pip -Package Management System

By Allen Huang

*common pip commands，the following commands need to be executed in terminal

* user guide: https://pip.pypa.io/en/stable/user_guide/

1. Basic commands

all the commands and options that you can use

```
pip help
```

you can also add a specific command name

```
pip help list
```

looking for a package. It will return package name and a brief description

```
pip search <package name>
```

install a package

```
pip install <package>
```

install a previous version of package

```
pip install <package>==<semantic version>
```

*semantic version: 2.20.1，2: the major version，20: the minor version, 1: patches/ bug fixes

* 2.20.*: latest compatible version of 2.20

Example:

1. pip install request ~= 2.9.0 ⇔ pip install request == 2.9.*

2. pip install requese == 2.*

show us all the package we have installed, also include the version number

```
pip list
```

uninstall package

```
pip uninstall <package name>
```

check if a package is the latest version

```
pip list -o
pip list --outdated
```

update a package

```
pip install -U <package name>
```

2. Freeze commands

store all of our packages and version numbers in a requirments format, pop out to a file

```
pip freeze > requirements.txt
```

receive

**how people receive your requirements and using pip to install*

**r means we are going to use a requirments file*

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

a way to update all of the outdate packages

```
pip freeze --local | grep -v '^\-e' | cut -d = -f 1 | xargs -n1 pip install -U
```

3. Virtual Environment (vritualenv)

**vritualeny can help us to separate different Python environment for different projects*

3.1 pipenv

install

```
pip install pipenv
```

install packages

```
pipenv install <package name>
```

find the directory of venv

```
pipenv --venv
```

activate

```
pipenv shell
```

deactivate

```
exit
```

delete the directory

```
rm -rf <path of that directory>
```

install all the dependencies in a pipfile

```
pipenv install
```

check all installed dependencies

```
pipenv graph
```

update a package

```
pipenv update <package name>
```

3.2 vritualenv

install

```
pip install vritualenv
```

check global packages

```
pip list
```

make a directory

```
mkdir Environments
```

cd to that directory, which is currently empty

```
cd !$
ls
```

make first environment

```
vritualenv project1_env
```

activate

```
source project1_env/bin/activate
```

find out if you are in this environment

**in this environment, global packages does not exist, the package installed in this env will not affect other envs*

```
which python
which pip
```

use these package in other project

**take only the local dependencies*

pip freeze --local > requirements.txt

cat on the requirements

```
cat requirements.txt
```

quit your local environment

```
deactivate
```

get rid of this local environment

```
rm -rf project1_env/
```

open another local environment

**we can specify this version of python we want to use*

```
vritualenv -p /usr/bin/python2.6 py26_env
```

```
source py26_env/bin/activate
```

check the version of python in this environment

```
pip --version
```

install the packages

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