rvlm.entrypoint Documentation

Release 0.0.2

Pavel Kretov

February 08, 2014

Contents

Python Module Index 3

Converts function's arguments to getopt-style command line options and arguments. This may definitely help in writing small and clear scripts, with no ugly command line parsing code.

This module can:

- Automatically generate argument parsers basing on "main" function signature and docstring.
- Automatically run the "main" function when a script is called directly, but not when it is included as a module.

Right after this module was written, its author discovered for himself the `entrypoint` module (https://pypi.python.org/pypi/entrypoint). But it appeared to be that original `entrypoint` behaves sometimes in a strange way. So, it was decided to continue development, and also to rename this module to `rvlm.entrypoint` from former `rvlm.argmap`, becase this name sounds better still doesn't introduce names conflict by having a prefix.

exception `rvlm.entrypoint.`**`ParserError`**
Exception to throw

`rvlm.entrypoint.`**`call`**(*func*, *args=None*, *emptyVarArgs=True*, *helpOptions=True*, *shortOptions=True*, *docStrings=True*, *changeNames=True*, *raiseOnError=True*)
Convert function *func* argument convention to command line options and runs it with converted arguments.

Parameters

- **func** – Function to be run. Must have parameter information available through inspection. It also must not have ***kwargs* parameter.
- **args** – Getopt-style command line parameters. It is a list of command line options and arguments mostly like sys.argv array, but unlike it without first item (`sys.argv[0]`) which commonly contains script name. Default value `None` means that `sys.argv` will be used to get that list.
- **emptyVarArgs** – Specifies whether target function 'func' can take empty arguments list as its **vararg* parameter. Default value `True` represents the fact that there is no way to set this restriction in Python syntax. But setting it to `False` will require at least one getopt-style argument to be passed as **vararg* (if it is present, of course).
- **helpOption** – Specifies whether to generate *--help* (and *-h*) option or not. Default value is `True` which means help option will be enabled. Setting this to `False` will disable help messages and also will make shorthand *-h* available for use by another option (see parameter *shortOptions* for more info).
- **shortOptions** – Enables automatic generation of short options. Short options will be the first letters of arguments name comverted to lower case. If several optional arguments start with the same letter, no short option will be generated. Default value is `True`. Note that short option *-h* is reserved for help message display if *helpOption* parameter is set to `True`.
- **docStrings** – Try to find parameters description in function's docstring (`__doc__`). Description are found using very simple regular expression, so this feature may fail sometimes. For this to work parameters must be described on separate lines like in the following:

```
"""
* cmd Command to run interactively.
- cmd Command to run interactively.
* cmd: Command to run interactively.
*** cmd - Command to run interactively.
:param cmd: Command to run interactively.
@param cmd Command to run interactively.
"""
```

where *cmd* must be the *exact* parameter name. Some more variants are available, though. Default value is `True`.

- **changeNames** – Enables automatic arguments renaming. Setting this to `True` will change naming convention of function arguments to dash-style. For the following example function:

```
def func(logFile=None, StartDate=None, stop_at_exit=False):
   pass;
```

arguments will be converted to *--log-file*, *--start-date* and *--stop-at-exit*. But the usage clause will look something like this:

```
Usage: func.py [--log-file LOG_FILE] [--start-date START_DATE] ...
```

Returns Function conveys return value from target function *func*.

rvlm.entrypoint.**mainfunction**(***kwargs*)

Runs the function if the module in which it is declared is being run directly from the command line. Putting the following before the function definition would be similar to:

```
if __name__ == '__main__':
    func()
```

This will work most expectedly as the outermost decorator, as it will call the function before any more outwards decorators have been applied.

r

rvlm.entrypoint, 1