_formats.FileFormat* *attribute*), 107
- `_abc_impl` (*annize.project.file\_formats.FileFormat.Marshaler* *attribute*), 107
- `_abc_impl` (*annize.project.file\_formats.xml.Marshaler* *attribute*), 110
- `_abc_impl` (*annize.project.file\_formats.xml.XmlFileFormat* *attribute*), 108
- `_abc_impl` (*annize.project.materializer.behaviors.Behavior* *attribute*), 112
- `_abc_impl` (*annize.project.materializer.behaviors.argument.ArgumentBehavior* *attribute*), 112
- `_abc_impl` (*annize.project.materializer.behaviors.argument.AssociateArgument* *attribute*), 112
- `_abc_impl` (*annize.project.materializer.behaviors.basket.BasketBehavior* *attribute*), 113
- `_abc_impl` (*annize.project.materializer.behaviors.block.BlockBehavior* *attribute*), 113
- `_abc_impl` (*annize.project.materializer.behaviors.feature\_unavailable.FeatureUnavailableBehavior* *attribute*), 114
- `_abc_impl` (*annize.project.materializer.behaviors.reference.ReferenceBehavior* *attribute*), 114
- `_abc_impl` (*annize.user\_feedback.UserFeedbackController* *attribute*), 122
- `_abc_impl` (*annize.user\_feedback.static.StaticUserFeedbackController* *attribute*), 123
- `_abc_impl` (*annize.user\_feedback.tty.TtyUserFeedbackController* *attribute*), 124
- `_add_controller_to_context()` (in module *annize.user\_feedback*), 122
- `_allowed_child_types()` (*annize.project.BlockNode* class method), 105
- `_allowed_child_types()` (*annize.project.FileNode* class method), 103
- `_allowed_child_types()` (*annize.project.Node* class method), 101
- `_allowed_child_types()` (*annize.project.ObjectNode* class method), 104
- `_allowed_child_types()` (*annize.project.ProjectNode* class method), 103
- `_allowed_child_types()` (*annize.project.ReferenceNode* class method), 105
- `_allowed_child_types()` (*annize.project.ScalarValueNode* class method), 105
- `_append_section()` (*annize.features.homepage.common.Homepage* method), 65
- `_changed__child_added()` (*annize.project.Node* method), 100
- `_changed__child_removed()` (*annize.project.Node* method), 100
- `_changed__property_changed()` (*annize.project.Node* method), 100



<code>_controllers_for_context()</code> (in module <code>annize.user_feedback</code> ), 122	<code>_mkpackage_dpkg()</code> (an-
<code>_controllers_tuples_for_context()</code> (in module <code>annize.user_feedback</code> ), 122	<code>nize.features.distributables.debian.Package</code>
<code>_debian_category()</code> (in module <code>annize.features.distributables.debian</code> ), 24	<code>class method</code> ), 34
<code>_debian_section()</code> (in module <code>annize.features.distributables.debian</code> ), 27	<code>_mkpackage_flatpak_build_export()</code> (an-
<code>_description_for_mediafile()</code> (an-	<code>nize.features.distributables.flatpak.FlatpakImage</code>
<code>nize.features.media_galleries.Gallery method</code> ), 77	<code>class method</code> ), 39
<code>_dispatch()</code> ( <code>annize.flow.runner.Runner method</code> ), 84	<code>_mkpackage_flatpak_build_finish()</code> (an-
<code>_generate_sources()</code> (an-	<code>nize.features.distributables.flatpak.FlatpakImage</code>
<code>nize.features.documentation.sphinx.common.AboutPage method</code> ), 50	<code>class method</code> ), 38
<code>_generate_sources()</code> (an-	<code>_mkpackage_flatpak_build_init()</code> (an-
<code>nize.features.documentation.sphinx.common.ApiReferencePage method</code> ), 48	<code>nize.features.distributables.flatpak.FlatpakImage</code>
<code>_generate_sources()</code> (an-	<code>class method</code> ), 38
<code>nize.features.documentation.sphinx.common.ArgparsePage method</code> ), 49	<code>_mkpackage_mksources()</code> (an-
<code>_generate_sources()</code> (an-	<code>nize.features.distributables.debian.Package</code>
<code>nize.features.documentation.sphinx.common.ComponentPage method</code> ), 47	<code>class method</code> ), 33
<code>_generate_sources()</code> (an-	<code>_mkpackage_mkcopyright()</code> (an-
<code>nize.features.documentation.sphinx.common.DocumentPage method</code> ), 47	<code>nize.features.distributables.debian.Package</code>
<code>_generate_sources()</code> (an-	<code>class method</code> ), 33
<code>nize.features.documentation.sphinx.common.DocumentPage method</code> ), 47	<code>_mkpackage_mkdebiancontrolfile()</code> (an-
<code>_generate_sources()</code> (an-	<code>nize.features.distributables.debian.Package</code>
<code>nize.features.documentation.sphinx.common.DocumentPage method</code> ), 47	<code>class method</code> ), 34
<code>_generate_sources()</code> (an-	<code>_mkpackage_mkexecclinks()</code> (an-
<code>nize.features.documentation.sphinx.common.DocumentPage method</code> ), 47	<code>nize.features.distributables.debian.Package</code>
<code>_generate_sources()</code> (an-	<code>class method</code> ), 33
<code>nize.features.documentation.sphinx.common.DocumentPage method</code> ), 47	<code>_mkpackage_mkexecclinks()</code> (an-
<code>_get_data()</code> (in module <code>annize.features.base</code> ), 74	<code>nize.features.distributables.python_wheel.Package</code>
<code>_get_type_info()</code> (in module <code>annize.object.parameter_info</code> ), 97	<code>class method</code> ), 42
<code>_license()</code> (in module <code>annize.features.licensing</code> ), 75	<code>_mkpackage_mkmanifestin()</code> (an-
<code>_materialize_hlp_childobjs()</code> (an-	<code>nize.features.distributables.python_wheel.Package</code>
<code>nize.project.materializer.core.ProjectMaterializer</code>	<code>class method</code> ), 42
<code>method</code> ), 115	<code>_mkpackage_mkmenuentries()</code> (an-
<code>_mkpackage()</code> ( <code>annize.features.distributables.debian.Package</code>	<code>nize.features.distributables.debian.Package</code>
<code>class method</code> ), 32	<code>class method</code> ), 33
<code>_mkpackage()</code> ( <code>annize.features.distributables.flatpak.FlatpakImage</code>	<code>_mkpackage_mkprepostcmds()</code> (an-
<code>class method</code> ), 39	<code>nize.features.distributables.debian.Package</code>
<code>_mkpackage()</code> ( <code>annize.features.distributables.python_wheel.Package</code>	<code>class method</code> ), 34
<code>class method</code> ), 41	<code>_mkpackage_mkservices()</code> (an-
<code>_mkpackage_applysource()</code> (an-	<code>nize.features.distributables.debian.Package</code>
<code>nize.features.distributables.flatpak.FlatpakImage</code>	<code>class method</code> ), 33
<code>class method</code> ), 38	<code>_mkpackage_mksetuppyconf()</code> (an-
<code>_mkpackage_bdist_wheel()</code> (an-	<code>nize.features.distributables.python_wheel.Package</code>
<code>nize.features.distributables.python_wheel.Package</code>	<code>class method</code> ), 42
<code>class method</code> ), 42	<code>_mkpackage_prepareinfos()</code> (an-
<code>_mkpackage_correctbuildsourcepermissions()</code> (an-	<code>nize.features.distributables.debian.Package</code>
<code>annize.features.distributables.debian.Package</code>	<code>class method</code> ), 33
<code>class method</code> ), 34	<code>_mkpackage_prepareinfos()</code> (an-
<code>_mkpackage_determinesize()</code> (an-	<code>nize.features.distributables.flatpak.FlatpakImage</code>
<code>nize.features.distributables.debian.Package</code>	<code>class method</code> ), 38
	<code>_mkpackage_prepareinfos()</code> (an-
	<code>nize.features.distributables.python_wheel.Package</code>
	<code>class method</code> ), 38

class method), 41  
 \_mkpackage\_setuppyconf\_classifiers() (annize.features.distributables.python\_wheel.Package class method), 42  
 \_mkpackage\_setuppyconf\_install\_requires() (annize.features.distributables.python\_wheel.Package class method), 41  
 \_mkpackage\_setuppyconf\_prepare() (annize.features.distributables.python\_wheel.Package class method), 41  
 \_mkpackage\_share() (annize.features.distributables.flatpak.FlatpakImage class method), 39  
 \_name\_anon() (annize.data.version.AbstractVersionPatternSegment method), 15  
 \_name\_anon() (annize.data.version.ConcatenatedVersionPatternSegment method), 17  
 \_name\_anon() (annize.data.version.NumericVersionPatternSegment method), 16  
 \_name\_anon() (annize.data.version.OptionalVersionPatternSegment method), 17  
 \_node\_clone\_link() (in module annize.project.materializer), 111  
 \_path() (annize.features.distributables.debian.Package method), 31  
 \_path() (annize.features.distributables.flatpak.FlatpakImage method), 37  
 \_path() (annize.features.distributables.flatpak.FlatpakrefFile method), 36  
 \_path() (annize.features.distributables.flatpak.GpgFile method), 37  
 \_path() (annize.features.distributables.python\_wheel.Package method), 40  
 \_path() (annize.features.distributables.tar.Package method), 43  
 \_path() (annize.features.documentation.common.GeneratedDocument method), 55  
 \_path() (annize.features.files.common.Directory method), 60  
 \_path() (annize.features.files.common.FsEntry method), 57  
 \_path() (annize.features.files.common.MachineRootDirectory method), 60  
 \_path() (annize.features.files.common.ProjectDirectory method), 60  
 \_path() (annize.features.homepage.common.GeneratedHomepage method), 66  
 \_path() (annize.fs.Path method), 86  
 \_path() (annize.fs.ext.DynamicFile method), 90  
 \_set\_entry\_path() (annize.features.documentation.common.DocumentGenerateResult method), 53  
 \_str\_helper() (annize.project.ArgumentNode method), 104  
 \_str\_helper() (annize.project.BlockNode method), 105  
 \_str\_helper() (annize.project.BlockNodeWithScope method), 106  
 \_str\_helper() (annize.project.FileNode method), 103  
 \_str\_helper() (annize.project.Node method), 102  
 \_str\_helper() (annize.project.ObjectNode method), 104  
 \_str\_helper() (annize.project.OnFeatureUnavailableNode method), 106  
 \_str\_helper() (annize.project.ProjectNode method), 103  
 \_str\_helper() (annize.project.ReferenceNode method), 105  
 \_str\_helper() (annize.project.ScalarValueNode method), 105  
 \_str\_helper() (annize.data.version.AbstractVersionPatternSegment method), 15  
 \_str\_helper() (annize.data.version.NumericVersionPatternSegment method), 16  
 \_str\_helper() (annize.features.documentation.sphinx.common.DocumentGenerateResult method), 47  
 \_translate\_from\_clone() (in module annize.project.materializer), 111

## A

AboutProjectDocument (class in annize.features.documentation.sphinx.common), 50  
 AbstractVersionPatternSegment (class in annize.data.version), 15  
 access\_filesystem() (annize.features.files.transfer.common.Endpoint method), 55  
 access\_filesystem() (annize.features.files.transfer.common.FsEndpoint method), 56  
 access\_filesystem() (annize.features.files.transfer.ssh.Endpoint method), 56  
 add\_answer() (annize.user\_feedback.static.StaticUserFeedbackController method), 123  
 add\_change() (annize.project.file\_formats.FileFormat.Marshaler method), 107  
 add\_change() (annize.project.file\_formats.xml.Marshaler method), 110  
 add\_change\_handler() (annize.project.Node method), 100  
 add\_element() (annize.project.file\_formats.xml.Marshaler method), 110  
 add\_element\_attr() (annize.project.file\_formats.xml.Marshaler method), 110  
 add\_element\_tree() (annize.project.file\_formats.xml.Marshaler method), 110

*method*), 110  
 add\_object() (*annize.flow.run\_context.RunContext*  
*method*), 80  
 add\_object() (*in module annize.flow.run\_context*), 82  
 add\_translation\_provider() (*in module an-*  
*nize.i18n*), 91  
 add\_translations() (*an-*  
*nize.features.i18n.common.ProjectDefinedTranslation*  
*module*), 66  
 additional\_info() (*annize.features.licensing.License*  
*method*), 75  
 AdministrationUtilitiesSection (*in module an-*  
*nize.features.distributables.debian*), 27  
 AFLv3 (*in module annize.features.licensing*), 75  
 AGPLv3 (*in module annize.features.licensing*), 75  
 all() (*annize.project.inspector.Inspector.ArgumentMatchings*  
*method*), 118  
 allows\_multiple\_args (*an-*  
*nize.object.parameter\_info.ListParameterInfo*  
*property*), 97  
 allows\_multiple\_args (*an-*  
*nize.object.parameter\_info.ParameterInfo*  
*property*), 96  
 allows\_multiple\_args (*an-*  
*nize.project.inspector.Inspector.ArgumentMatchings*  
*property*), 117  
 annize  
   module, 13  
 annize.annize\_cli  
   module, 124  
 annize.asset  
   module, 13  
 annize.asset.data  
   module, 13  
 annize.asset.project\_info  
   module, 13  
 annize.data  
   module, 13  
 annize.data.color  
   module, 13  
 annize.data.container  
   module, 14  
 annize.data.unique  
   module, 14  
 annize.data.version  
   module, 15  
 annize.features  
   module, 18  
 annize.features.authors  
   module, 72  
 annize.features.base  
   module, 72  
 annize.features.changelog  
   module, 18  
 annize.features.changelog.common  
   module, 18  
 annize.features.dependencies  
   module, 19  
 annize.features.dependencies.common  
   module, 19  
 annize.features.dependencies.python  
   module, 20  
 annize.features.distributables  
   module, 21  
 annize.features.distributables.common  
   module, 21  
 annize.features.distributables.debian  
   module, 23  
 annize.features.distributables.flatpak  
   module, 34  
 annize.features.distributables.python\_wheel  
   module, 39  
 annize.features.distributables.store  
   module, 21  
 annize.features.distributables.store.pypi  
   module, 21  
 annize.features.distributables.tar  
   module, 42  
 annize.features.documentation  
   module, 43  
 annize.features.documentation.common  
   module, 53  
 annize.features.documentation.sphinx  
   module, 43  
 annize.features.documentation.sphinx.\_utils  
   module, 46  
 annize.features.documentation.sphinx.common  
   module, 46  
 annize.features.documentation.sphinx.cpp  
   module, 50  
 annize.features.documentation.sphinx.doxygen\_compat  
   module, 50  
 annize.features.documentation.sphinx.javascript  
   module, 52  
 annize.features.documentation.sphinx.output  
   module, 43  
 annize.features.documentation.sphinx.output.common  
   module, 43  
 annize.features.documentation.sphinx.output.html  
   module, 44  
 annize.features.documentation.sphinx.output.pdf  
   module, 45  
 annize.features.documentation.sphinx.output.plaintext  
   module, 45  
 annize.features.documentation.sphinx.python  
   module, 52  
 annize.features.documentation.sphinx.rst  
   module, 53

annize.features.files	annize.features.testing.pytest
module, 55	module, 69
annize.features.files.common	annize.features.testing.pyunit
module, 57	module, 69
annize.features.files.transfer	annize.features.version
module, 55	module, 77
annize.features.files.transfer.common	annize.features.version_control
module, 55	module, 70
annize.features.files.transfer.ssh	annize.features.version_control.common
module, 56	module, 70
annize.features.homepage	annize.features.version_control.git
module, 60	module, 71
annize.features.homepage.common	annize.flow
module, 63	module, 78
annize.features.homepage.sections	annize.flow.run_context
module, 60	module, 78
annize.features.homepage.sections.about	annize.flow.runner
module, 60	module, 83
annize.features.homepage.sections.changelog	annize.fs
module, 61	module, 85
annize.features.homepage.sections.documentation	annize.fs.ext
module, 61	module, 90
annize.features.homepage.sections.download	annize.i18n
module, 61	module, 91
annize.features.homepage.sections.gallery	annize.object
module, 62	module, 94
annize.features.homepage.sections.imprint	annize.object.controller
module, 62	module, 95
annize.features.homepage.sections.license	annize.object.parameter_info
module, 63	module, 96
annize.features.i18n	annize.project
module, 66	module, 97
annize.features.i18n.common	annize.project.feature_loader
module, 66	module, 116
annize.features.i18n.gettext	annize.project.file_formats
module, 67	module, 107
annize.features.injections	annize.project.file_formats.xml
module, 67	module, 108
annize.features.injections.common	annize.project.inspector
module, 67	module, 117
annize.features.injections.python	annize.project.loader
module, 68	module, 119
annize.features.licensing	annize.project.materializer
module, 75	module, 110
annize.features.media_galleries	annize.project.materializer.behaviors
module, 76	module, 111
annize.features.task	annize.project.materializer.behaviors.argument
module, 77	module, 112
annize.features.testing	annize.project.materializer.behaviors.basket
module, 68	module, 112
annize.features.testing.common	annize.project.materializer.behaviors.block
module, 68	module, 113
annize.features.testing.pylint	annize.project.materializer.behaviors.feature_unavailable
module, 69	module, 113

annize.project.materializer.behaviors.reference  
     module, 114  
 annize.project.materializer.core  
     module, 114  
 annize.project.materializer.preprocessors  
     module, 116  
 annize.ui  
     module, 120  
 annize.ui.apps  
     module, 120  
 annize.user\_feedback  
     module, 121  
 annize.user\_feedback.static  
     module, 123  
 annize.user\_feedback.tty  
     module, 123  
 annize\_config\_rootpath (annize.project.Project  
     property), 98  
 annize\_user\_interaction\_culture() (in module  
     annize.i18n), 94  
 Apache2 (in module annize.features.licensing), 75  
 ApiReferenceDocument (class in an-  
     nize.features.documentation.sphinx.common),  
     48  
 ApiReferenceLanguage (class in an-  
     nize.features.documentation.sphinx.common),  
     48  
 ApiReferenceLanguage.ApiReferenceGenerateInfo  
     (class in an-  
     nize.features.documentation.sphinx.common),  
     48  
 app() (in module annize.ui), 120  
 append\_child() (annize.project.Node method), 101  
 append\_rst() (annize.features.homepage.common.Homepage  
     method), 64  
 append\_to (annize.project.ArgumentNode property),  
     104  
 ApplicationsAccessibilityCategory (in module  
     annize.features.distributables.debian), 24  
 ApplicationsAmateurradioCategory (in module an-  
     nize.features.distributables.debian), 24  
 ApplicationsDatamanagementCategory (in module  
     annize.features.distributables.debian), 24  
 ApplicationsEditorsCategory (in module an-  
     nize.features.distributables.debian), 24  
 ApplicationsEducationCategory (in module an-  
     nize.features.distributables.debian), 24  
 ApplicationsEmulatorsCategory (in module an-  
     nize.features.distributables.debian), 24  
 ApplicationsFilemanagementCategory (in module  
     annize.features.distributables.debian), 24  
 ApplicationsGraphicsCategory (in module an-  
     nize.features.distributables.debian), 24  
 ApplicationsMobiledevicesCategory (in module  
     annize.features.distributables.debian), 24  
 ApplicationsNetworkCategory (in module an-  
     nize.features.distributables.debian), 24  
 ApplicationsNetworkCommunicationCategory (in  
     module annize.features.distributables.debian),  
     24  
 ApplicationsNetworkFiletransferCategory (in  
     module annize.features.distributables.debian),  
     24  
 ApplicationsNetworkMonitoringCategory (in mod-  
     ule annize.features.distributables.debian), 24  
 ApplicationsNetworkWebbrowsingCategory (in  
     module annize.features.distributables.debian),  
     24  
 ApplicationsNetworkWebnewsCategory (in module  
     annize.features.distributables.debian), 24  
 ApplicationsOfficeCategory (in module an-  
     nize.features.distributables.debian), 24  
 ApplicationsProgrammingCategory (in module an-  
     nize.features.distributables.debian), 24  
 ApplicationsProjectmanagementCategory (in mod-  
     ule annize.features.distributables.debian), 25  
 ApplicationsScienceAstronomyCategory (in mod-  
     ule annize.features.distributables.debian), 25  
 ApplicationsScienceBiologyCategory (in module  
     annize.features.distributables.debian), 25  
 ApplicationsScienceCategory (in module an-  
     nize.features.distributables.debian), 25  
 ApplicationsScienceChemistryCategory (in mod-  
     ule annize.features.distributables.debian), 25  
 ApplicationsScienceDataanalysisCategory (in  
     module annize.features.distributables.debian),  
     25  
 ApplicationsScienceElectronicsCategory (in  
     module annize.features.distributables.debian),  
     25  
 ApplicationsScienceEngineeringCategory (in  
     module annize.features.distributables.debian),  
     25  
 ApplicationsScienceGeoscienceCategory (in mod-  
     ule annize.features.distributables.debian), 25  
 ApplicationsScienceMathematicsCategory (in  
     module annize.features.distributables.debian),  
     25  
 ApplicationsScienceMedicineCategory (in module  
     annize.features.distributables.debian), 25  
 ApplicationsSciencePhysicsCategory (in module  
     annize.features.distributables.debian), 25  
 ApplicationsScienceSocialCategory (in module  
     annize.features.distributables.debian), 25  
 ApplicationsShellsCategory (in module an-  
     nize.features.distributables.debian), 25  
 ApplicationsSoundsCategory (in module an-  
     nize.features.distributables.debian), 25



- ApplicationsSystemAdministrationCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsSystemCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsSystemHardwareCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsSystemLanguageenvironmentCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsSystemMonitoringCategory (in module *annize.features.distributables.debian*), 26
- ApplicationsSystemPackagemanagementCategory (in module *annize.features.distributables.debian*), 26
- ApplicationsSystemSecurityCategory (in module *annize.features.distributables.debian*), 26
- ApplicationsTerminalEmulatorsCategory (in module *annize.features.distributables.debian*), 26
- ApplicationsTextCategory (in module *annize.features.distributables.debian*), 26
- ApplicationsTvandradiocategory (in module *annize.features.distributables.debian*), 26
- ApplicationsVideoCategory (in module *annize.features.distributables.debian*), 26
- ApplicationsViewersCategory (in module *annize.features.distributables.debian*), 26
- ApplicationsWebdevelopmentCategory (in module *annize.features.distributables.debian*), 26
- architecture (*annize.features.distributables.debian.Package.\_BuildInfo* attribute), 32
- arg\_name (*annize.project.ArgumentNode* property), 104
- argname (*annize.project.inspector.Inspector.ArgumentMatchings.ArgumentMatching* property), 117
- ArgparseCommandLineInterfaceDocument (class in *annize.features.documentation.sphinx.common*), 48
- ArgumentBehavior (class in *annize.project.materializer.behaviors.argument*), 112
- ArgumentNode (class in *annize.project*), 104
- Artistic1 (in module *annize.features.licensing*), 75
- Artwork (class in *annize.features.dependencies.common*), 20
- AssociateArgumentNodeBehavior (class in *annize.project.materializer.behaviors.argument*), 112
- attach\_media\_file() (*annize.features.homepage.common.HomepageSection.Content* method), 64
- attr\_name (*annize.project.file\_formats.xml.Marshaler.XmlParser* attribute), 110
- ATTRIBUTE\_COMMAND\_END (in *annize.project.file\_formats.xml.\_XmlParser* attribute), 109
- ATTRIBUTE\_COMMAND\_START (in *annize.project.file\_formats.xml.\_XmlParser* attribute), 108
- author (*annize.features.distributables.debian.Package.\_BuildInfo* attribute), 32
- author (*annize.features.distributables.python\_wheel.Package.\_BuildInfo* attribute), 41
- Author (class in *annize.features.authors*), 72
- authors (*annize.features.documentation.sphinx.common.Document* property), 47
- authorstring (*annize.features.distributables.debian.Package.\_BuildInfo* attribute), 32
- available\_cultures() (*annize.features.documentation.common.Document* method), 54
- available\_cultures() (*annize.features.documentation.sphinx.common.AboutProjectDocument* method), 50
- available\_cultures() (*annize.features.documentation.sphinx.common.ApiReferenceDocument* method), 48
- available\_cultures() (*annize.features.documentation.sphinx.common.ArgparseCommandLineInterfaceDocument* method), 49
- available\_cultures() (*annize.features.documentation.sphinx.common.ReadmeDocument* method), 50
- available\_cultures() (*annize.features.documentation.sphinx.common.RstDocument* method), 49
- ## B
- background\_image (in *annize.features.documentation.sphinx.output.html.HtmlOutputSpec* property), 44
- BadStructureError, 106
- Basket (class in *annize.data.container*), 14
- Basket (class in *annize.features.base*), 73
- BasketBehavior (class in *annize.project.materializer.behaviors.basket*), 112
- Behavior (class in *annize.project.materializer.behaviors*), 111
- BLOCK (*annize.project.BlockNodeWithScope.Scope* attribute), 106
- BlockBehavior (class in *annize.project.materializer.behaviors.block*), 113
- BlockNode (class in *annize.project*), 105
- BlockNodeWithScope (class in *annize.project*), 106
- BlockNodeWithScope.Scope (class in *annize.project*), 106

blue (*annize.data.color.Color* property), 13  
 brand\_color() (in module *annize.features.base*), 74  
 BrandColor (class in *annize.features.base*), 73  
 BSD2clause (in module *annize.features.licensing*), 75  
 BSD3clause (in module *annize.features.licensing*), 75  
 BuildVersion (class in *annize.features.version\_control.common*), 70  
 ByVersionControlSystemCommitMessagesChangelog (class in *annize.features.changelog.common*), 18  
**C**  
 category (*annize.features.distributables.debian.MenuEntry* property), 23  
 category (*annize.features.distributables.flatpak.MenuEntry* property), 35  
 Category (class in *annize.features.distributables.debian*), 23  
 Cc0v1 (in module *annize.features.licensing*), 75  
 CcBy3 (in module *annize.features.licensing*), 75  
 CcByNc3 (in module *annize.features.licensing*), 75  
 CcByNcNd3 (in module *annize.features.licensing*), 75  
 CcByNcSa3 (in module *annize.features.licensing*), 76  
 CcByNd3 (in module *annize.features.licensing*), 76  
 CcBySa3 (in module *annize.features.licensing*), 76  
 Changelog (class in *annize.features.changelog.common*), 18  
 child\_node (*annize.project.Node.ChildrenListChangeEvent* property), 99  
 child\_position (*annize.project.Node.ChildrenListChangeEvent* property), 99  
 children (*annize.project.Node* property), 101  
 children() (*annize.fs.Path* method), 86  
 ChildrenNotMaterializableError, 116  
 choice\_dialog() (*annize.user\_feedback.NullUserFeedbackController* method), 122  
 choice\_dialog() (*annize.user\_feedback.static.StaticUserFeedbackController* method), 123  
 choice\_dialog() (*annize.user\_feedback.tty.TtyUserFeedbackController* method), 124  
 choice\_dialog() (*annize.user\_feedback.UserFeedbackController* method), 121  
 choice\_dialog() (in module *annize.user\_feedback*), 123  
 clone() (*annize.project.Node* method), 101  
 Color (class in *annize.data.color*), 13  
 command (*annize.features.distributables.debian.MenuEntry* property), 23  
 command (*annize.features.distributables.flatpak.FlatpakImage.\_BuildInfo* property), 65  
 command (*annize.features.distributables.flatpak.MenuEntry* property), 35  
 Commands (class in *annize.annize\_cli*), 125  
 Commands.ConsoleRunner (class in *annize.annize\_cli*), 125  
 comment (*annize.features.dependencies.common.Dependency* property), 19  
 CommonVersionPattern (class in *annize.features.version*), 78  
 CommunicationProgramsSection (in module *annize.features.distributables.debian*), 27  
 CompositeDocument (class in *annize.features.documentation.sphinx.common*), 47  
 ConcatenatedVersionPatternSegment (class in *annize.data.version*), 17  
 confdir (*annize.features.documentation.sphinx.common.Document.Generated* attribute), 46  
 config\_files (*annize.features.distributables.debian.Package.\_BuildInfo* attribute), 32  
 configlines (*annize.features.documentation.sphinx.common.Document.Generated* attribute), 47  
 configvalues (*annize.features.documentation.sphinx.common.Document.Generated* attribute), 47  
 Connection (class in *annize.features.distributables.store.pypi*), 21  
 construct\_from\_string() (*annize.object.parameter\_info.ParameterInfo* method), 97  
 Content (*annize.features.injections.common.FilesystemContentInjection* property), 68  
 content() (in module *annize.fs*), 89  
 copy\_to() (*annize.fs.Path* method), 87  
 CppApiReferenceLanguage (class in *annize.features.documentation.sphinx.cpp*), 50  
 create\_new() (*annize.project.Project* static method), 98  
 create\_object() (*annize.object.controller.\_CreateObjectHelper* static method), 95  
 create\_object() (in module *annize.object.controller*), 95  
 ctime() (*annize.fs.Path* method), 86  
 culture (*annize.features.documentation.sphinx.common.Document.Generated* attribute), 46  
 culture (*annize.features.documentation.sphinx.common.RstDocumentVariant* property), 49  
 culture (*annize.features.homepage.common.HomepageSection.\_Generated* attribute), 63  
 Culture (class in *annize.features.i18n.common*), 67  
 Culture (class in *annize.i18n*), 93  
 cultures (*annize.features.homepage.common.HomepageSection* property), 65  
 current() (in module *annize.flow.run\_context*), 81

current\_culture() (in module *annize.i18n*), 94  
 custom\_arg(*annize.features.homepage.common.HomepageSection.\_GenerateInfo* attribute), 63  
 custom\_arg(*annize.features.homepage.common.HomepageSection.\_PrePostProcGenerateInfo* attribute), 64

## D

Data (class in *annize.features.base*), 72  
 DatabasesSection (in module *annize.features.distributables.debian*), 27  
 DateTime (class in *annize.features.base*), 73  
 debian\_name(*annize.features.distributables.debian.Category* property), 23  
 DebianInstallerUdebPackagesSection (in module *annize.features.distributables.debian*), 27  
 DebugPackagesSection (in module *annize.features.distributables.debian*), 27  
 default\_changelog() (in module *annize.features.changelog.common*), 19  
 default\_version\_control\_system() (in module *annize.features.version\_control.common*), 71  
 default\_version\_pattern() (in module *annize.features.version*), 77  
 DefaultFeatureLoader (class in *annize.project.feature\_loader*), 116  
 dependencies(*annize.features.distributables.python\_wheel.Package.\_BuildInfo* attribute), 41  
 dependencies\_to\_rst\_text() (in module *annize.features.dependencies.common*), 20  
 Dependency (class in *annize.features.dependencies.common*), 19  
 description(*annize.features.distributables.common.Group* property), 22  
 description(*annize.features.distributables.debian.Package.\_BuildInfo* attribute), 32  
 description(*annize.features.distributables.flatpak.Group* property), 36  
 description(*annize.features.distributables.python\_wheel.Package.\_BuildInfo* attribute), 41  
 description(*annize.features.media\_galleries.Gallery.Item* property), 77  
 description() (*annize.project.Node* method), 102  
 destination(*annize.fs.ext.Mount* property), 91  
 destination\_is\_parent (in *annize.features.files.common.DirectoryPart* property), 59  
 destination\_path (in *annize.features.files.common.DirectoryPart* property), 59  
 DevelopmentSection (in module *annize.features.distributables.debian*), 27  
 Directory (class in *annize.features.files.common*), 59  
 DirectoryPart (class in *annize.features.files.common*), 58  
 do (*annize.project.OnFeatureUnavailableNode* property), 66  
 do() (*annize.annize\_cli.Commands* method), 126  
 Document (class in *annize.features.documentation.common*), 54  
 Document (class in *annize.features.documentation.sphinx.common*), 46  
 Document.GenerateInfo (class in *annize.features.documentation.sphinx.common*), 46  
 document\_root\_directory (in *annize.features.homepage.common.HomepageSection.\_PrePostProcGenerateInfo* attribute), 64  
 document\_root\_url (in *annize.features.homepage.common.HomepageSection.\_GenerateInfo* attribute), 63  
 document\_root\_url (in *annize.features.homepage.common.HomepageSection.\_PrePostProcGenerateInfo* attribute), 64  
 document\_variant\_directory (in *annize.features.homepage.common.HomepageSection.\_GenerateInfo* attribute), 63  
 document\_variant\_url (in *annize.features.homepage.common.HomepageSection.\_GenerateInfo* attribute), 63  
 documentation\_source (in *annize.features.distributables.debian.Package.\_BuildInfo* attribute), 32  
 DocumentationSection (in module *annize.features.distributables.debian*), 27  
 DocumentGenerateAllCulturesResult (class in *annize.features.documentation.common*), 53  
 DocumentGenerateResult (class in *annize.features.documentation.common*), 53  
 does\_exclude() (*annize.features.files.common.Exclude* method), 58  
 does\_exclude() (*annize.features.files.common.ExcludeAllBut* method), 58  
 does\_exclude() (*annize.features.version\_control.git.ExcludeByGitIgnore* method), 71  
 DoxygenSupportedApiReferenceLanguage (class in *annize.features.documentation.sphinx.doxygen\_compat*), 50  
 dynamic\_file() (in module *annize.fs*), 89  
 DynamicFile (class in *annize.fs.ext*), 90

## E

EditorsSection (in module *annize.features.distributables.debian*), 27  
 EducationSection (in module *annize.features.distributables.debian*), 27



ElectronicsSection (in module *annize.features.distributables.debian*), 27  
 element (*annize.project.file\_formats.xml.Marshaler.XmlDocumentLoader* attribute), 110  
 email (*annize.features.authors.Author* property), 72  
 EmbeddedSoftwareSection (in module *annize.features.distributables.debian*), 27  
 Endpoint (class in *annize.features.files.transfer.common*), 55  
 Endpoint (class in *annize.features.files.transfer.ssh*), 56  
 english\_lang\_name (*annize.i18n.Culture* property), 93  
 entries (*annize.features.changelog.common.ByVersionControlSystemCommitMessagesChangelog* property), 19  
 entries (*annize.features.changelog.common.Changelog* property), 18  
 Entry (class in *annize.features.changelog.common*), 18  
 entry\_path (*annize.features.documentation.common.DocumentGenerateResult* property), 53  
 entry\_path (*annize.features.documentation.sphinx.common.DocumentGenerateResult* attribute), 47  
 entry\_path\_for\_language() (*annize.features.documentation.common.DocumentGenerateResult* method), 54  
 environment (*annize.features.distributables.flatpak.FlatpakImage.\_BuildInfo* attribute), 38  
 EnvironmentVariable (class in *annize.features.distributables.flatpak*), 35  
 erroneous\_nodes() (*annize.project.materializer.MaterializationResult* method), 111  
 errors\_for\_node() (*annize.project.materializer.MaterializationResult* method), 111  
 escape\_attribute\_string() (*annize.project.file\_formats.xml.\_XmlParser* class method), 109  
 Exclude (class in *annize.features.files.common*), 57  
 ExcludeAllBut (class in *annize.features.files.common*), 58  
 ExcludeByGitIgnores (class in *annize.features.version\_control.git*), 71  
 excludes (*annize.features.files.common.Directory* property), 60  
 excludes (*annize.features.files.common.DirectoryPart* property), 59  
 exec() (*annize.features.files.transfer.ssh.Endpoint* method), 57  
 executable\_links (*annize.features.distributables.debian.Package.\_BuildInfo* attribute), 32  
 executable\_links (*annize.features.distributables.python\_wheel.Package.\_BuildInfo* attribute), 41  
 ExecutableLink (class in *annize.features.distributables.debian*), 23  
 ExecutableLink (class in *annize.features.distributables.python\_wheel*), 39  
 explicit\_only (*annize.object.controller.\_CreateObjectHelper.Parameter* attribute), 95  
 explicit\_only() (in module *annize.object*), 94  
**F**  
 FAIL (*annize.project.OnFeatureUnavailableNode.Action* attribute), 106  
 fail (*annize.project.ReferenceNode.OnUnresolvableAction* attribute), 105  
 fallback\_cultures (*annize.i18n.Culture* property), 93  
 feature (*annize.project.inspector.Inspector.TypeInfo* property), 118  
 feature (*annize.project.ObjectNode* property), 104  
 feature (*annize.project.OnFeatureUnavailableNode* property), 106  
 FeatureLoader (class in *annize.project.feature\_loader*), 116  
 FeatureUnavailableBehavior (class in *annize.project.materializer.behaviors.feature\_unavailable*), 116  
 FeatureUnavailableError, 106  
 file (*annize.features.documentation.common.DocumentGenerateResult* property), 53  
 file (*annize.features.media\_galleries.Gallery.Item* property), 77  
 FILE (*annize.project.BlockNodeWithScope.Scope* attribute), 106  
 File (class in *annize.features.files.common*), 57  
 file\_size() (*annize.fs.Path* method), 86  
 FileFormat (class in *annize.project.file\_formats*), 107  
 FileFormat.Marshaler (class in *annize.project.file\_formats*), 107  
 filehash() (in module *annize.features.homepage.sections.download*), 62  
 FileNode (class in *annize.project*), 103  
 files() (*annize.features.distributables.common.Group* method), 22  
 files() (*annize.features.distributables.flatpak.Group* method), 36  
 Filesystem (class in *annize.features.distributables.flatpak*), 35  
 FilesystemContent (class in *annize.fs*), 85  
 FilesystemContentInjection (class in *annize.features.injections.common*), 68  
 filesystems (*annize.features.distributables.flatpak.FlatpakImage.\_BuildInfo* attribute), 38  
 find\_output\_generator\_for\_outputspec() (in module *annize.features.documentation.sphinx.output.common*), 116

- 43  
 find\_project\_annize\_config\_root\_file() (in module *annize.project.loader*), 119  
 FirstOf (class in *annize.features.base*), 73  
 FlatpakImage (class in *annize.features.distributables.flatpak*), 37  
 FlatpakImage.\_BuildInfo (class in *annize.features.distributables.flatpak*), 37  
 FlatpakrefFile (class in *annize.features.distributables.flatpak*), 36  
 FontsSection (in module *annize.features.distributables.debian*), 28  
 format() (*annize.i18n.TrStr* method), 92  
 formatname() (*annize.features.documentation.sphinx.output.common.OutputGenerator* method), 43  
 formatname() (*annize.features.documentation.sphinx.output.html.HtmlOutputGenerator* method), 45  
 formatname() (*annize.features.documentation.sphinx.output.pdf.PdfOutputGenerator* method), 45  
 formatname() (*annize.features.documentation.sphinx.output.plaintext.PlaintextOutputGenerator* method), 45  
 freedesktop\_name (an-*annize.features.distributables.debian.Category* property), 23  
 fresh\_temp\_directory() (in module *annize.fs*), 89  
 FreshTempDirectory (class in *annize.fs.ext*), 90  
 friendly\_filesize() (in module *annize.features.homepage.sections.download*), 62  
 friendly\_join\_string\_list() (in module *annize.i18n*), 94  
 friendly\_name\_suggestion (an-*annize.features.distributables.flatpak.Repository* property), 35  
 FromRequirementsFile (class in *annize.features.dependencies.python*), 20  
 FsEndpoint (class in *annize.features.files.transfer.common*), 55  
 FsEntry (class in *annize.features.files.common*), 57  
 fullname (*annize.features.authors.Author* property), 72  
 fullname (*annize.i18n.Culture* property), 93
- ## G
- Gallery (class in *annize.features.media\_galleries*), 76  
 Gallery.Item (class in *annize.features.media\_galleries*), 76  
 GamesActionCategory (in module *annize.features.distributables.debian*), 26  
 GamesAdventureCategory (in module *annize.features.distributables.debian*), 26  
 GamesBlocksCategory (in module *annize.features.distributables.debian*), 26  
 GamesBoardCategory (in module *annize.features.distributables.debian*), 26  
 GamesCardCategory (in module *annize.features.distributables.debian*), 26  
 GamesPuzzlesCategory (in module *annize.features.distributables.debian*), 26  
 GamesSection (in module *annize.features.distributables.debian*), 28  
 GamesSimulationCategory (in module *annize.features.distributables.debian*), 26  
 GamesStrategyCategory (in module *annize.features.distributables.debian*), 26  
 GamesToolsCategory (in module *annize.features.distributables.debian*), 26  
 GamesToysCategory (in module *annize.features.distributables.debian*), 26  
 generate() (*annize.features.documentation.common.Document* method), 43  
 generate() (*annize.features.documentation.sphinx.common.Document* method), 45  
 generate() (*annize.features.homepage.common.Homepage* method), 47  
 generate\_all\_cultures() (an-*annize.features.documentation.common.Document* method), 54  
 generate\_all\_cultures() (an-*annize.features.documentation.sphinx.common.Document* method), 47  
 generate\_content() (an-*annize.features.homepage.common.HomepageSection* method), 64  
 generate\_content() (an-*annize.features.homepage.sections.about.Section* method), 60  
 generate\_content() (an-*annize.features.homepage.sections.changelog.Section* method), 61  
 generate\_content() (an-*annize.features.homepage.sections.documentation.Section* method), 61  
 generate\_content() (an-*annize.features.homepage.sections.download.Section* method), 62  
 generate\_content() (an-*annize.features.homepage.sections.gallery.Section* method), 62  
 generate\_content() (an-*annize.features.homepage.sections.imprint.Section* method), 62  
 generate\_content() (an-*annize.features.homepage.sections.license.Section* method), 63  
 generate\_sources() (an-*annize.features.documentation.sphinx.common.ApiReferenceLanguage* method), 48  
 generate\_sources() (an-

`nize.features.documentation.sphinx.doxygen_compat.DoxygenSupportLanguage`  
`method`), 52  
`get_node_by_name()` (an-  
`nize.project.inspector.Inspector` `method`),  
`nize.features.documentation.sphinx.javascript.JavaScriptApiReferenceLanguage`  
`method`), 52  
`get_project_node()` (an-  
`nize.project.inspector.Inspector` `method`),  
`nize.features.documentation.sphinx.python.Python3ApiReferenceLanguage`  
`method`), 53  
`get_revision_list()` (an-  
`nize.features.version_control.common.VersionControlSystem`  
`method`), 70  
`GeneratedDocument` (class in an-  
`nize.features.documentation.common`), 55  
`GeneratedHomepage` (class in an-  
`nize.features.homepage.common`), 66  
`GenerateMOs` (class in `annize.features.i18n.gettext`), 67  
`get_all_available_feature_names()` (an-  
`nize.project.feature_loader.DefaultFeatureLoader`  
`method`), 117  
`get_all_available_feature_names()` (an-  
`nize.project.feature_loader.FeatureLoader`  
`method`), 116  
`get_all_types()` (`annize.project.inspector.Inspector`  
`method`), 118  
`get_changes()` (`annize.project.ProjectNode` `method`),  
102  
`get_commit_message()` (an-  
`nize.features.version_control.common.VersionControlSystem`  
`method`), 70  
`get_commit_message()` (an-  
`nize.features.version_control.git.VersionControlSystem`  
`method`), 71  
`get_commit_time()` (an-  
`nize.features.version_control.common.VersionControlSystem`  
`method`), 70  
`get_commit_time()` (an-  
`nize.features.version_control.git.VersionControlSystem`  
`method`), 71  
`get_culture()` (in module `annize.i18n`), 94  
`get_current_revision()` (an-  
`nize.features.version_control.common.VersionControlSystem`  
`method`), 70  
`get_current_revision()` (an-  
`nize.features.version_control.git.VersionControlSystem`  
`method`), 71  
`get_format()` (in module `annize.project.file_formats`),  
108  
`get_from_iso_639_1_lang_code()` (an-  
`nize.i18n.Culture` `static method`), 93  
`get_materialized()` (an-  
`nize.project.materializer.core.ProjectMaterializerGroup`  
`method`), 115  
`get_materialized_children()` (an-  
`nize.project.materializer.core.NodeMaterialization`  
`method`), 114  
`get_materialized_children_tuples()` (an-  
`nize.project.materializer.core.NodeMaterialization`  
`method`), 115  
`get_node_by_name()` (an-  
`nize.project.inspector.Inspector` `method`),  
`get_project_node()` (an-  
`nize.project.inspector.Inspector` `method`),  
`get_revision_list()` (an-  
`nize.features.version_control.common.VersionControlSystem`  
`method`), 70  
`get_revision_list()` (an-  
`nize.features.version_control.git.VersionControlSystem`  
`method`), 71  
`get_revision_number()` (an-  
`nize.features.version_control.common.VersionControlSystem`  
`method`), 70  
`get_revision_number()` (an-  
`nize.features.version_control.git.VersionControlSystem`  
`method`), 71  
`get_selected_task()` (`annize.flow.runner.Runner`  
`method`), 84  
`get_success_state()` (`annize.flow.runner.Runner`  
`method`), 84  
`get_tasks()` (`annize.flow.runner.Runner` `method`), 84  
`get_types_for_argument()` (an-  
`nize.project.inspector.Inspector` `method`),  
118  
`get_variant()` (`annize.i18n.ProvidedTrStr` `method`), 92  
`get_variant()` (`annize.i18n.TrStr` `method`), 92  
`GettextTranslationProvider` (class in `annize.i18n`),  
91  
`GnomeSection` (in module an-  
`nize.features.distributables.debian`), 28  
`GnuLinux` (class in an-  
`nize.features.dependencies.common`), 20  
`GnuRSection` (in module an-  
`nize.features.distributables.debian`), 28  
`GnuStepsSection` (in module an-  
`nize.features.distributables.debian`), 28  
`GpgFile` (class in `annize.features.distributables.flatpak`),  
36  
`GPLv3` (in module `annize.features.licensing`), 76  
`GraphicsSection` (in module an-  
`nize.features.distributables.debian`), 28  
`green` (`annize.data.color.Color` `property`), 13  
`Group` (class in `annize.features.distributables.common`),  
22  
`Group` (class in `annize.features.distributables.flatpak`), 35  

## H

`HamRadioSection` (in module an-  
`nize.features.distributables.debian`), 28  
`has_result` (`annize.project.materializer.core.NodeMaterialization`  
`property`), 115

[has\\_shell\\_access](#) ([annize.features.files.transfer.ssh.Endpoint](#) property), 56  
[HaskellSection](#) (in module [annize.features.distributables.debian](#)), 28  
[head](#) ([annize.features.homepage.common.HomepageSection](#) property), 64  
[heading](#) ([annize.features.documentation.sphinx.common.ApiReferenceElement](#) property), 48  
[heading\(\)](#) ([annize.features.documentation.sphinx.rst.RstGenerator](#) static method), 53  
[HelpCategory](#) (in module [annize.features.distributables.debian](#)), 27  
[homepage](#) ([annize.features.distributables.debian.Package](#) attribute), 32  
[homepage](#) ([annize.features.distributables.python\\_wheel.Package](#) attribute), 41  
[Homepage](#) (class in [annize.features.homepage.common](#)), 64  
[homepage\\_url](#) ([annize.features.base.Data](#) property), 73  
[homepage\\_url\(\)](#) (in module [annize.features.base](#)), 74  
[HomepageSection](#) (class in [annize.features.homepage.common](#)), 63  
[HomepageSection.\\_GenerateInfo](#) (class in [annize.features.homepage.common](#)), 63  
[HomepageSection.\\_PrePostProcGenerateInfo](#) (class in [annize.features.homepage.common](#)), 63  
[HomepageSection.Content](#) (class in [annize.features.homepage.common](#)), 64  
[host](#) ([annize.features.files.transfer.ssh.Endpoint](#) property), 56  
[html\\_color\\_spec](#) ([annize.data.color.Color](#) property), 14  
[HtmlOutputGenerator](#) (class in [annize.features.documentation.sphinx.output.html](#)), 44  
[HtmlOutputSpec](#) (class in [annize.features.documentation.common](#)), 54  
[HtmlOutputSpec](#) (class in [annize.features.documentation.sphinx.output.html](#)), 44  
[hue](#) ([annize.data.color.Color](#) property), 13  
  
[|](#)  
[icon](#) ([annize.features.dependencies.common.Dependency](#) property), 19  
[icon](#) ([annize.features.distributables.debian.MenuEntry](#) property), 23  
[icon](#) ([annize.features.distributables.flatpak.MenuEntry](#) property), 35  
[IdCulture](#) (class in [annize.features.i18n.common](#)), 67  
[IdCulture](#) (class in [annize.i18n](#)), 93  
  
[identity\\_file](#) ([annize.features.files.transfer.ssh.Endpoint](#) property), 56  
[IMAGE](#) ([annize.features.media\\_galleries.MediaType](#) attribute), 76  
[importance](#) ([annize.features.dependencies.common.Dependency](#) property), 19  
[importance](#) ([annize.features.dependencies.common.Kind](#) property), 19  
[imprint](#) ([annize.features.base.Data](#) property), 73  
[imprint\(\)](#) (in module [annize.features.base](#)), 74  
[in\\_file\(\)](#) ([annize.project.file\\_formats.xml.XmlParser.\\_Context](#) method), 108  
[in\\_node\(\)](#) ([annize.project.file\\_formats.xml.XmlParser.\\_Context](#) method), 108  
[Included](#) (class in [annize.features.dependencies.common](#)), 20  
[Inject](#) (class in [annize.features.injections.common](#)), 68  
[inject\(\)](#) ([annize.features.injections.common.FilesystemContentInjection](#) method), 68  
[inject\(\)](#) ([annize.features.injections.common.Injection](#) method), 67  
[inject\(\)](#) ([annize.features.injections.python.ProjectInfoInjection](#) method), 68  
[Injection](#) (class in [annize.features.injections.common](#)), 67  
[inner\\_type\\_info](#) ([annize.object.parameter\\_info.ListParameterInfo](#) property), 97  
[inner\\_type\\_info](#) ([annize.object.parameter\\_info.ParameterInfo](#) property), 96  
[input\\_dialog\(\)](#) ([annize.user\\_feedback.NullUserFeedbackController](#) method), 122  
[input\\_dialog\(\)](#) ([annize.user\\_feedback.static.StaticUserFeedbackController](#) method), 123  
[input\\_dialog\(\)](#) ([annize.user\\_feedback.tty.TtyUserFeedbackController](#) method), 124  
[input\\_dialog\(\)](#) ([annize.user\\_feedback.UserFeedbackController](#) method), 121  
[input\\_dialog\(\)](#) (in module [annize.user\\_feedback](#)), 122  
[insert\\_child\(\)](#) ([annize.project.Node](#) method), 101  
[insert\\_child\(\)](#) ([annize.project.ProjectNode](#) method), 102  
  
[Inspector](#) (class in [annize.project.inspector](#)), 117  
[Inspector.ArgumentMatchings](#) (class in [annize.project.inspector](#)), 117  
[Inspector.ArgumentMatchings.ArgumentMatching](#) (class in [annize.project.inspector](#)), 117  
[Inspector.TypeInfo](#) (class in [annize.project.inspector](#)), 118  
[InternalError](#), 115  
[InterpretersSection](#) (in module [annize.features.distributables.debian](#)), 28  
[IntrospectionSection](#) (in module [annize.features.distributables.debian](#)), 28



[nize.features.distributables.debian](#)), 28  
[intstruct](#) ([annize.features.documentation.sphinx.common.DocumentGenerateInfo](#) attribute), 46  
[is\\_advanced](#) ([annize.features.task.Task](#) property), 77  
[is\\_compatible\\_for\(\)](#) ([annize.features.documentation.sphinx.output.common.OutputGenerator](#) class method), 43  
[is\\_compatible\\_for\(\)](#) ([annize.features.documentation.sphinx.output.html.HtmlOutputGenerator](#) class method), 45  
[is\\_compatible\\_for\(\)](#) ([annize.features.documentation.sphinx.output.pdf.PdfOutputGenerator](#) class method), 45  
[is\\_compatible\\_for\(\)](#) ([annize.features.documentation.sphinx.output.plaintext.PlaintextOutputGenerator](#) class method), 45  
[is\\_constructable\\_from\\_string](#) ([annize.object.parameter\\_info.ParameterInfo](#) property), 97  
[is\\_finished\(\)](#) ([annize.flow.runner.Runner](#) method), 84  
[is\\_friendly\\_name\(\)](#) ([annize.flow.run\\_context.RunContext](#) method), 80  
[is\\_friendly\\_name\(\)](#) (in module [annize.flow.run\\_context](#)), 83  
[is\\_gui](#) ([annize.features.distributables.debian.MenuEntry](#) property), 23  
[is\\_gui](#) ([annize.features.distributables.flatpak.MenuEntry](#) property), 35  
[is\\_gui](#) ([annize.features.distributables.python\\_wheel.ExecutableLink](#) property), 39  
[is\\_homepage](#) ([annize.features.documentation.common.HtmlOutputSpec](#) property), 54  
[is\\_optional](#) ([annize.object.parameter\\_info.ParameterInfo](#) property), 96  
[is\\_toplevel\\_object\(\)](#) ([annize.flow.run\\_context.RunContext](#) method), 80  
[is\\_toplevel\\_object\(\)](#) (in module [annize.flow.run\\_context](#)), 83  
[iso\\_639\\_1\\_lang\\_code](#) ([annize.i18n.Culture](#) property), 93  
[Item](#) (class in [annize.features.changelog.common](#)), 18  
[items](#) ([annize.features.changelog.common.Entry](#) property), 18  
[items](#) ([annize.features.media\\_galleries.Gallery](#) property), 77  
**J**  
[JavaScriptApiReferenceLanguage](#) (class in [annize.features.documentation.sphinx.javascript](#)), 52  
[JavascriptSection](#) (in module [annize.features.distributables.debian](#)), 28  
[JavaSection](#) (in module [annize.features.distributables.debian](#)), 28  
[join\\_authors\(\)](#) (in module [annize.features.authors](#)), 72  
**K**  
[KernelSection](#) (in module [annize.features.distributables.debian](#)), 28  
[Keyword](#) (class in [annize.features.base](#)), 73  
[keywords](#) ([annize.features.base.Keywords](#) property), 73  
[keywords](#) ([annize.features.distributables.python\\_wheel.Package.\\_BuildInfo](#) attribute), 41  
[Keywords](#) (class in [annize.features.base](#)), 73  
[Kind](#) (class in [annize.features.dependencies.common](#)), 19  
[kitversion](#) ([annize.features.distributables.flatpak.FlatpakImage.\\_BuildInfo](#) attribute), 38  
**L**  
[label](#) ([annize.features.dependencies.common.Dependency](#) property), 19  
[label](#) ([annize.features.dependencies.common.Kind](#) property), 19  
[LanguagePacksSection](#) (in module [annize.features.distributables.debian](#)), 28  
[languages](#) ([annize.features.documentation.common.DocumentGenerateAll](#) property), 54  
[LGPLV3](#) (in module [annize.features.licensing](#)), 76  
[LibrariesSection](#) (in module [annize.features.distributables.debian](#)), 28  
[LibraryDevelopmentSection](#) (in module [annize.features.distributables.debian](#)), 28  
[license](#) ([annize.features.distributables.python\\_wheel.Package.\\_BuildInfo](#) attribute), 41  
[License](#) (class in [annize.features.licensing](#)), 75  
[licensename](#) ([annize.features.distributables.debian.Package.\\_BuildInfo](#) attribute), 32  
[lightness](#) ([annize.data.color.Color](#) property), 13  
[Line](#) (class in [annize.features.version](#)), 77  
[linkname](#) ([annize.features.distributables.python\\_wheel.ExecutableLink](#) property), 39  
[LispSection](#) (in module [annize.features.distributables.debian](#)), 28  
[List](#) (class in [annize.features.base](#)), 73  
[ListParameterInfo](#) (class in [annize.object.parameter\\_info](#)), 97  
[load\(\)](#) ([annize.project.Project](#) static method), 98  
[load\(\)](#) ([annize.project.ProjectNode](#) static method), 103  
[load\\_feature\(\)](#) ([annize.project.feature\\_loader.DefaultFeatureLoader](#) method), 117  
[load\\_feature\(\)](#) ([annize.project.feature\\_loader.FeatureLoader](#) method), 116

[load\\_project\(\)](#) (in module `annize.project.loader`), 119  
[LocalRepository](#) (class in `annize.features.distributables.flatpak`), 35  
[locationstring\(\)](#) (`annize.features.files.transfer.ssh.Endpoint` method), 56  
[logo\\_image](#) (`annize.features.documentation.sphinx.output.html.HtmlOutputSpec` property), 44  
[long\\_description](#) (`annize.features.base.Data` property), 72  
[long\\_description](#) (`annize.features.distributables.python_wheel.Package._BuildInfo` attribute), 41  
[long\\_description\(\)](#) (in module `annize.features.base`), 74  
[long\\_str](#) (`annize.data.unique.UniqueId` property), 14

## M

[MachineRootDirectory](#) (class in `annize.features.files.common`), 60  
[MailSection](#) (in module `annize.features.distributables.debian`), 28  
[main\(\)](#) (in module `annize.annize_cli`), 124  
[main\\_document](#) (`annize.features.documentation.sphinx.common.DocumentGenerator` attribute), 46  
[mark\\_object\\_as\\_toplevel\(\)](#) (`annize.flow.run_context.RunContext` method), 80  
[marshaler](#) (`annize.project.file_formats.xml._XmlParser.Context` property), 108  
[marshaler](#) (`annize.project.FileNode` property), 103  
[Marshaler](#) (class in `annize.project.file_formats.xml`), 109  
[Marshaler.XmlDocumentLocation](#) (class in `annize.project.file_formats.xml`), 109  
[masterlink](#) (`annize.features.documentation.sphinx.output.html.HtmlOutputSpec` property), 44  
[match\\_arguments\(\)](#) (`annize.project.inspector.Inspector` method), 118  
[match\\_node\(\)](#) (`annize.project.inspector.Inspector` method), 118  
[matches\\_inner\\_type\(\)](#) (`annize.object.parameter_info.ParameterInfo` method), 96  
[matches\\_object\(\)](#) (`annize.object.parameter_info.ParameterInfo` method), 96  
[matches\\_object\(\)](#) (`annize.object.parameter_info.UnionParameterInfo` method), 97  
[matches\\_type\(\)](#) (`annize.object.parameter_info.ParameterInfo` method), 96  
[matching\\_by\\_argname\(\)](#) (`annize.project.inspector.Inspector.ArgumentMatchings` method), 117  
[MaterializationResult](#) (class in `annize.project.materializer`), 110  
[materialize\(\)](#) (in module `annize.project.materializer`), 110  
[MaterializerError](#), 107  
[MathematicsSection](#) (in module `annize.features.distributables.debian`), 29  
[media\\_files](#) (`annize.features.homepage.common.HomepageSection.Content` property), 64  
[mediatype](#) (`annize.features.media_galleries.Gallery.Item` property), 77  
[MediaType](#) (class in `annize.features.media_galleries`), 76  
[menu\\_entries](#) (`annize.features.distributables.flatpak.FlatpakImage._BuildInfo` attribute), 38  
[menuentries](#) (`annize.features.distributables.debian.Package._BuildInfo` attribute), 32  
[MenuEntry](#) (class in `annize.features.distributables.debian`), 23  
[MenuEntry](#) (class in `annize.features.distributables.flatpak`), 34  
[message\\_dialog\(\)](#) (`annize.user_feedback.NullUserFeedbackController` method), 122  
[message\\_dialog\(\)](#) (`annize.user_feedback.static.StaticUserFeedbackController` method), 123  
[message\\_dialog\(\)](#) (`annize.user_feedback.tty.TtyUserFeedbackController` method), 124  
[message\\_dialog\(\)](#) (`annize.user_feedback.UserFeedbackController` method), 121  
[message\\_dialog\(\)](#) (in module `annize.user_feedback`), 122  
[MetaPackagesSection](#) (in module `annize.features.distributables.debian`), 29  
[methodname](#) (`annize.features.distributables.python_wheel.ExecutableLink` property), 39  
[MiscellaneousSection](#) (in module `annize.features.distributables.debian`), 29  
[MIT](#) (in module `annize.features.licensing`), 76  
[module](#)  
[annize](#), 13  
[annize.annize\\_cli](#), 124  
[annize.asset](#), 13  
[annize.asset.data](#), 13  
[annize.asset.project\\_info](#), 13  
[annize.data](#), 13  
[annize.data.color](#), 13  
[annize.data.container](#), 14  
[annize.data.unique](#), 14

annize.data.version, 15  
 annize.features, 18  
 annize.features.authors, 72  
 annize.features.base, 72  
 annize.features.changelog, 18  
 annize.features.changelog.common, 18  
 annize.features.dependencies, 19  
 annize.features.dependencies.common, 19  
 annize.features.dependencies.python, 20  
 annize.features.distributables, 21  
 annize.features.distributables.common, 21  
 annize.features.distributables.debian, 23  
 annize.features.distributables.flatpak, 34  
 annize.features.distributables.python\_wheel, 39  
 annize.features.distributables.store, 21  
 annize.features.distributables.store.pypi, 21  
 annize.features.distributables.tar, 42  
 annize.features.documentation, 43  
 annize.features.documentation.common, 53  
 annize.features.documentation.sphinx, 43  
 annize.features.documentation.sphinx.\_utils, 46  
 annize.features.documentation.sphinx.common, 46  
 annize.features.documentation.sphinx.cpp, 50  
 annize.features.documentation.sphinx.doxygen, 50  
 annize.features.documentation.sphinx.javascript, 52  
 annize.features.documentation.sphinx.output, 43  
 annize.features.documentation.sphinx.output.common, 43  
 annize.features.documentation.sphinx.output.html, 44  
 annize.features.documentation.sphinx.output.pdf, 45  
 annize.features.documentation.sphinx.output.plaintext, 45  
 annize.features.documentation.sphinx.python, 52  
 annize.features.documentation.sphinx.rst, 53  
 annize.features.files, 55  
 annize.features.files.common, 57  
 annize.features.files.transfer, 55  
 annize.features.files.transfer.common, 55  
 annize.features.files.transfer.ssh, 56  
 annize.features.homepage, 60  
 annize.features.homepage.common, 63  
 annize.features.homepage.sections, 60  
 annize.features.homepage.sections.about, 60  
 annize.features.homepage.sections.changelog, 61  
 annize.features.homepage.sections.documentation, 61  
 annize.features.homepage.sections.download, 61  
 annize.features.homepage.sections.gallery, 62  
 annize.features.homepage.sections.imprint, 62  
 annize.features.homepage.sections.license, 63  
 annize.features.i18n, 66  
 annize.features.i18n.common, 66  
 annize.features.i18n.gettext, 67  
 annize.features.injections, 67  
 annize.features.injections.common, 67  
 annize.features.injections.python, 68  
 annize.features.licensing, 75  
 annize.features.media\_galleries, 76  
 annize.features.task, 77  
 annize.features.testing, 68  
 annize.features.testing.common, 68  
 annize.features.testing.pylint, 69  
 annize.features.testing.pytest, 69  
 annize.features.testing.pyunit, 69  
 annize.features.version, 77  
 annize.features.version\_control, 70  
 annize.features.version\_control.common, 70  
 annize.features.version\_control.git, 71  
 annize.flow, 78  
 annize.flow.run\_context, 78  
 annize.flow.runner, 83  
 annize.fs, 85  
 annize.fs.ext, 90  
 annize.i18n, 91  
 annize.object, 94  
 annize.object.controller, 95  
 annize.object.parameter\_info, 96  
 annize.project, 97  
 annize.project.feature\_loader, 116  
 annize.project.file\_formats, 107  
 annize.project.file\_formats.xml, 108  
 annize.project.inspector, 117  
 annize.project.loader, 119  
 annize.project.materializer, 110  
 annize.project.materializer.behaviors, 111  
 annize.project.materializer.behaviors.argument, 112

annize.project.materializer.behaviors.basketNetworkSection (in module an-  
 112 nize.features.distributables.debian), 29  
 annize.project.materializer.behaviors.blockNew\_value (annize.project.Node.PropertyChangedEvent  
 113 property), 100  
 annize.project.materializer.behaviors.featureUnavailable (in module an-  
 113 nize.features.distributables.debian), 29  
 annize.project.materializer.behaviors.referenceReferentCultureError, 94  
 114 node (annize.project.materializer.core.NodeMaterialization  
 annize.project.materializer.core, 114 property), 114  
 annize.project.materializer.preprocessors, node (annize.project.Project property), 98  
 116 Node (class in annize.project), 98  
 annize.ui, 120 Node.ChangeEvent (class in annize.project), 99  
 annize.ui.apps, 120 Node.ChildAddedEvent (class in annize.project), 99  
 annize.user\_feedback, 121 Node.ChildRemovedEvent (class in annize.project), 99  
 annize.user\_feedback.static, 123 Node.ChildrenListChangeEvent (class in an-  
 annize.user\_feedback.tty, 123 nize.project), 99  
 module name (annize.features.distributables.python\_wheel.Node.PropertyChangedEvent (class in an-  
 property), 39 nize.project), 99  
 MonoCliSection (in module an- node\_context() (annize.project.materializer.behaviors.argument.Argume  
 nize.features.distributables.debian), 27 method), 112  
 Mount (class in annize.fs.ext), 90 node\_context() (annize.project.materializer.behaviors.argument.Associa  
 move\_to() (annize.fs.Path method), 87 method), 112  
 MPLv11 (in module annize.features.licensing), 76 node\_context() (annize.project.materializer.behaviors.basket.BasketBeh  
 MPLv2 (in module annize.features.licensing), 76 method), 113  
 mtime() (annize.fs.Path method), 86 node\_context() (annize.project.materializer.behaviors.Behavior  
 multilanguage\_frame() (an- method), 111  
 nize.features.documentation.sphinx.output.common node\_context() (annize.project.materializer.behaviors.block.BlockBehav  
 method), 44 method), 113  
 multilanguage\_frame() (an- node\_context() (annize.project.materializer.behaviors.feature\_unavailab  
 nize.features.documentation.sphinx.output.html.HtmlOutputGenerator), 113  
 method), 45 node\_context() (annize.project.materializer.behaviors.reference.Referenc  
 MultipleValuesForSingleArgumentError, 96 method), 114  
  
**N** NodeMaterialization (class in an-  
 nize.project.materializer.core), 114  
 name (annize.features.dependencies.python.PythonPackage nodes (annize.project.inspector.Inspector.ArgumentMatchings.ArgumentMa  
 property), 20 property), 117  
 name (annize.features.distributables.debian.ExecutableLink normalize\_blockscopes() (in module an-  
 property), 23 nize.project.materializer.preprocessors),  
 name (annize.features.distributables.debian.MenuEntry 116  
 property), 23 NullUserFeedbackController (class in an-  
 name (annize.features.distributables.debian.Package.\_BuildInfo nize.user\_feedback), 122  
 attribute), 32 NumericVersionPatternSegment (class in an-  
 name (annize.features.distributables.debian.Section prop- nize.data.version), 15  
 erty), 27  
 name (annize.features.distributables.flatpak.FlatpakImage.\_BuildInfo  
 attribute), 38 object\_by\_name() (an-  
 name (annize.features.distributables.flatpak.MenuEntry nize.flow.run\_context.RunContext  
 property), 34 method),  
 name (annize.features.distributables.python\_wheel.Package object\_by\_name() (in module an-  
 attribute), 41 nize.flow.run\_context), 82  
 name (annize.features.licensing.License property), 75 object\_metadata() (an-  
 name (annize.object.parameter\_info.ParameterInfo prop- nize.flow.run\_context.RunContext  
 erty), 96 method),  
 name (annize.project.ArgumentNode property), 104 80



object\_metadata() (in module *annize.flow.run\_context*), 83  
 object\_name() (*annize.flow.run\_context.RunContext* method), 79  
 object\_name() (in module *annize.flow.run\_context*), 82  
 object\_names() (*annize.flow.run\_context.RunContext* method), 79  
 object\_names() (in module *annize.flow.run\_context*), 82  
 ObjectNode (class in *annize.project*), 104  
 objects\_by\_type() (*annize.flow.run\_context.RunContext* method), 79  
 objects\_by\_type() (in module *annize.flow.run\_context*), 82  
 objects\_for\_node() (*annize.project.materializer.MaterializationResult* method), 111  
 OcamlSection (in module *annize.features.distributables.debian*), 29  
 old\_value (*annize.project.Node.PropertyChangedEvent* property), 100  
 OldLibrariesSection (in module *annize.features.distributables.debian*), 29  
 on\_unresolvable (*annize.project.ReferenceNode* property), 105  
 OnFeatureUnavailableNode (class in *annize.project*), 106  
 OnFeatureUnavailableNode.Action (class in *annize.project*), 106  
 OptionalVersionPatternSegment (class in *annize.data.version*), 16  
 OtherOSsAndFssSection (in module *annize.features.distributables.debian*), 29  
 outdir (*annize.features.documentation.sphinx.common.Documentation* attribute), 46  
 OutOfContextError, 81  
 OutputGenerator (class in *annize.features.documentation.sphinx.output.common*), 43  
 outputspec (*annize.features.documentation.sphinx.output.common* property), 43  
 OutputSpec (class in *annize.features.documentation.common*), 54  
**P**  
 Package (class in *annize.features.distributables.debian*), 30  
 Package (class in *annize.features.distributables.python\_wheel*), 39  
 Package (class in *annize.features.distributables.tar*), 42  
 package() (*annize.features.distributables.common.PackageStore* method), 21  
 Package.\_BuildInfo (class in *annize.features.distributables.debian*), 31  
 Package.\_BuildInfo (class in *annize.features.distributables.python\_wheel*), 40  
 package\_history() (*annize.features.distributables.common.PackageStore* method), 22  
 package\_store (*annize.features.distributables.common.Group* property), 22  
 PackageStore (class in *annize.features.distributables.common*), 21  
 parameter\_name (*annize.object.controller.\_CreateObjectHelper.ParameterInfo* attribute), 95  
 ParameterInfo (class in *annize.object.parameter\_info*), 96  
 parent (*annize.project.Node* property), 101  
 parse() (in module *annize.project.file\_formats*), 108  
 parse\_file() (*annize.project.file\_formats.FileFormat* class method), 107  
 parse\_file() (*annize.project.file\_formats.xml.\_XmlParser* method), 109  
 parse\_file() (*annize.project.file\_formats.xml.XmlFileFormat* class method), 108  
 parser() (in module *annize.annize\_cli*), 124  
 ParserError, 107  
 partname (*annize.data.version.NumericVersionPatternSegment* property), 15  
 parts (*annize.features.files.common.Directory* property), 60  
 path (*annize.features.distributables.debian.ExecutableLink* property), 23  
 path (*annize.features.version\_control.git.VersionControlSystem* property), 71  
 path (*annize.features.version\_control.fresh\_temp\_directory* property), 90  
 path (*annize.project.FileNode* property), 103  
 Path (class in *annize.fs*), 85  
 path() (*annize.fs.FilesystemContent* method), 85  
 path() (*annize.fs.Path* method), 86  
 Path.\_TransferHelper (class in *annize.fs*), 88  
 Path.InTransferFilters (class in *annize.fs*), 88  
 Path.TransferFilters.And (class in *annize.fs*), 88  
 pattern (*annize.data.version.Version* property), 18  
 PdfOutputGenerator (class in *annize.features.documentation.sphinx.output.pdf*), 45  
 PdfOutputSpec (class in *annize.features.documentation.common*), 55  
 PerlSection (in module *annize.features.distributables.debian*), 29  
 PhpSection (in module *annize.features.distributables.debian*), 29  
 pkgpath\_debian (*annize.features.distributables.debian.Package.\_BuildInfo* attribute), 32

pkgpath\_documentation (an- pre\_process\_generate() (an-  
 nize.features.distributables.debian.Package.\_BuildInfo nize.features.homepage.sections.download.Section  
 attribute), 32 method), 61  
 pkgpath\_pixmaps (an- pre\_process\_generate() (an-  
 nize.features.distributables.debian.Package.\_BuildInfo nize.features.homepage.sections.gallery.Section  
 attribute), 32 method), 62  
 pkgpath\_setuppy (an- prepare() (annize.flow.run\_context.RunContext  
 nize.features.distributables.python\_wheel.Package.\_BuildInfo method), 78  
 attribute), 41 prepare\_generate() (an-  
 pkgpath\_share (annize.features.distributables.flatpak.FlatpakImage.\_BuildInfo nize.features.documentation.sphinx.output.common.OutputGenera  
 attribute), 38 method), 43  
 pkgpath\_share\_applications (an- prepare\_generate() (an-  
 nize.features.distributables.flatpak.FlatpakImage.\_BuildInfo nize.features.documentation.sphinx.output.html.HtmlOutputGenera  
 attribute), 38 method), 45  
 pkgpath\_share\_icons (an- prerm (annize.features.distributables.debian.Package.\_BuildInfo  
 nize.features.distributables.flatpak.FlatpakImage.\_BuildInfo attribute), 32  
 attribute), 38 pretty\_project\_name (annize.features.base.Data  
 pkgpath\_usrbin (annize.features.distributables.debian.Package.\_BuildInfo property), 72  
 attribute), 32 pretty\_project\_name() (in module an-  
 pkgrootpath (annize.features.distributables.debian.Package.\_BuildInfo nize.features.base), 74  
 attribute), 32 problems (annize.project.materializer.core.NodeMaterialization  
 pkgrootpath (annize.features.distributables.flatpak.FlatpakImage.\_BuildInfo property), 115  
 attribute), 38 processonly\_long\_str (annize.data.unique.UniqueId  
 pkgrootpath (annize.features.distributables.python\_wheel.Package.\_BuildInfo property), 15  
 attribute), 41 processonly\_short\_str (an-  
 pkgsize (annize.features.distributables.debian.Package.\_BuildInfo nize.data.unique.UniqueId property), 15  
 attribute), 32 PROJECT (annize.project.BlockNodeWithScope.Scope at-  
 PlaintextOutputGenerator (class in an- attribute), 106  
 nize.features.documentation.sphinx.output.plaintextProject (class in annize.project), 97  
 45 project\_authors() (in module an-  
 PlaintextOutputSpec (class in an- nize.features.authors), 72  
 nize.features.documentation.common), 55 project\_cultures() (in module an-  
 platform (annize.features.distributables.flatpak.FlatpakImage.\_BuildInfo nize.features.i18n.common), 67  
 attribute), 38 project\_default (annize.annize\_cli.Commands  
 port (annize.features.files.transfer.ssh.Endpoint prop- attribute), 125  
 erty), 56 project\_directory (annize.features.base.Data prop-  
 post\_process\_generate() (an- erty), 73  
 nize.features.homepage.common.HomepageSection project\_directory() (in module an-  
 method), 64 nize.features.base), 74  
 post\_process\_generate() (an- project\_keywords() (in module annize.features.base),  
 nize.features.homepage.sections.download.Section 73  
 method), 61 project\_licenses() (in module an-  
 postinst (annize.features.distributables.debian.Package.\_BuildInfo nize.features.licensing), 76  
 attribute), 32 project\_name (annize.features.base.Data property), 72  
 postproc() (annize.features.documentation.sphinx.output.common.OutputGenerator (annize.features.documentation.sphinx.common.Document  
 method), 43 property), 47  
 postproc() (annize.features.documentation.sphinx.output.common.OutputGenerator (annize.features.documentation.sphinx.common.Document  
 method), 45 property), 47  
 pre\_process\_generate() (an- project\_root\_directory() (in module an-  
 nize.features.homepage.common.HomepageSection nize.project.loader), 119  
 method), 64 project\_versions() (in module an-  
 pre\_process\_generate() (an- ProjectCultures (class in an-  
 nize.features.homepage.sections.documentation.Section nize.features.i18n.common), 67  
 method), 61 ProjectDefinedTranslationProvider (class in an-

*nize.features.i18n.common*), 66  
**ProjectDirectory** (class in *annize.features.files.common*), 60  
**ProjectInfoInjection** (class in *annize.features.injections.python*), 68  
**ProjectMaterializer** (class in *annize.project.materializer.core*), 115  
**projectname\_\_original** (*annize.features.documentation.sphinx.common.Document* property), 43  
**ProjectNode** (class in *annize.project*), 102  
**property\_name** (*annize.project.Node.PropertyChangedEventManager* property), 100  
**ProvidedTrStr** (class in *annize.i18n*), 92  
**public\_url** (*annize.features.distributables.flatpak.Repository* property), 35  
**PublicDomain** (in module *annize.features.licensing*), 76  
**Python** (class in *annize.features.dependencies.python*), 20  
**Python3ApiReferenceLanguage** (class in *annize.features.documentation.sphinx.python*), 52  
**PythonPackage** (class in *annize.features.dependencies.python*), 20  
**PythonSection** (in module *annize.features.distributables.debian*), 29

## R

**readme\_pdf()** (in module *annize.asset.data*), 13  
**ReadmeDocument** (class in *annize.features.documentation.sphinx.common*), 50  
**Recommended** (class in *annize.features.dependencies.common*), 20  
**red** (*annize.data.color.Color* property), 13  
**reference\_key** (*annize.project.ReferenceNode* property), 105  
**ReferenceBehavior** (class in *annize.project.materializer.behaviors.reference*), 114  
**ReferenceNode** (class in *annize.project*), 105  
**ReferenceNode.OnUnresolvableAction** (class in *annize.project*), 105  
**regex\_string()** (*annize.data.version.AbstractVersionPatternSegment* method), 15  
**regex\_string()** (*annize.data.version.ConcatenatedVersionPatternSegment* method), 17  
**regex\_string()** (*annize.data.version.NumericVersionPatternSegment* method), 16  
**regex\_string()** (*annize.data.version.OptionalVersionPatternSegment* method), 16  
**regex\_string()** (*annize.data.version.SeparatorVersionPatternSegment* method), 16  
**register\_file\_format()** (in module *annize.project.file\_formats*), 107  
**register\_output\_generator()** (in module *annize.features.documentation.sphinx.output.common*), 43  
**relative\_path** (*annize.features.files.common.FsEntry* property), 57  
**release** (*annize.features.documentation.sphinx.common.Document* property), 47  
**remove()** (*annize.fs.Path* method), 86  
**remove\_change\_handler()** (*annize.project.Node* method), 100  
**remove\_child()** (*annize.project.Node* method), 101  
**Repository** (class in *annize.features.distributables.flatpak*), 35  
**Required** (class in *annize.features.dependencies.common*), 20  
**resolve\_appendtonodes()** (in module *annize.project.materializer.preprocessors*), 116  
**resolve\_reference\_node()** (*annize.project.inspector.Inspector* method), 119  
**resolved\_type** (*annize.object.parameter\_info.ParameterInfo* property), 96  
**resolved\_type** (*annize.object.parameter\_info.UnionParameterInfo* property), 97  
**result** (*annize.features.distributables.debian.Package.\_BuildInfo* attribute), 32  
**result** (*annize.features.distributables.flatpak.FlatpakImage.\_BuildInfo* attribute), 38  
**result** (*annize.features.distributables.python\_wheel.Package.\_BuildInfo* attribute), 41  
**result** (*annize.project.materializer.core.NodeMaterialization* property), 115  
**root** (*annize.features.files.common.DirectoryPart* property), 59  
**root** (*annize.features.files.common.FsEntry* property), 57  
**root\_objects** (*annize.project.materializer.MaterializationResult* property), 111  
**rst\_text** (*annize.features.homepage.common.HomepageSection.Content* property), 64  
**RstDocument** (class in *annize.features.documentation.sphinx.common*), 49  
**RstDocumentVariant** (class in *annize.features.documentation.sphinx.common*), 49  
**RstGenerator** (class in *annize.features.documentation.sphinx.common*), 49

*nize.features.documentation.sphinx.rst*), 53  
 RubySection (in module *annize.features.distributables.debian*), 29  
 run() (*annize.features.testing.common.Test* method), 69  
 run() (*annize.features.testing.common.TestGroup* method), 69  
 run() (*annize.features.testing.pylint.Test* method), 69  
 run() (*annize.features.testing.pytest.Test* method), 69  
 run() (*annize.features.testing.pyunit.Test* method), 69  
 run\_runner() (*annize.annize\_cli.Commands.ConsoleRunner* method), 125  
 run\_runner() (*annize.flow.runner.Runner* method), 84  
 RunContext (class in *annize.flow.run\_context*), 78  
 Runner (class in *annize.flow.runner*), 83  
 RunTests (class in *annize.features.testing.common*), 68  
 RustSection (in module *annize.features.distributables.debian*), 29

## S

saturation (*annize.data.color.Color* property), 14  
 save() (*annize.project.Project* method), 98  
 save() (*annize.project.ProjectNode* method), 102  
 save\_filencode\_to\_file() (*annize.project.file\_formats.xml.Marshaler* method), 110  
 ScalarValueNode (class in *annize.project*), 104  
 scalehue() (*annize.data.color.Color* method), 14  
 ScienceSection (in module *annize.features.distributables.debian*), 29  
 scope (*annize.project.BlockNodeWithScope* property), 106  
 ScreenLockingCategory (in module *annize.features.distributables.debian*), 27  
 ScreenSavingCategory (in module *annize.features.distributables.debian*), 27  
 sdk (*annize.features.distributables.flatpak.FlatpakImage.Builder* attribute), 38  
 section (*annize.features.distributables.debian.Package.\_BuildInfo* attribute), 32  
 Section (class in *annize.features.distributables.debian*), 27  
 Section (class in *annize.features.homepage.sections.about*), 60  
 Section (class in *annize.features.homepage.sections.changelog*), 61  
 Section (class in *annize.features.homepage.sections.documentation*), 61  
 Section (class in *annize.features.homepage.sections.download*), 61  
 Section (class in *annize.features.homepage.sections.gallery*),

62  
 Section (class in *annize.features.homepage.sections.imprint*), 62  
 Section (class in *annize.features.homepage.sections.license*), 63  
 sections (*annize.features.homepage.common.Homepage* property), 65  
 segment\_names (*annize.data.version.AbstractVersionPatternSegment* property), 15  
 segment\_names (*annize.data.version.ConcatenatedVersionPatternSegment* property), 17  
 segment\_names (*annize.data.version.NumericVersionPatternSegment* property), 15  
 segment\_names (*annize.data.version.OptionalVersionPatternSegment* property), 16  
 segment\_names (*annize.data.version.VersionPattern* property), 17  
 segments (*annize.data.version.VersionPattern* property), 17  
 segments\_tuples (*annize.data.version.Version* property), 18  
 segments\_tuples (*annize.features.version\_control.common.BuildVersion* property), 70  
 segments\_tuples\_to\_text() (*annize.data.version.AbstractVersionPatternSegment* method), 15  
 segments\_tuples\_to\_text() (*annize.data.version.ConcatenatedVersionPatternSegment* method), 17  
 segments\_tuples\_to\_text() (*annize.data.version.NumericVersionPatternSegment* method), 16  
 segments\_tuples\_to\_text() (*annize.data.version.OptionalVersionPatternSegment* method), 16  
 segments\_tuples\_to\_text() (*annize.data.version.SeparatorVersionPatternSegment* method), 16  
 segments\_tuples\_to\_text() (*annize.data.version.VersionPattern* method), 17  
 segments\_values (*annize.data.version.Version* property), 18  
 SeparatorVersionPatternSegment (class in *annize.data.version*), 16  
 serialize\_for\_confpy() (in module *annize.features.documentation.sphinx.\_utils*), 46  
 ServiceDescription (class in *annize.features.distributables.debian*), 30  
 services (*annize.features.distributables.debian.Package.\_BuildInfo*



attribute), 32  
 set\_materialized\_result() (annize.project.materializer.core.NodeMaterialization method), 114  
 set\_object\_metadata() (annize.flow.run\_context.RunContext method), 81  
 set\_object\_metadata() (in module annize.flow.run\_context), 83  
 set\_object\_name() (annize.flow.run\_context.RunContext method), 79  
 set\_object\_name() (in module annize.flow.run\_context), 82  
 set\_problems() (annize.project.materializer.core.NodeMaterialization method), 114  
 set\_selected\_task() (annize.flow.runner.Runner method), 84  
 setuppy\_conf (annize.features.distributables.python\_wheel.Package.\_BuildInfo attribute), 41  
 Share (class in annize.features.distributables.flatpak), 35  
 shares (annize.features.distributables.flatpak.FlatpakImage.\_BuildInfo attribute), 38  
 ShellsSection (in module annize.features.distributables.debian), 29  
 short\_desc (annize.features.documentation.sphinx.output.TrStr property), 44  
 short\_desc (annize.features.homepage.common.HomepageSection property), 64  
 short\_str (annize.data.unique.UniqueId property), 15  
 short\_title (annize.features.documentation.sphinx.output.TrStr property), 44  
 show\_task\_chooser() (annize.annize\_cli.Commands.ConsoleRunner method), 125  
 show\_task\_chooser() (annize.flow.runner.Runner method), 84  
 show\_task\_execution() (annize.annize\_cli.Commands.ConsoleRunner method), 125  
 show\_task\_execution() (annize.flow.runner.Runner method), 84  
 show\_task\_execution\_success() (annize.annize\_cli.Commands.ConsoleRunner method), 125  
 show\_task\_execution\_success() (annize.flow.runner.Runner method), 84  
 SimpleProjectHomepage (class in annize.features.homepage.common), 65  
 SKIP (annize.project.ReferenceNode.OnUnresolvableAction attribute), 105  
 SKIP\_BLOCK (annize.project.OnFeatureUnavailableNode.Action attribute), 106  
 SKIP\_NODE (annize.project.OnFeatureUnavailableNode.Action attribute), 106  
 sockets (annize.features.distributables.flatpak.FlatpakImage.\_BuildInfo attribute), 38  
 sort\_index (annize.features.homepage.common.HomepageSection property), 64  
 SoundSection (in module annize.features.distributables.debian), 29  
 source (annize.features.distributables.debian.Package.\_BuildInfo attribute), 32  
 source (annize.features.distributables.flatpak.FlatpakImage.\_BuildInfo attribute), 38  
 source (annize.features.distributables.python\_wheel.Package.\_BuildInfo attribute), 41  
 source (annize.features.documentation.sphinx.common.ApiReferenceLanguage property), 48  
 source (annize.features.documentation.sphinx.common.RstDocumentVariable property), 49  
 source\_path (annize.features.files.common.DirectoryPart property), 69  
 StaticUserFeedbackController (class in annize.user\_feedback.static), 123  
 store\_build\_package() (annize.features.distributables.common.PackageStore method), 21  
 String (class in annize.features.i18n.common), 66  
 string (annize.features.i18n.common.TrStr property), 92  
 stringname (annize.i18n.TrStr property), 92  
 studio() (annize.annize\_cli.Commands method), 126  
 subcode (annize.i18n.Culture property), 93  
 summary (annize.features.base.Data property), 72  
 summary (annize.features.distributables.debian.Package.\_BuildInfo attribute), 32  
 summary() (in module annize.features.base), 74  
**T**  
 target\_node (annize.project.Node.ChangeEvent property), 99  
 Task (class in annize.features.task), 77  
 TasksSection (in module annize.features.distributables.debian), 29  
 temp\_clone() (annize.fs.Path method), 87  
 Test (class in annize.features.testing.common), 69  
 Test (class in annize.features.testing.pylint), 69  
 Test (class in annize.features.testing.pytest), 69  
 Test (class in annize.features.testing.pyunit), 69  
 TestGroup (class in annize.features.testing.common), 69  
 TexSection (in module annize.features.distributables.debian), 29  
 text (annize.data.version.Version property), 18  
 text (annize.features.changelog.common.Item property), 18  
 text (annize.features.licensing.License property), 75  
 text (annize.features.version\_control.common.BuildVersion property), 71

text\_to\_segments\_tuples() (annize.data.version.VersionPattern method), 17  
 TextProcessingSection (in module annize.features.distributables.debian), 29  
 theme (annize.features.documentation.sphinx.output.html.HtmlOutputSize.user\_feedback.tty), 123  
 type (annize.project.inspector.Inspector.TypeInfo property), 118  
 type\_name (annize.project.ObjectNode property), 104  
 type\_parameter\_infos() (in module annize.object.parameter\_info), 97  
 typename (annize.project.inspector.Inspector.TypeInfo property), 118  
**U**  
 undo\_changes() (annize.project.ProjectNode method), 103  
 UnresolvableReferenceError, 107  
 UnsatisfiableUserFeedbackAttemptError, 122  
 UnspecifiedCulture (class in annize.features.i18n.common), 67  
 UnspecifiedCulture (class in annize.i18n), 93  
 UpdatePOs (class in annize.features.i18n.gettext), 67  
 Upload (class in annize.features.distributables.store.pypi), 21  
 Upload (class in annize.features.files.transfer.common), 56  
 upload() (annize.features.distributables.flatpak.LocalRepository method), 35  
 upload() (annize.features.distributables.flatpak.Repository method), 35  
 UserFeedbackController (class in annize.user\_feedback), 121  
 username (annize.features.files.transfer.ssh.Endpoint property), 56  
 UtilitiesSection (in module annize.features.distributables.debian), 29  
**V**  
 value (annize.project.ScalarValueNode property), 105  
 version (annize.features.changelog.common.Entry property), 18  
 version (annize.features.dependencies.python.PythonPackage property), 20  
 version (annize.features.distributables.debian.Package.\_BuildInfo attribute), 32  
 version (annize.features.distributables.python\_wheel.Package.\_BuildInfo attribute), 41  
 version (annize.features.documentation.sphinx.common.Document property), 47  
 version (annize.features.version.Line property), 77

Version (class in *annize.data.version*), 17  
 Version (class in *annize.features.version*), 77  
 VersionControlSystem (class in *annize.features.version\_control.common*), 70  
 VersionControlSystem (class in *annize.features.version\_control.git*), 71  
 VersionControlSystemsSection (in module *annize.features.distributables.debian*), 30  
 VersionPattern (class in *annize.data.version*), 17  
 VIDEO (*annize.features.media\_galleries.MediaType* attribute), 76  
 VideoSection (in module *annize.features.distributables.debian*), 30  
 VirtualPackagesSection (in module *annize.features.distributables.debian*), 30

## W

wait\_finished() (*annize.flow.runner.Runner* method), 85  
 WebServersSection (in module *annize.features.distributables.debian*), 28  
 WebSoftwareSection (in module *annize.features.distributables.debian*), 30  
 with\_modified() (*annize.data.color.Color* method), 14  
 with\_updates() (*annize.object.controller.\_CreateObjectHelper.ParameterConfig* method), 95  
 write\_file() (*annize.fs.Path* method), 86

## X

XfceSection (in module *annize.features.distributables.debian*), 30  
 XmlFileFormat (class in *annize.project.file\_formats.xml*), 108  
 XWindowSystemSoftwareSection (in module *annize.features.distributables.debian*), 30

## Z

ZopePloneFrameworkSection (in module *annize.features.distributables.debian*), 30