

---

# mork Documentation

*Release 0.0.0.dev0*

**Dan Ryan <[dan@danryan.co](mailto:dan@danryan.co)>**

**Oct 28, 2019**



---

## Contents:

---

<b>1</b>	<b>mork: A project for installing packages across the virtualenv boundary.</b>	<b>1</b>
1.1	Summary . . . . .	1
1.1.1	See What's Installed . . . . .	1
1.1.2	Install A Package . . . . .	1
1.1.3	Uninstall a Package . . . . .	2
1.1.4	Display Information about Python . . . . .	2
1.1.5	Run Commands Inside the Virtualenv . . . . .	2
<b>2</b>	<b>mork package</b>	<b>3</b>
2.1	Submodules . . . . .	7
2.1.1	mork.virtualenv module . . . . .	7
<b>3</b>	<b>Indices and tables</b>	<b>13</b>
	<b>Python Module Index</b>	<b>15</b>
	<b>Index</b>	<b>17</b>



# CHAPTER 1

---

mork: A project for installing packages across the virtualenv boundary.

---

## 1.1 Summary

**Mork\_** is a library designed for installing and querying python packages inside virtual environments.

### 1.1.1 See What's Installed

```
>>> import mork
>>> venv = mork.VirtualEnv.from_project_path('/home/user/git/pipenv')
>>> dists = venv.get_distributions()
>>> [dist for dist in dists][:3]
[wheel 0.31.1 (/home/user/.virtualenvs/pipenv-MfOPs1lW/lib/python3.7/site-packages), Werkzeug 0.14.1 (/home/user/.virtualenvs/pipenv-MfOPs1lW/lib/python3.7/site-packages), vistir 0.1.4 (/home/user/.virtualenvs/pipenv-MfOPs1lW/lib/python3.7/site-packages)]
```

### 1.1.2 Install A Package

```
>>> from requirementslib.models.requirements import Requirement
>>> r = Requirement.from_line("requests")
>>> venv.install(r, editable=False)
```

### 1.1.3 Uninstall a Package

```
>>> pkg = "pytz"
>>> with venv.uninstall(pkg, auto_confirm=True) as uninstall:
    if uninstall.paths:
        cleaned = pkg
>>> print("Removed package: %s" % cleaned)
```

### 1.1.4 Display Information about Python

```
>>> venv.python
'/home/user/.virtualenvs/pipenv-MfOPs11W/bin/python'
>>> venv.python_version
'3.7'
```

### 1.1.5 Run Commands Inside the Virtualenv

```
>>> cmd = venv.run("env")
>>> [line for line in cmd.out.splitlines() if line.startswith("VIRTUAL_ENV")]
['VIRTUAL_ENV=/user/hawk/.virtualenvs/pipenv-MfOPs11W']
>>> cmd = venv.run_py(["import os; print(os.environ.get('VIRTUAL_ENV'))"])
Deactivating virtualenv...
>>> cmd.out
'/home/user/.virtualenvs/pipenv-MfOPs11W\n'
>>> with venv.activated():
    print(os.environ["VIRTUAL_ENV"])
/home/hawk/.virtualenvs/pipenv-MfOPs11W
```

Read the documentation.

# CHAPTER 2

---

## mork package

---

```
class mork.VirtualEnv(prefix=None, base_working_set=None, is_venv=True)
Bases: object
```

**activated**(*include\_extras=True, extra\_dists=[]*)

A context manager which activates the virtualenv.

**Parameters** **extra\_dists** (*list*) – Paths added to the context after the virtualenv is activated.

**This context manager sets the following environment variables:**

- *PYTHONUSERBASE*
- *VIRTUAL\_ENV*
- *PYTHONIOENCODING*
- *PYTHONDONTWRITEBYTECODE*

In addition, it activates the virtualenv inline by calling *activate\_this.py*.

**add\_dist**(*dist\_name*)

**base\_paths**

Returns the context appropriate paths for the environment.

**Returns** A dictionary of environment specific paths to be used for installation operations

**Return type** *dict*

---

**Note:** The implementation of this is borrowed from a combination of pip and virtualenv and is likely to change at some point in the future.

---

```
>>> from pipenv.core import project
>>> from pipenv.environment import Environment
>>> env = Environment(prefix=project.virtualenv_location, is_venv=True,
-> sources=project.sources)
>>> import pprint
>>> pprint.pprint(env.base_paths)
{'PATH': '/home/hawk/.virtualenvs/pipenv-MfOPs11W/bin:/bin:/usr/bin',
'PYTHONPATH': '/home/hawk/.virtualenvs/pipenv-MfOPs11W/lib/python3.7/site-
->packages',
'data': '/home/hawk/.virtualenvs/pipenv-MfOPs11W',
'include': '/home/hawk/.pyenv/versions/3.7.1/include/python3.7m',
'libdir': '/home/hawk/.virtualenvs/pipenv-MfOPs11W/lib/python3.7/site-
->packages',
'platinclude': '/home/hawk/.pyenv/versions/3.7.1/include/python3.7m',
'platlib': '/home/hawk/.virtualenvs/pipenv-MfOPs11W/lib/python3.7/site-
->packages',
'platstdlib': '/home/hawk/.virtualenvs/pipenv-MfOPs11W/lib/python3.7',
'prefix': '/home/hawk/.virtualenvs/pipenv-MfOPs11W',
'purelib': '/home/hawk/.virtualenvs/pipenv-MfOPs11W/lib/python3.7/site-
->packages',
'scripts': '/home/hawk/.virtualenvs/pipenv-MfOPs11W/bin',
'stdlib': '/home/hawk/.pyenv/versions/3.7.1/lib/python3.7'}
```

**dist\_is\_in\_project**(*dist*)

**classmethod filter\_sources**(*requirement, sources*)

**find\_egg**(*egg\_dist*)

**classmethod from\_project\_path**(*path*)

Utility for finding a virtualenv location based on a project path

**get\_distributions**()

Retrieves the distributions installed on the library path of the virtualenv

**Returns** A set of distributions found on the library path

**Return type** iterator

**get\_finder**()

**get\_installed\_packages**()

**get\_monkeypatched\_pathset**()

Returns a monkeypatched *UninstallPathset* for using to uninstall packages from the virtualenv

**Returns** A patched *UninstallPathset* which enables uninstallation of venv packages

**Return type** pip.\_internal.req.req\_uninstall.UninstallPathset

**get\_outdated\_packages**()

**get\_package\_info**()

**get\_setup\_install\_args**(*pkgname, setup\_py, develop=False*)

Get setup.py install args for installing the supplied package in the virtualenv

**Parameters**

- **pkgname** (*str*) – The name of the package to install
- **setup\_py** (*str*) – The path to the setup file of the package
- **develop** (*bool*) – Whether the package is in development mode

**Returns** The installation arguments to pass to the interpreter when installing

**Return type** list

**classmethod** `get_sys_path(python_path)`

Get the `sys.path` data for a given python executable.

**Parameters** `python_path (str)` – Path to a specific python executable.

**Returns** The system path information for that python runtime.

**Return type** list

**get\_working\_set()**

Retrieve the working set of installed packages for the virtualenv.

**Returns** The working set for the virtualenv

**Return type** `pkg_resources.WorkingSet`

**classmethod** `get_workon_home()`

**initial\_working\_set**

**install (req, editable=False, sources=[])**

Install a package into the virtualenv

**Parameters**

- `req` (`requirementslib.models.requirement.Requirement`) – A requirement to install
- `editable` (`bool`) – Whether the requirement is editable, defaults to False
- `sources` (`list`) – A list of pip sources to consult, defaults to []

**Returns** A return code, 0 if successful

**Return type** int

**is\_installed (pkgnname)**

Given a package name, returns whether it is installed in the virtual environment

**Parameters** `pkgnname (str)` – The name of a package

**Returns** Whether the supplied package is installed in the environment

**Return type** bool

**libdir**

**locate\_dist (dist)**

**classmethod** `normalize_path(path)`

**paths**

**python**

Path to the environment python

**python\_version**

**pyversion**

**classmethod** `resolve_dist (dist, working_set)`

Given a local distribution and a working set, returns all dependencies from the set.

**Parameters**

- **dist** (`pkg_resources.Distribution`) – A single distribution to find the dependencies of

- **working\_set** (`pkg_resources.WorkingSet`) – A working set to search for all packages

**Returns** A set of distributions which the package depends on, including the package

**Return type** `set(pkg_resources.Distribution)`

**run** (`cmd, cwd='.'`)

Run a command with `Popen` in the context of the virtualenv

**Parameters**

- **cmd** (`str or list`) – A command to run in the virtual environment

- **cwd** (`str`) – The working directory in which to execute the command, defaults to `os.getcwd()`

**Returns** A finished command object

**Return type** `Popen`

**run\_py** (`cmd, cwd='.'`)

Run a python command in the virtualenv context.

**Parameters**

- **cmd** (`str or list`) – A command to run in the virtual environment - runs with `python -c`

- **cwd** (`str`) – The working directory in which to execute the command, defaults to `os.getcwd()`

**Returns** A finished command object

**Return type** `Popen`

**safe\_import** (`name`)

Helper utility for reimporting previously imported modules while inside the venv

**script\_basedir**

Path to the environment scripts dir

**scripts\_dir**

**setuptools\_install** (`chdir_to, pkg_name, setup_py_path=None, editable=False`)

Install an sdist or an editable package into the virtualenv

**Parameters**

- **chdir\_to** (`str`) – The location to change to

- **setup\_py\_path** (`str`) – The path to the setup.py, if applicable defaults to None

- **editable** (`bool`) – Whether the package is editable, defaults to False

**sys\_path**

The system path inside the environment

**Returns** The `sys.path` from the environment

**Return type** `list`

**sys\_prefix**

The prefix run inside the context of the environment

**Returns** The python prefix inside the environment

**Return type** `sys.prefix`

**system\_paths**

**uninstall** (`pkgname, *args, **kwargs`)  
A context manager which allows uninstallation of packages from the virtualenv

**Parameters** `pkgnome (str)` – The name of a package to uninstall

```
>>> venv = VirtualEnv("/path/to/venv/root")
>>> with venv.uninstall("pytz", auto_confirm=True, verbose=False) as _:
...     uninstalier:
...         cleaned = uninstalier.paths
... if cleaned:
...     print("uninstalled packages: %s" % cleaned)
```

## 2.1 Submodules

### 2.1.1 mork.virtualenv module

**class** `mork.virtualenv.PatchedUninstaller`  
Bases: `object`

**class** `mork.virtualenv.VirtualEnv (prefix=None, base_working_set=None, is_venv=True)`  
Bases: `object`

**activated** (`include_extras=True, extra_dists=[]`)  
A context manager which activates the virtualenv.

**Parameters** `extra_dists (list)` – Paths added to the context after the virtualenv is activated.

This context manager sets the following environment variables:

- `PYTHONUSERBASE`
- `VIRTUAL_ENV`
- `PYTHONIOENCODING`
- `PYTHONDONTWRITEBYTECODE`

In addition, it activates the virtualenv inline by calling `activate_this.py`.

**add\_dist** (`dist_name`)

**base\_paths**  
Returns the context appropriate paths for the environment.

**Returns** A dictionary of environment specific paths to be used for installation operations

**Return type** `dict`

---

**Note:** The implementation of this is borrowed from a combination of pip and virtualenv and is likely to change at some point in the future.

```
>>> from pipenv.core import project
>>> from pipenv.environment import Environment
>>> env = Environment(prefix=project.virtualenv_location, is_venv=True,
-> sources=project.sources)
>>> import pprint
>>> pprint.pprint(env.base_paths)
{'PATH': '/home/hawk/.virtualenvs/pipenv-MfOPs11W/bin:/bin:/usr/bin',
'PYTHONPATH': '/home/hawk/.virtualenvs/pipenv-MfOPs11W/lib/python3.7/site-
->packages',
'data': '/home/hawk/.virtualenvs/pipenv-MfOPs11W',
'include': '/home/hawk/.pyenv/versions/3.7.1/include/python3.7m',
'libdir': '/home/hawk/.virtualenvs/pipenv-MfOPs11W/lib/python3.7/site-
->packages',
'platinclude': '/home/hawk/.pyenv/versions/3.7.1/include/python3.7m',
'platlib': '/home/hawk/.virtualenvs/pipenv-MfOPs11W/lib/python3.7/site-
->packages',
'platstdlib': '/home/hawk/.virtualenvs/pipenv-MfOPs11W/lib/python3.7',
'prefix': '/home/hawk/.virtualenvs/pipenv-MfOPs11W',
'purelib': '/home/hawk/.virtualenvs/pipenv-MfOPs11W/lib/python3.7/site-
->packages',
'scripts': '/home/hawk/.virtualenvs/pipenv-MfOPs11W/bin',
'stdlib': '/home/hawk/.pyenv/versions/3.7.1/lib/python3.7'}
```

**dist\_is\_in\_project** (*dist*)

**classmethod filter\_sources** (*requirement, sources*)

**find\_egg** (*egg\_dist*)

**classmethod from\_project\_path** (*path*)

Utility for finding a virtualenv location based on a project path

**get\_distributions** ()

Retrieves the distributions installed on the library path of the virtualenv

**Returns** A set of distributions found on the library path

**Return type** iterator

**get\_finder** ()

**get\_installed\_packages** ()

**get\_monkeypatched\_pathset** ()

Returns a monkeypatched *UninstallPathset* for using to uninstall packages from the virtualenv

**Returns** A patched *UninstallPathset* which enables uninstallation of venv packages

**Return type** pip.\_internal.req.req\_uninstall.UninstallPathset

**get\_outdated\_packages** ()

**get\_package\_info** ()

**get\_setup\_install\_args** (*pkgname, setup\_py, develop=False*)

Get setup.py install args for installing the supplied package in the virtualenv

**Parameters**

- **pkgname** (*str*) – The name of the package to install
- **setup\_py** (*str*) – The path to the setup file of the package
- **develop** (*bool*) – Whether the package is in development mode

**Returns** The installation arguments to pass to the interpreter when installing

**Return type** list

**classmethod** `get_sys_path(python_path)`

Get the `sys.path` data for a given python executable.

**Parameters** `python_path (str)` – Path to a specific python executable.

**Returns** The system path information for that python runtime.

**Return type** list

**get\_working\_set()**

Retrieve the working set of installed packages for the virtualenv.

**Returns** The working set for the virtualenv

**Return type** `pkg_resources.WorkingSet`

**classmethod** `get_workon_home()`

**initial\_working\_set**

**install (req, editable=False, sources=[])**

Install a package into the virtualenv

**Parameters**

- `req` (`requirementslib.models.requirement.Requirement`) – A requirement to install
- `editable` (`bool`) – Whether the requirement is editable, defaults to False
- `sources` (`list`) – A list of pip sources to consult, defaults to []

**Returns** A return code, 0 if successful

**Return type** int

**is\_installed (pkgnname)**

Given a package name, returns whether it is installed in the virtual environment

**Parameters** `pkgnname (str)` – The name of a package

**Returns** Whether the supplied package is installed in the environment

**Return type** bool

**libdir**

**locate\_dist (dist)**

**classmethod** `normalize_path(path)`

**paths**

**python**

Path to the environment python

**python\_version**

**pyversion**

**classmethod** `resolve_dist (dist, working_set)`

Given a local distribution and a working set, returns all dependencies from the set.

**Parameters**

- **dist** (`pkg_resources.Distribution`) – A single distribution to find the dependencies of

- **working\_set** (`pkg_resources.WorkingSet`) – A working set to search for all packages

**Returns** A set of distributions which the package depends on, including the package

**Return type** `set(pkg_resources.Distribution)`

**run** (`cmd, cwd='.'`)

Run a command with `Popen` in the context of the virtualenv

**Parameters**

- **cmd** (`str or list`) – A command to run in the virtual environment

- **cwd** (`str`) – The working directory in which to execute the command, defaults to `os.getcwd()`

**Returns** A finished command object

**Return type** `Popen`

**run\_py** (`cmd, cwd='.'`)

Run a python command in the virtualenv context.

**Parameters**

- **cmd** (`str or list`) – A command to run in the virtual environment - runs with `python -c`

- **cwd** (`str`) – The working directory in which to execute the command, defaults to `os.getcwd()`

**Returns** A finished command object

**Return type** `Popen`

**safe\_import** (`name`)

Helper utility for reimporting previously imported modules while inside the venv

**script\_basedir**

Path to the environment scripts dir

**scripts\_dir**

**setuptools\_install** (`chdir_to, pkg_name, setup_py_path=None, editable=False`)

Install an sdist or an editable package into the virtualenv

**Parameters**

- **chdir\_to** (`str`) – The location to change to

- **setup\_py\_path** (`str`) – The path to the setup.py, if applicable defaults to None

- **editable** (`bool`) – Whether the package is editable, defaults to False

**sys\_path**

The system path inside the environment

**Returns** The `sys.path` from the environment

**Return type** `list`

**sys\_prefix**

The prefix run inside the context of the environment

**Returns** The python prefix inside the environment

**Return type** `sys.prefix`

### `system_paths`

#### `uninstall(pkgname, *args, **kwargs)`

A context manager which allows uninstallation of packages from the virtualenv

**Parameters** `pkgname (str)` – The name of a package to uninstall

```
>>> venv = VirtualEnv("/path/to/venv/root")
>>> with venv.uninstall("pytz", auto_confirm=True, verbose=False) as _:
...     uninstaller:
...         cleaned = uninstaller.paths
... if cleaned:
...     print("uninstalled packages: %s" % cleaned)
```



# CHAPTER 3

---

## Indices and tables

---

- genindex
- modindex
- search



---

## Python Module Index

---

### m

`mork`, 3  
`mork.virtualenv`, 7



---

## Index

---

### A

activated () (*mork.VirtualEnv method*), 3  
activated () (*mork.virtualenv.VirtualEnv method*), 7  
add\_dist () (*mork.VirtualEnv method*), 3  
add\_dist () (*mork.virtualenv.VirtualEnv method*), 7

### B

base\_paths (*mork.VirtualEnv attribute*), 3  
base\_paths (*mork.virtualenv.VirtualEnv attribute*), 7

### D

dist\_is\_in\_project () (*mork.VirtualEnv method*),  
    4  
dist\_is\_in\_project ()  
    (*mork.virtualenv.VirtualEnv method*), 8

### F

filter\_sources () (*mork.VirtualEnv class method*),  
    4  
filter\_sources ()  
    (*mork.virtualenv.VirtualEnv*  
        *class method*), 8  
find\_egg () (*mork.VirtualEnv method*), 4  
find\_egg () (*mork.virtualenv.VirtualEnv method*), 8  
from\_project\_path () (*mork.VirtualEnv*  
    *class method*), 4  
from\_project\_path () (*mork.virtualenv.VirtualEnv*  
    *class method*), 8

### G

get\_distributions () (*mork.VirtualEnv method*), 4  
get\_distributions () (*mork.virtualenv.VirtualEnv*  
    *method*), 8  
get\_finder () (*mork.VirtualEnv method*), 4  
get\_finder () (*mork.virtualenv.VirtualEnv method*), 8  
get\_installed\_packages ()  
    (*mork.VirtualEnv*  
        *method*), 4  
get\_installed\_packages ()  
    (*mork.virtualenv.VirtualEnv method*), 8

get\_monkeypatched\_pathset ()  
    (*mork.VirtualEnv method*), 4  
get\_monkeypatched\_pathset ()  
    (*mork.virtualenv.VirtualEnv method*), 8  
get\_outdated\_packages ()  
    (*mork.VirtualEnv*  
        *method*), 4  
get\_outdated\_packages ()  
    (*mork.virtualenv.VirtualEnv method*), 8  
get\_package\_info () (*mork.VirtualEnv method*), 4  
get\_package\_info ()  
    (*mork.virtualenv.VirtualEnv*  
        *method*), 8  
get\_setup\_install\_args ()  
    (*mork.VirtualEnv*  
        *method*), 4  
get\_setup\_install\_args ()  
    (*mork.virtualenv.VirtualEnv method*), 8  
get\_sys\_path () (*mork.VirtualEnv class method*), 5  
get\_sys\_path ()  
    (*mork.virtualenv.VirtualEnv*  
        *class method*), 9  
get\_working\_set () (*mork.VirtualEnv method*), 5  
get\_working\_set ()  
    (*mork.virtualenv.VirtualEnv*  
        *method*), 9  
get\_workon\_home ()  
    (*mork.VirtualEnv*  
        *method*), 5  
get\_workon\_home ()  
    (*mork.virtualenv.VirtualEnv*  
        *class method*), 9

### I

initial\_working\_set (*mork.VirtualEnv attribute*),  
    5  
initial\_working\_set (*mork.virtualenv.VirtualEnv*  
    *attribute*), 9  
install () (*mork.VirtualEnv method*), 5  
install () (*mork.virtualenv.VirtualEnv method*), 9  
is\_installed () (*mork.VirtualEnv method*), 5  
is\_installed ()  
    (*mork.virtualenv.VirtualEnv*  
        *method*), 9

### L

libdir (*mork.VirtualEnv attribute*), 5  
libdir (*mork.virtualenv.VirtualEnv attribute*), 9

locate\_dist () (*mork.VirtualEnv method*), 5  
locate\_dist () (*mork.virtualenv.VirtualEnv method*),  
9  
system\_paths (*mork.VirtualEnv attribute*), 7  
system\_paths (*mork.virtualenv.VirtualEnv attribute*),  
11

## M

*mork (module)*, 3  
*mork.virtualenv (module)*, 7

## N

normalize\_path () (*mork.VirtualEnv class method*),  
5  
normalize\_path () (*mork.virtualenv.VirtualEnv  
class method*), 9

## P

PatchedUninstaller (*class in mork.virtualenv*), 7  
paths (*mork.VirtualEnv attribute*), 5  
paths (*mork.virtualenv.VirtualEnv attribute*), 9  
python (*mork.VirtualEnv attribute*), 5  
python (*mork.virtualenv.VirtualEnv attribute*), 9  
python\_version (*mork.VirtualEnv attribute*), 5  
python\_version (*mork.virtualenv.VirtualEnv at-  
tribute*), 9  
pyversion (*mork.VirtualEnv attribute*), 5  
pyversion (*mork.virtualenv.VirtualEnv attribute*), 9

## R

resolve\_dist () (*mork.VirtualEnv class method*), 5  
resolve\_dist () (*mork.virtualenv.VirtualEnv class  
method*), 9  
run () (*mork.VirtualEnv method*), 6  
run () (*mork.virtualenv.VirtualEnv method*), 10  
run\_py () (*mork.VirtualEnv method*), 6  
run\_py () (*mork.virtualenv.VirtualEnv method*), 10

## S

safe\_import () (*mork.VirtualEnv method*), 6  
safe\_import () (*mork.virtualenv.VirtualEnv method*),  
10  
script\_basedir (*mork.VirtualEnv attribute*), 6  
script\_basedir (*mork.virtualenv.VirtualEnv at-  
tribute*), 10  
scripts\_dir (*mork.VirtualEnv attribute*), 6  
scripts\_dir (*mork.virtualenv.VirtualEnv attribute*),  
10  
setuptools\_install () (*mork.VirtualEnv method*),  
6  
setuptools\_install ()  
    (*mork.virtualenv.VirtualEnv method*), 10  
sys\_path (*mork.VirtualEnv attribute*), 6  
sys\_path (*mork.virtualenv.VirtualEnv attribute*), 10  
sys\_prefix (*mork.VirtualEnv attribute*), 6  
sys\_prefix (*mork.virtualenv.VirtualEnv attribute*), 10

## U

uninstall () (*mork.VirtualEnv method*), 7  
uninstall () (*mork.virtualenv.VirtualEnv method*), 11

## V

VirtualEnv (*class in mork*), 3  
VirtualEnv (*class in mork.virtualenv*), 7