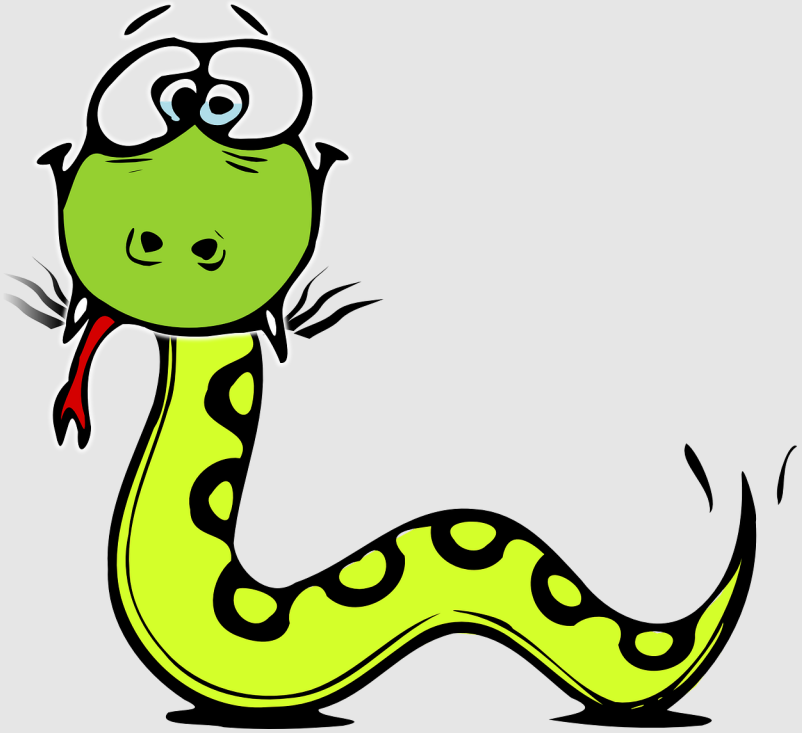

What is that...?

`pip -r` Option

Install from the given requirements file

```
$ pip install -r requirements.txt
```

`python -m` Option

Runs the named library module as a script

```
python -m venv venv
python -m unittest discover
python -m pdb some_script.py
```

`coverage -m` Option

The same as `python -m` - runs the named library module as a script

```
coverage run -m unittest discover
coverage report -m
```

`pylint --recursive=y` Option

This option makes `pylint` attempt to discover all modules (files ending with .py extension) and all explicit packages (all directories containing a `__init__.py` file).

```
pylint --recursive=y ./src ./test
```

`autopep8 --in-place --aggressive --aggressive`

Options

- `--in-place` - make changes to files in place
- `--aggressive` - enable non-whitespace changes; multiple `--aggresive` result in more aggressive changes

```
autopep8 --in-place --aggressive --aggressive src/*.py test/*.py
```

`__init__.py`

> The `__init__.py` files are required to make Python treat directories containing the file as packages. This prevents directories with a common name, such as `string`, unintentionally hiding valid modules that occur later on the module search path. In the simplest case, `__init__.py` can just be an empty file, but it can also execute initialization code for the package.

`__pycache__`

> To speed up loading modules, Python caches the compiled version of each module in the `__pycache__` directory under the name `module.version.pyc`, where the version encodes the format of the compiled file; it generally contains the Python version number.

PEP-20 or The Zen of Python

```
(env) $ python
```

```
>>> import this
```