
Caparg

Release 19.11.0

Nov 11, 2019

Contents

| | | |
|----------|----------------------------|----------|
| 1 | API | 1 |
| | Python Module Index | 3 |
| | Index | 5 |

Captain arguments – ready for subcommand!

A more functional way of doing argument parsing.

`caparg.command(_name, *args, **kwargs)`

A command (or a subcommand)

Parameters

- **_name** (*str*) – the name of the subcommand (for top-level, “”)
- ***args** (*tuple*) – commands and options
- ****kwargs** (*dict*) – options by name

`caparg.option(type, required=False, have_default=False)`

An option

Note that an option does not know its name. It will usually be used in a dictionary where the name is specified as its key, and it has a method `with_name` to add the name when processed.

Parameters

- **type** (*class*) – the expected input type
- **required** (*bool*) – whether option name is expected
- **have_default** (*bool*) – whether to auto-create a default based on the type

`caparg.positional(name, type, required=False, have_default=False)`

A positional argument

Parameters

- **name** (*str*) – name of argument
- **type** (*type*) – expected type
- **required** (*boolean*) – argument is required (default `False`)

- **have_default** (*boolean*) – if argument is not given, generate a default (default False)

Returns Something with `add_to` and `get_value`

`caparg.options(**kwargs)`

Wrap options

This is used to be able to put options at the beginning of the argument list of a command. Since options are given by keywords, they must follow positional arguments.”

Parameters ****kwargs** (*Dict[str, option]*) – Mapping

exception `caparg.ParseError` (*message*)

Command-line arguments are invalid

C

caparg, [1](#)

C

`caparg` (*module*), 1

`command()` (*in module caparg*), 1

O

`option()` (*in module caparg*), 1

`options()` (*in module caparg*), 2

P

`ParseError`, 2

`positional()` (*in module caparg*), 1