

국민대 빅데이터경영MBA 파이썬 프로그래밍을 통한 데이터 분석

2016년 1학기 강의
정광윤 anappleofmyeyes@gmail.com



국민대학교
KOOKMIN UNIVERSITY

경영대학원
GRADUATE SCHOOL OF BUSINESS ADMINISTRATION

01

Development Environment

Python으로 개발하기 전에 개발 환경에 친숙해질 필요가 있습니다. 그래야 더 나은 패키지를 관리하고 더 나은 분석이 가능해질 수 있습니다.

Kookmin University
Graduate School of Business Administration
MBA in Big Data Analytics

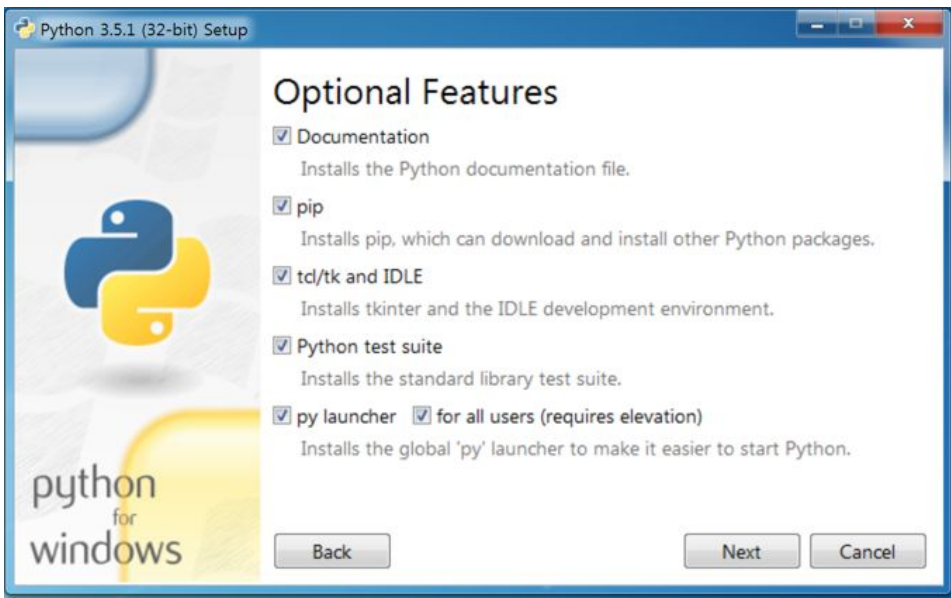
Python 설치

Python



www.python.org에서 각 운영체제에 맞게 다운로드를 받은 후 설치를 시작합니다.
Add Python 3.5 to PATH를 꼭 체크 후 Customize installation을 클릭합니다.

Python



기본적으로 모두 체크가 되어있지만 눈여겨 봐야할 것은 두 번째 **pip**이 체크가 되어있는 것을 꼭 확인하고 **Next**를 클릭합니다.
또 뭔가 설정하는 화면이 나오는데 신경쓰지 말고 설치를 계속 진행합니다.

The image shows a Windows XP desktop environment. In the background, a '시스템 속성' (System Properties) window is open, displaying the '환경 변수' (Environment Variables) tab. The 'initialkommit' user's variables are listed, with the 'Path' variable selected. The '시스템 변수' (System Variables) section is also visible, showing variables like 'ComSpec', 'FP_NO_HOST_C...', 'NUMBER_OF_PR...', 'OS', and 'Path'. The '확인' (OK) button is highlighted in blue.

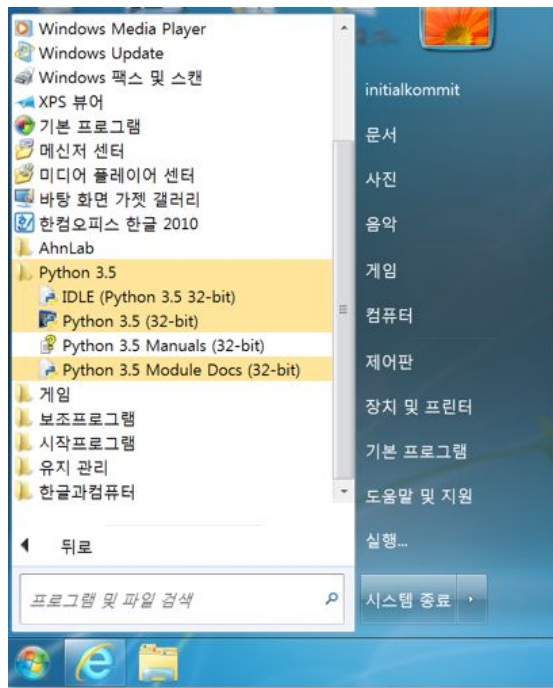
이전 화면에서 **Path** 추가를 체크하지 않았다면 수동으로 입력해야 합니다. 어느 경로에서든 **Python**을 실행하기 위해서이지요. 이전 슬라이드에서 **Path**를 추가하지 않았다면 직접 아래 경로 2개를 입력해줍니다.

```
C:\Users\사용자\로봇계정\AppData\Local\Programs\Python\Python35-32\Scripts\;
```

C:\Users\사용자\로봇계정\AppData\Local\Programs\Python\Python35-32\python.exe 이 있는 경로

만약 Add Python 3.5 to path에 체크를 하지 않고 넘어갔다면 위와 같이 설정을 할 수 있습니다.

Python



잘 설치했는지 확인해봅니다.

Python

```
관리자: C:\Windows\system32\cmd.exe
C:\Users\Winitalkommit>python
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:38:48) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> exit()

C:\Users\Winitalkommit>_
```

```
관리자: C:\Windows\system32\cmd.exe
C:\Users\Winitalkommit>pip

Usage:
  pip <command> [options]

Commands:
  install           Install packages.
  uninstall         Uninstall packages.
  freeze            Output installed packages in requirements format.
  list              List installed packages.
  show              Show information about installed packages.
  search            Search PyPI for packages.
  wheel             Build wheels from your requirements.
  help             Show help for commands.

General Options:
  -h, --help        Show help.
  --isolated        Run pip in an isolated mode, ignoring
                   environment variables and user configuration.
  -v, --verbose     Give more output. Option is additive, and can be
                   used up to 3 times.
```

잘 설치했는지 두 가지를 확인해봅니다.

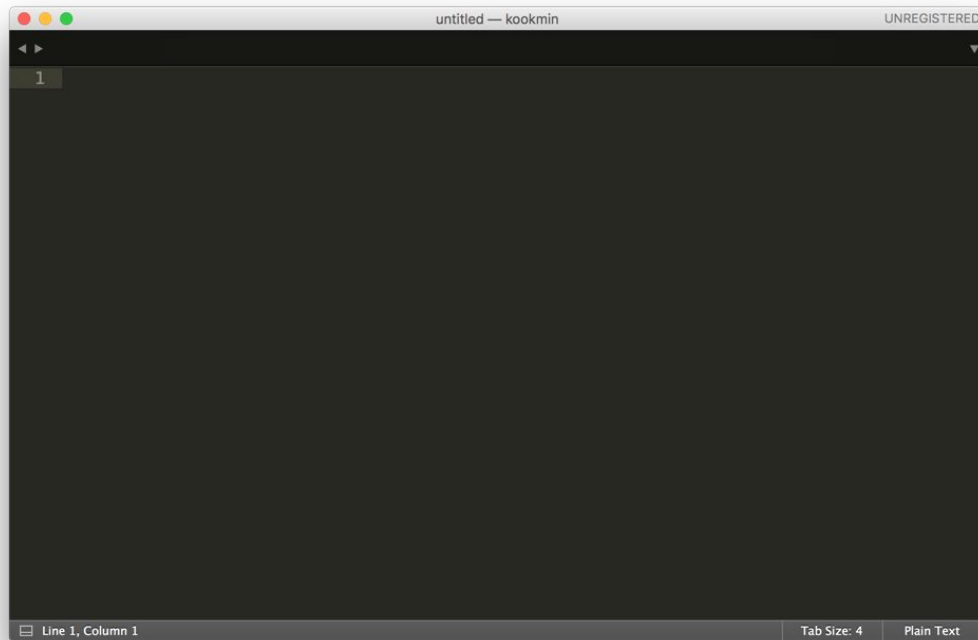
1. 왼쪽 화면처럼 python을 실행해봅니다.
2. 오른쪽 화면처럼 pip을 실행해봅니다.

만약 잘 실행이 되지 않는다면 Path설정이 잘 못 됐을 가능성이 큼니다.

Editor - Sublime Text 3 설치

코딩을 할 편집기(Editor)를 설치해보겠습니다.

Sublime Text 3

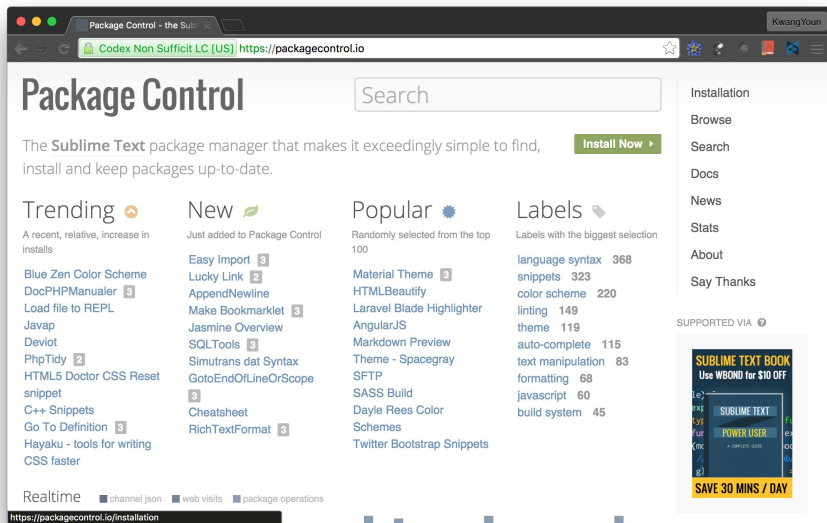


<https://www.sublimetext.com/3> 에서 다운로드 받아 설치 후 실행하면 위와 같은 모습이 나옵니다.

Sublime Text 3

Package Control

- SBT3는 메모장처럼 단순한 에디터가 아닙니다.
- 에디터에 사용자가 필요한 여러가지 기능을 패키지는 이름으로 설치할 수 있습니다.
- 먼저 패키지를 설치하고 관리할 수 있는 기능이 바로 **Package Control**이라고 합니다.
- <http://packagecontrol.io> 으로 이동합니다.
- 우측 상단의 **Install Now** 버튼을 클릭하여 SBT3의 패키지를 관리할 수 있도록 **Package Control**을 설치하겠습니다.

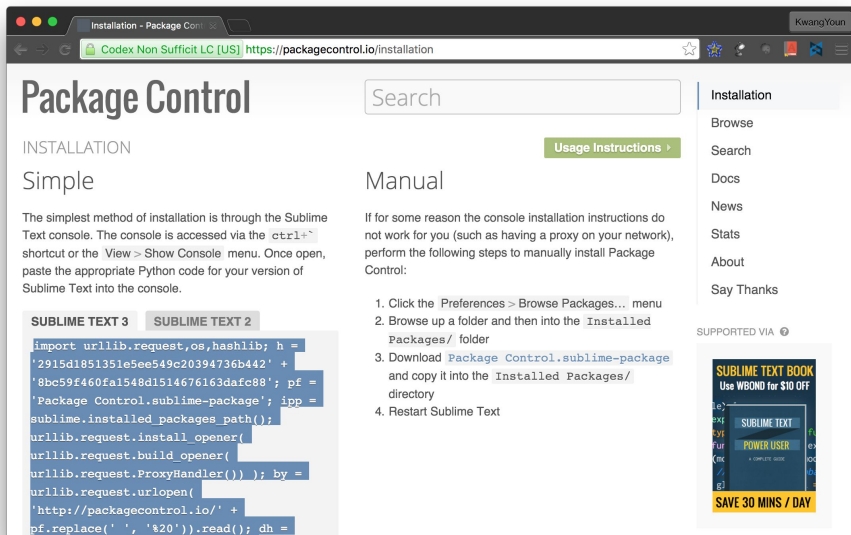


SBT3는 Sublime Text 3를 뜻합니다.

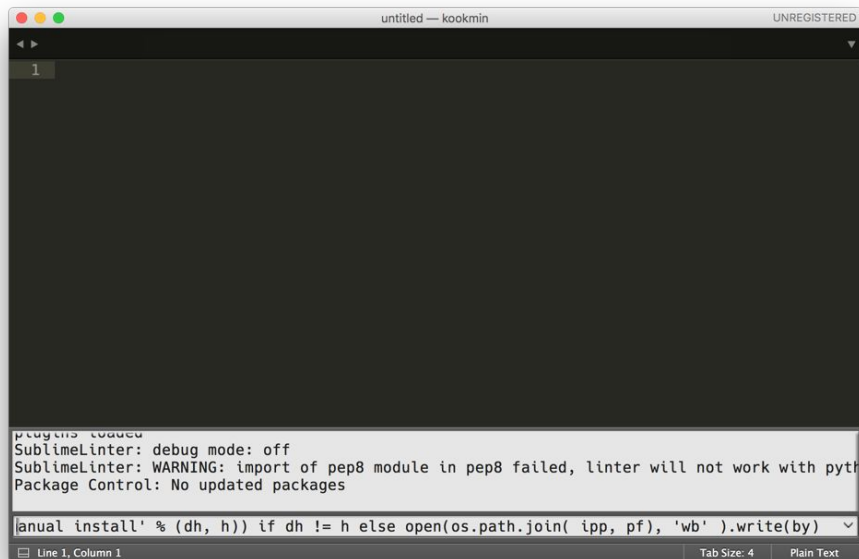
Sublime Text 3

Package Control

- Install Now 버튼을 클릭해 들어가면 왼쪽에 있는 화면이 보입니다.
- 좌측 하단에 뭐라 알 수 없는 글(?)들이 많은데 이를 전부 복사합니다.
- 우리는 Sublime Text 3를 설치했기 때문에 Sublime Text 3 탭을 선택해 복사합니다.



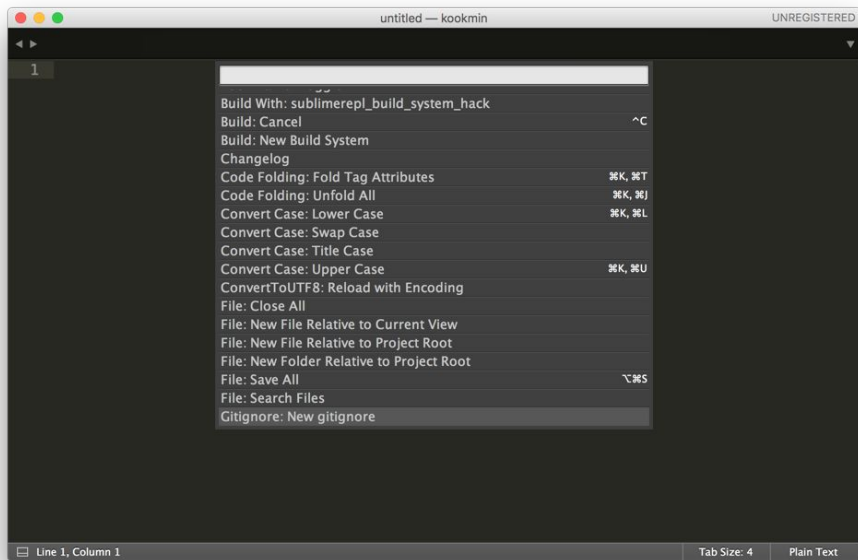
Sublime Text 3



Package 설치

- SBT3를 실행하여 단축키 **Ctrl+`** 을 이용해 Console을 띄웁니다.
- 그리고 앞 장 슬라이드에서 복사했던 것을 붙여넣기하고 엔터를 입력합니다.
- 그러면 이 후 알아서 설치를 합니다.
- 설치가 다 됐다고 해서 특별히 메시지가 나오는 것은 아닙니다. 무언가 메시지를 계속 보여주다 메시지를 안보여주는 순간이 오는데 메시지 내용을 잘 보고 설치가 다 됐음을 알 수 있습니다.
- 다만 설치 중 **missing** 등의 에러 메시지 나올 수도 있는데 신경쓰지 않고 넘어갑니다.
- 설치가 다 되면 재시작합니다.

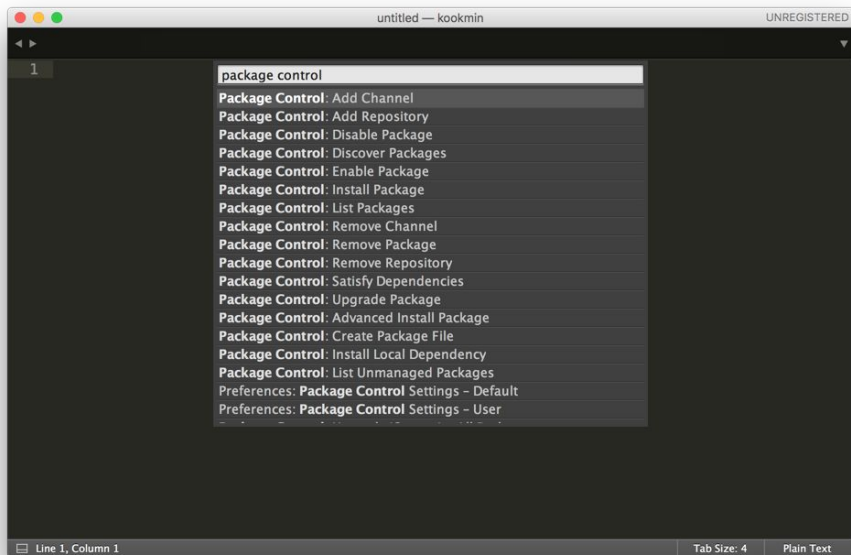
Sublime Text 3



Command Palette

- SBT3에 명령어를 찾고 실행할 수 있는 기능이 Command Palette입니다.
- 단축키 **Ctrl + Shift + P** 를 통해 실행합니다.
- 혹은 메뉴 > Tools > Command Palette로 실행합니다.
- Command Palette 기능으로 Package Control을 실행해 보도록 하겠습니다.

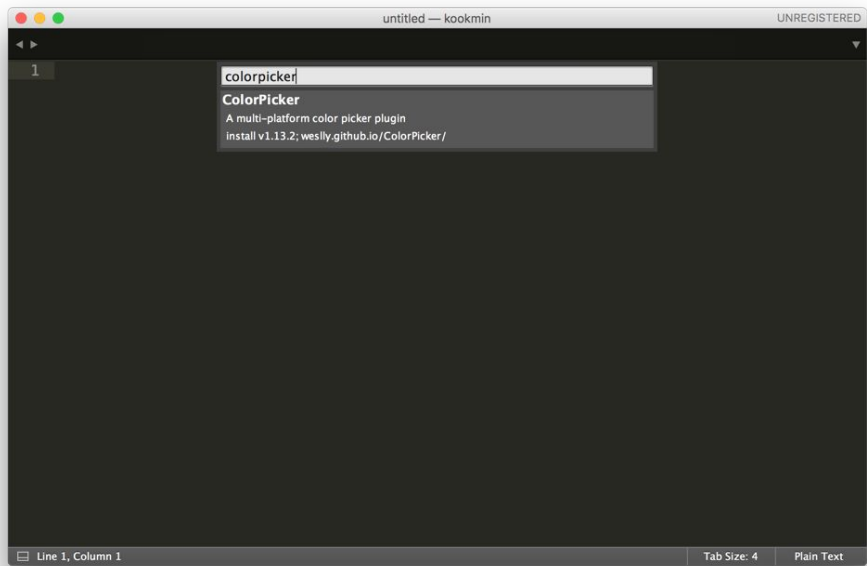
Sublime Text 3



Command Palette: package control 찾기

- Command Palette에 package control을 입력하면 package control과 관련된 명령어들이 나열됩니다.
- 이중에 우리는 Package Control: Install Package를 실행하여 Package를 아주 쉽게 찾아 설치해보겠습니다.

Sublime Text 3

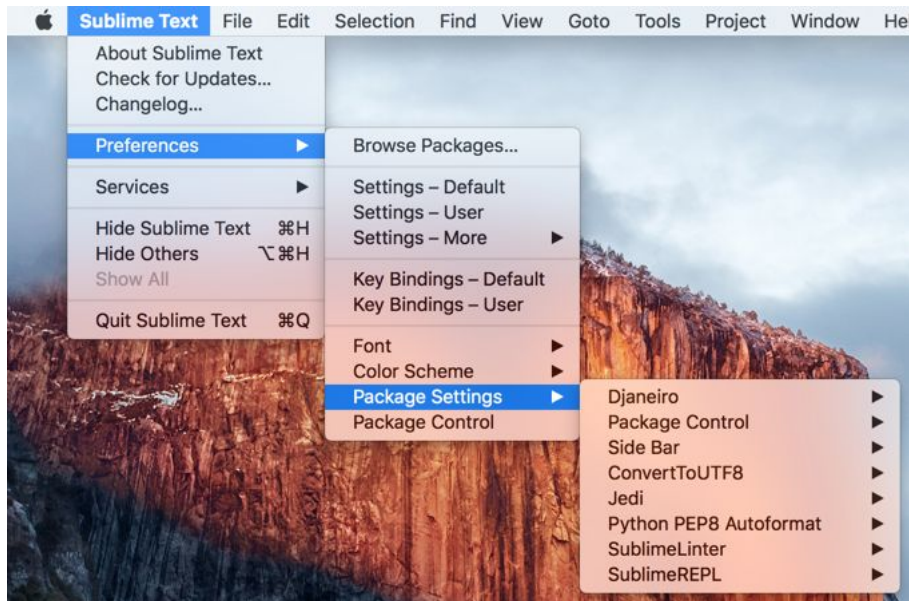


Package 설치

- Python 자동 생성 도구:
Jedi - Python Autocompletion
- Python 문법 검사 도구:
SublimeLinter, SublimeLinter - pylint, pep8
- Python 실행 도구: SublimeREPL
- 한글 처리 관련: ConvertToUTF-8

여러분들이 직접 위의 **Package**를 찾아 설치해 보도록 하겠습니다. 몇 가지 **Keyword**를 입력하는 것만으로도 다양한 **Package**를 쉽게 찾을 수 있습니다.

Sublime Text 3



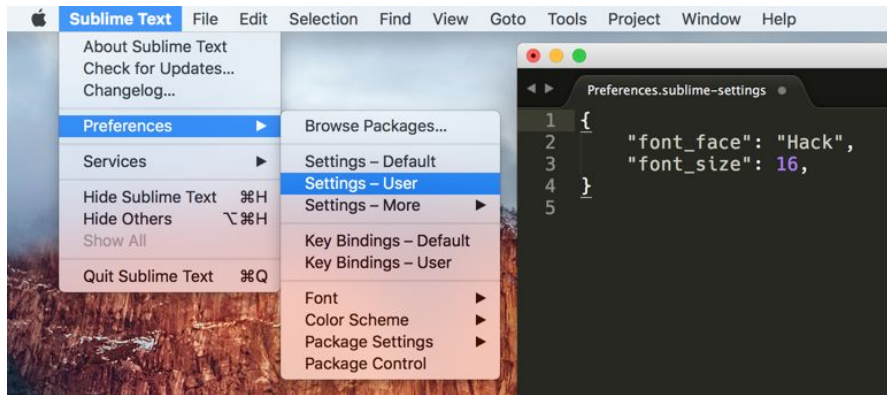
Package Settings

- Menu > Preferences > Package Settings 에 위에서 우리가 설치한 패키지가 있음을 확인할 수 있고 또한 여러가지 세팅도 할 수 있습니다.

Programming Font 설치

코딩할 때 i, l 등의 글자가 헛갈려 실수를 할 가능성이 있습니다.
그렇기 때문에 가독성이 좋은 폰트를 설치해보도록 하겠습니다.

Programming Font



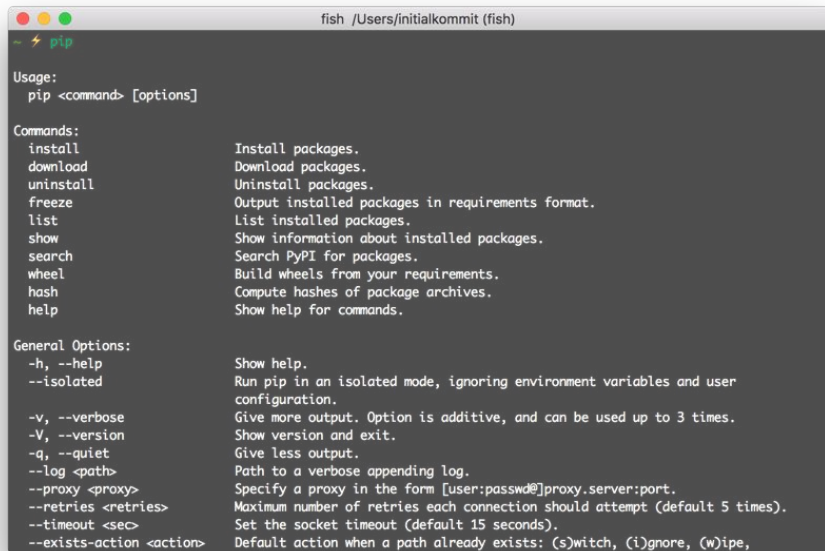
Settings - User

- 코딩을 할 때는 눈이 침침해집니다. 이 때 침침해지는 눈에 가독성을 더하면 눈이 좀 더 편안해지겠죠!?
- 그래서 많은 개발자들이 코딩 폰트(혹은 프로그래밍 폰트)를 따로 세팅합니다.
- 나눔고딩코딩, 핵(Hack) 등 다양하며 개인의 취향대로 다운로드 받아 설치합니다.
- 이후 SBT3의 Menu > Preferences > Package Settings 에 각자 설치한 폰트를 옆의 화면처럼 설정합니다.

Package Manager 설치

Package Manager 라는 개념을 알아보고
이를 통해 쉽고 편하게 Package를 관리해보도록 하겠습니다.

Package Manager

A terminal window titled 'fish /Users/initialkommit (fish)' showing the output of the 'pip --help' command. The output is divided into 'Usage:', 'Commands:', and 'General Options:' sections, listing various pip commands and their functions.

```
fish /Users/initialkommit (fish)
~ pip
Usage:
  pip <command> [options]

Commands:
  install      Install packages.
  download    Download packages.
  uninstall   Uninstall packages.
  freeze      Output installed packages in requirements format.
  list        List installed packages.
  show        Show information about installed packages.
  search      Search PyPI for packages.
  wheel       Build wheels from your requirements.
  hash        Compute hashes of package archives.
  help        Show help for commands.

General Options:
  -h, --help            Show help.
  --isolated            Run pip in an isolated mode, ignoring environment variables and user configuration.
  -v, --verbose         Give more output. Option is additive, and can be used up to 3 times.
  -V, --version         Show version and exit.
  -q, --quiet           Give less output.
  --log <path>         Path to a verbose appending log.
  --proxy <proxy>       Specify a proxy in the form [user:passwd@]proxy.server:port.
  --retries <retries>   Maximum number of retries each connection should attempt (default 5 times).
  --timeout <sec>       Set the socket timeout (default 15 seconds).
  --exists-action <action> Default action when a path already exists: (s)witch, (i)gnore, (w)ipe,
```

Package를 관리해주는 매니저

- SBT에만 Package Control이 있는 것이 아닙니다. 아래와 같이 OS, Language별로 Package Manager라는 것이 존재합니다.

- OS
 - Ubuntu: apt-get
 - CentOS: yum
 - OSX: brew
- Language
 - Front-End for Web: npm, bower
 - Python: pip (pip installs package)
 - Ruby: gem

- SBT와 마찬가지로 Python도 오픈되어 있는 패키지가 많습니다. 이를 PIP으로 관리해줍니다. 국민대 파이썬 데이터

Package Manager

```
fish /Users/initialkommit (fish)
~ ⚡ pip freeze
astroid==1.3.8
logilab-common==1.0.2
mysql-connector-python-cext==2.1.3
pbr==1.8.0
pep257==0.6.0
pep8==1.6.2
pylint==1.4.4
six==1.10.0
stevedore==1.8.0
vboxapi==1.0
virtualenv==13.1.2
virtualenv-clone==0.2.6
virtualenvwrapper==4.7.1
virtualfish==1.0.0
wheel==0.26.0
You are using pip version 8.0.3, however version 8.1.0 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
~ ⚡
```

Python Package 목록 확인

- pip freeze
- freeze! 꼼짝마! 라는 command로 pip으로 설치한 Package를 확인해볼 수 있습니다.
- 그런데 노란색 글씨로 위화감을 조성하는 글이 나옵니다.
- 괜찮습니다. pip 명령어가 업그레이드 버전이 나왔으니 업그레이드 하라는 공지입니다.
- 노란색 글씨가 시키는대로 `pip install --upgrade pip`을 실행해 보겠습니다.

python -m pip install --upgrade pip 라고 해도 됩니다.

Package Manager

```
fish /Users/initialkommit (fish)
pep8==1.6.2
pylint==1.4.4
six==1.10.0
stevedore==1.8.0
vboxapi==1.0
virtualenv==13.1.2
virtualenv-clone==0.2.6
virtualenvwrapper==4.7.1
virtualfish==1.0.0
wheel==0.26.0
You are using pip version 8.0.3, however version 8.1.0 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
~ ✨ pip install --upgrade pip
Collecting pip
  Downloading pip-8.1.0-py2.py3-none-any.whl (1.2MB)
    100% |#####| 1.2MB 308kB/s
Installing collected packages: pip
  Found existing installation: pip 8.0.3
    Uninstalling pip-8.0.3:
      Successfully uninstalled pip-8.0.3
  Successfully installed pip-8.1.0
~ ✨
```

Python Package Install

- 편하게 자기가 알아서 해줍니다.

- 1) Downloading
- 2) Installing
- 3) Uninstalling if found existing installation

- 이게 바로 Package Manager 라는 것입니다.

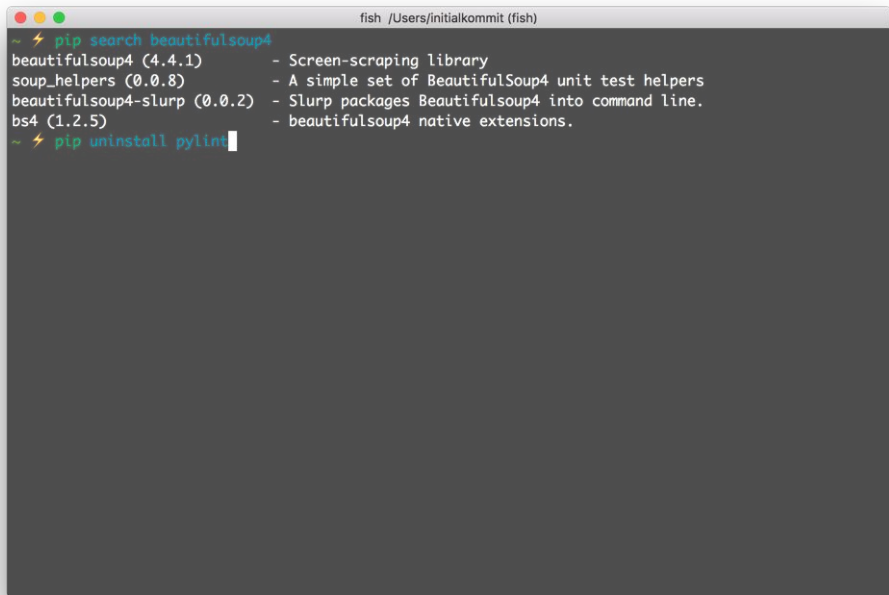
Package Manager

```
fish /Users/initialkommit (fish)
~ $ pip install pylint
Collecting pylint
  Downloading pylint-1.5.4-py2.py3-none-any.whl (547kB)
    100% |#####| 552kB 2.4MB/s
Collecting colorama (from pylint)
  Downloading colorama-0.3.7-py2.py3-none-any.whl
Requirement already satisfied (use --upgrade to upgrade): six in /usr/local/lib/python3.5/site-packages (from pylint)
Collecting astroid<1.5.0,>=1.4.1 (from pylint)
  Downloading astroid-1.4.4-py2.py3-none-any.whl (211kB)
    100% |#####| 215kB 1.2MB/s
Collecting lazy-object-proxy (from astroid<1.5.0,>=1.4.1->pylint)
  Downloading lazy-object-proxy-1.2.1.tar.gz
Collecting wrapt (from astroid<1.5.0,>=1.4.1->pylint)
  Downloading wrapt-1.10.6.tar.gz
Building wheels for collected packages: lazy-object-proxy, wrapt
  Running setup.py bdist_wheel for lazy-object-proxy ... done
  Stored in directory: /Users/initialkommit/Library/Caches/pip/wheels/d5/a1/c9/c42eabc6c5a6b352393b2cc4c500da7c0253e218556cf1aee
  Running setup.py bdist_wheel for wrapt ... done
  Stored in directory: /Users/initialkommit/Library/Caches/pip/wheels/00/c5/bc/d83286f6a26384eb928e7f1818358f324ad15ac1447df6e267
Successfully built lazy-object-proxy wrapt
Installing collected packages: colorama, lazy-object-proxy, wrapt, astroid, pylint
Successfully installed astroid-1.4.4 colorama-0.3.7 lazy-object-proxy-1.2.1 pylint-1.5.4 wrapt-1.10.6
~ $
```

Python Package Install

- pip install 패키지이름
- 이번에는 pylint라는 Package를 설치해보겠습니다.
- SBT3에서 SublimeLinter - pylint라는 Package를 설치했었습니다. pylint는 Linter라는 것으로 Python의 문법검사를 할 수 있게끔 해줍니다. 그래서 SBT3에서 SublimeLinter - pylint를 사용하기 위해서는 pylint를 미리 설치해놓고 있어야 합니다.
- 우리는 옆 화면가 같이 pip install 이라는 명령어로 아주 쉽게 패키지를 설치했습니다. 그렇다면 이런 패키지를 어떻게 찾고 어떻게 지울까요?

Package Manager



```
fish /Users/initialkommit (fish)
~ $ pip search beautifulsoup4
beautifulsoup4 (4.4.1) - Screen-scraping library
soup_helpers (0.0.8) - A simple set of BeautifulSoup4 unit test helpers
beautifulsoup4-slurp (0.0.2) - Slurp packages BeautifulSoup4 into command line.
bs4 (1.2.5) - BeautifulSoup4 native extensions.
~ $ pip uninstall pylint
```

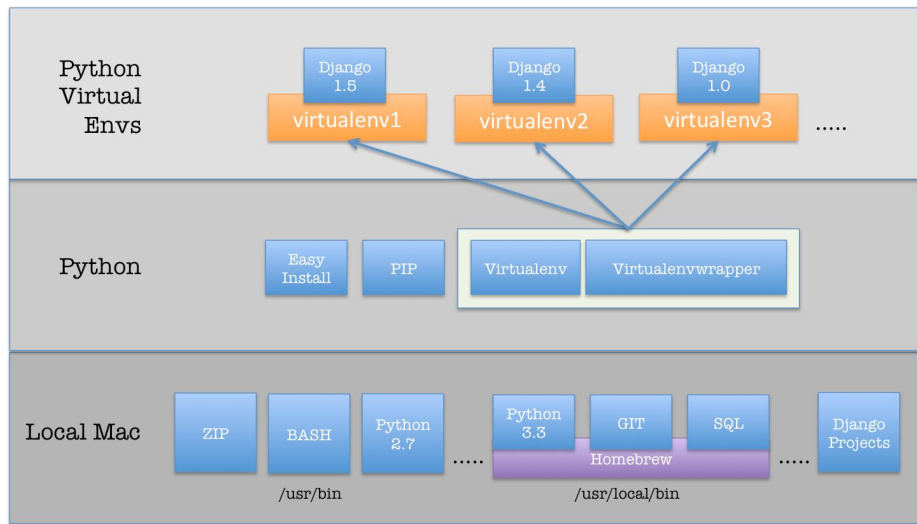
Python Package Search/Uninstall

- pip search 패키지이름
- 보통 구글에서 필요한 패키지를 검색해볼 수 있고, 위의 명령어로 패키지를 찾을 수도 있습니다.
- pip uninstall 패키지이름
- 설치했던 패키지를 삭제하는 명령어입니다.

가상 환경 설치

많은 프로젝트를 진행하다보면 각 프로젝트마다 개발 환경이 다를 수 있습니다.
이를 위해 가상 환경을 설정할 수 있습니다.

가상 환경



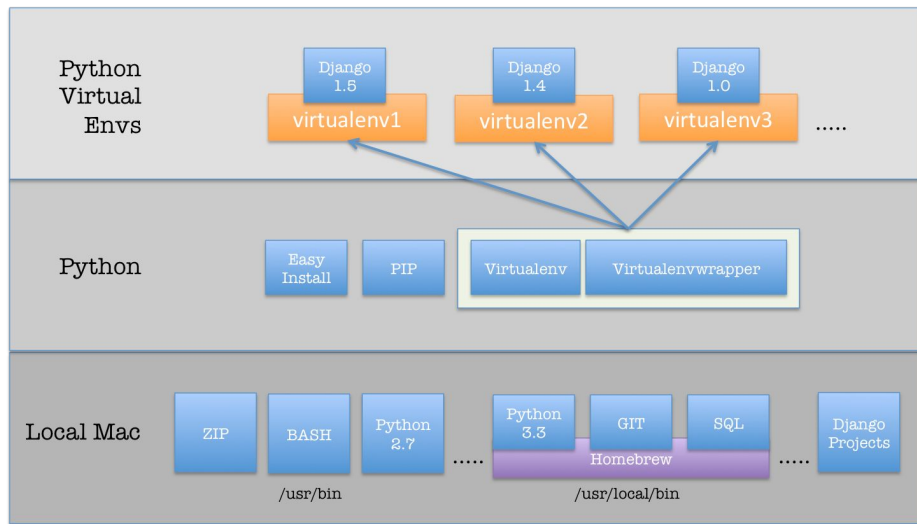
Python과 가상 환경

www.jackalventure.com/blog/python-programming-language

가상 환경

- 개발을 하다보면 다양한 환경이 생깁니다.
- 개발할 때 보통은 개발 서버에서 개발하고 테스트를 진행한 후 운영 서버로 이관을 합니다. (Development에서 Production으로 Deploy(혹은 Deployment)한다고 표현합니다.)
- 생각해볼까요?! 개발 시 필요한 다양한 패키지와 다양한 개발 변수들이 뒤죽박죽 섞여 있으면 어떨까요?
- 운영 서버로 이관을 해야하는데 뒤죽박죽 섞여서 어떤 패키지와 어떤 환경 변수를 운영으로 심어야 될지 모르는 상황이 발생할 수 있습니다.
- 그래서 가상 환경을 통해서 **독립적인 환경**을 만들어주는 것입니다.

가상 환경



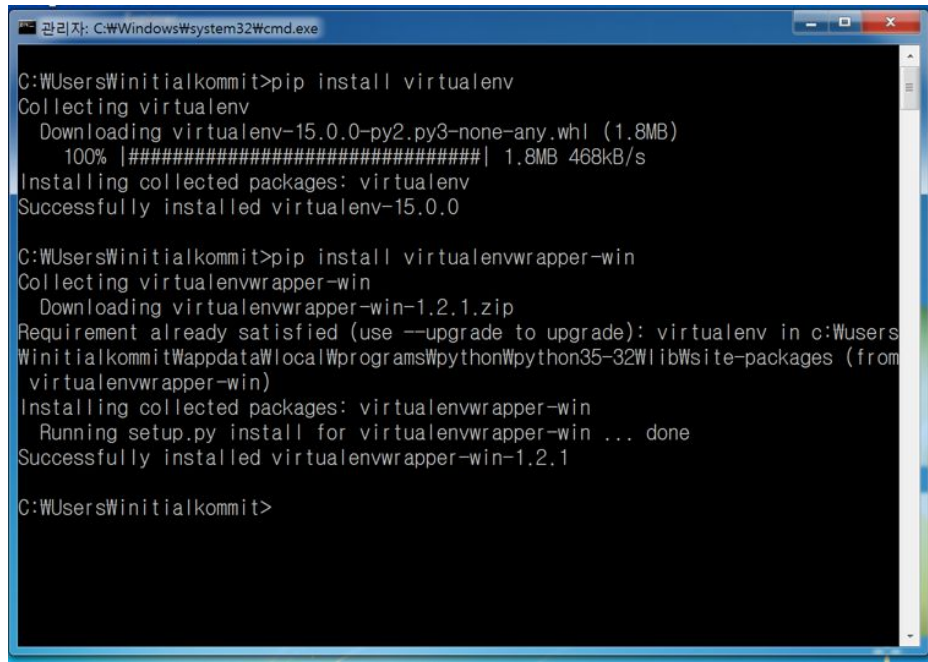
Python과 가상 환경

www.jackalventure.com/blog/tag/python-programming-language

가상 환경

- 보통은 프로젝트별로 가상환경을 만들어줍니다.
- 우리는 **virtualenv**라는 것으로 가상환경을 만들어 주도록 하겠습니다.
- 단, 이것은 가상으로 분리된 환경일 뿐이지 디스크 공간을 나눠사용한다는 개념이 아닙니다.
- 모든 **Resource**는 어떤 가상 환경이던 똑같이 사용합니다. 다만 패키지와 환경 변수만 달라질 뿐입니다.

가상 환경



```
C:\Users\Winitalkommit>pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-15.0.0-py2.py3-none-any.whl (1.8MB)
    100% |#####| 1.8MB 468kB/s
Installing collected packages: virtualenv
Successfully installed virtualenv-15.0.0

C:\Users\Winitalkommit>pip install virtualenvwrapper-win
Collecting virtualenvwrapper-win
  Downloading virtualenvwrapper-win-1.2.1.zip
Requirement already satisfied (use --upgrade to upgrade): virtualenv in c:\Users\Winitalkommit\AppData\Local\Programs\Python\Python35-32\lib\site-packages (from virtualenvwrapper-win)
Installing collected packages: virtualenvwrapper-win
  Running setup.py install for virtualenvwrapper-win ... done
Successfully installed virtualenvwrapper-win-1.2.1

C:\Users\Winitalkommit>
```

virtualenv, virtualenvwrapper-win 설치

- pip install virtualenv
- 가상 환경을 만들어주는 패키지입니다.
- pip install virtualenvwrapper-win
- virtualenv 명령어는 기본적으로 불편합니다. 이를 간편하게 사용할 수 있도록 도와주는 패키지입니다. (패키지 이름에 wrapper가 있습니다. 예쁘게 포장해는 정도로 이해하면 되겠습니다.)
- 만약 Mac이나 Linux에서 설치하고 싶다면 뒤에 '-win'을 뺀 virtualenvwrapper를 설치하면 됩니다.

가상 환경

```
관리자: C:\Windows\system32\cmd.exe
C:\Users\Winitalkommit>mkvirtualenv test
Using base prefix 'c:\Users\Winitalkommit\AppData\Local\Programs\Python\Python35-32'
New python executable in C:\Users\Winitalkommit\Envs\test\Scripts\python.exe
Installing setuptools, pip, wheel...done.

(test) C:\Users\Winitalkommit>deactivate

C:\Users\Winitalkommit>workon

Pass a name to activate one of the following virtualenvs:
=====
test

C:\Users\Winitalkommit>dir Envs
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: B8C2-C8F5

C:\Users\Winitalkommit\Envs 디렉터리

2016. 03. 11. 오후 10:43 <DIR>      .
2016. 03. 11. 오후 10:43 <DIR>      ..
2016. 03. 11. 오후 10:43 <DIR>      test
                                0 개 파일              0 바이트
                                3 개 디렉터리  56,952,193,024 바이트 남음

C:\Users\Winitalkommit>rmvirtualenv test
디렉터리가 비어 있지 않습니다.

Deleted C:\Users\Winitalkommit\Envs\test

C:\Users\Winitalkommit>folder_delete test
C:\Users\Winitalkommit>
```

가상 환경 만들기/나오기/목록보기/지우기

- mkvirtualenv 가상환경이름
- c:\Users\원도우계정이름\Envs 경로에 가상환경 파일이 자동으로 생성됩니다.
- 또한 특이한 점이 있습니다. 가상환경을 만드는 즉시 프롬프트 맨 앞에 (가상환경이름) 이라는 것이 생깁니다.
- 가상 환경을 정상적으로 만들고 들어와있다는 뜻입니다. 축하합니다.
- deactivate
- 가상 환경을 나오는 명령어입니다.
- workon
- 가상 환경 목록을 보는 명령어입니다.
- rmvirtualenv/folder_delete 가상환경이름
- 가상 환경을 지우는 명령어입니다.

가상 환경

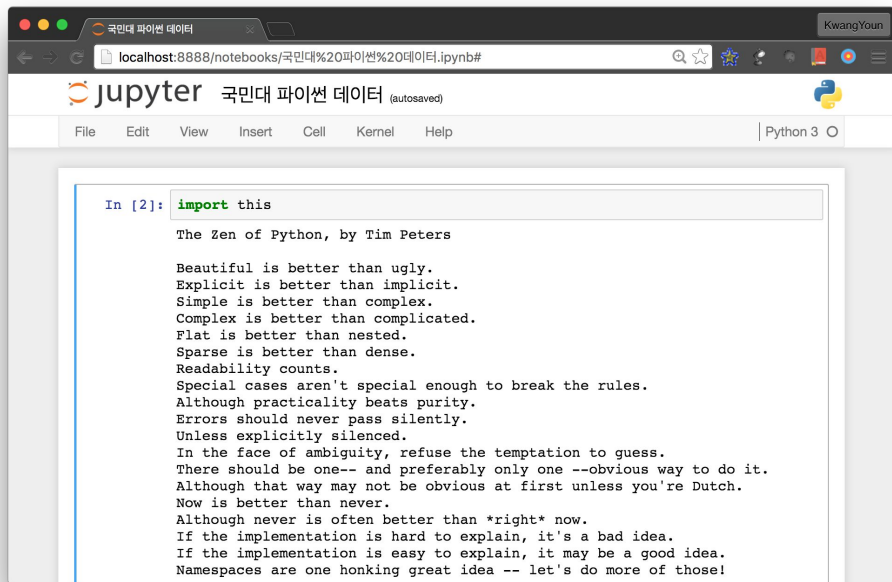
연습하기

1. 가상 환경 kookmin1을 만들어보고 가상 환경을 나와봅니다.
2. 다시 kookmin1 가상 환경으로 들어가 BeautifulSoup4라는 Package를 설치해봅니다.
3. kookmin1 가상 환경을 나와 kookmin2 가상 환경을 만듭니다.
4. kookmin2 가상 환경에서 numpy Package를 설치해봅니다.
5. kookmin1과 kookmin2 가상 환경에 각각 설치되어 있는 Package가 다른 것을 확인해봅시다.

iPython 설치

우리는 Python을 데이터 분석용으로 공부할 것입니다.
이 때 많이 쓰이는 도구인 iPython에 대해 알아보도록 하겠습니다.

iPython(Jupyter)

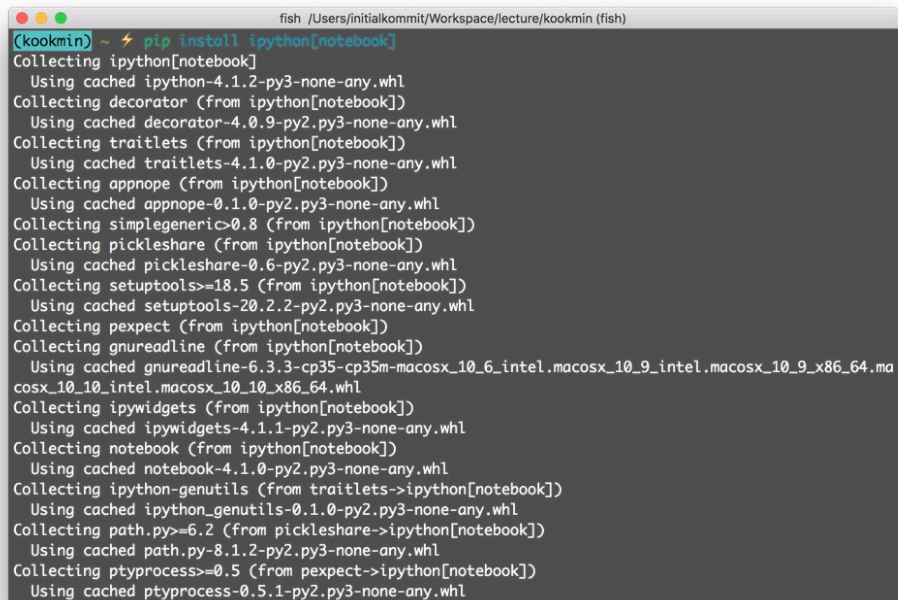


A powerful interactive shell: iPython

- 우리는 앞으로 Python을 이용해 데이터 분석을 할 때 혹은 공부를 할 때 대부분 iPython을 이용할 것입니다. 정확히 말하면 jupyter notebook을 이용할 것입니다.
- 원래는 iPython으로 시작했는데 버전이 업그레이드 되면서 Jupyter로 이름이 바뀌었습니다.
- 다양한 기능이 있어서 많은 개발자들이 스터디를 할 때 jupyter notebook으로 자료를 만들고 공유를 하곤 합니다.
- 심지어 Github에서 이 파일을 바로 볼 수 있도록 지원하고 있습니다.
- 심지어 슬라이드도 됩니다. 아래는 그 예입니다.

http://www.slideviper.oquanta.info/tutorial/slideshow_tutorial_slides.html

iPython(Jupyter)



```
fish /Users/initialkommit/Workspace/lecture/kookmin (fish)
(kookmin) ~ ✨ pip install ipython[notebook]
Collecting ipython[notebook]
  Using cached ipython-4.1.2-py3-none-any.whl
Collecting decorator (from ipython[notebook])
  Using cached decorator-4.0.9-py2.py3-none-any.whl
Collecting traitlets (from ipython[notebook])
  Using cached traitlets-4.1.0-py2.py3-none-any.whl
Collecting appnope (from ipython[notebook])
  Using cached appnope-0.1.0-py2.py3-none-any.whl
Collecting simplegeneric<=0.8 (from ipython[notebook])
Collecting pickleshare (from ipython[notebook])
  Using cached pickleshare-0.6-py2.py3-none-any.whl
Collecting setuptools>=18.5 (from ipython[notebook])
  Using cached setuptools-20.2.2-py3-none-any.whl
Collecting pexpect (from ipython[notebook])
Collecting gnu readline (from ipython[notebook])
  Using cached gnu readline-6.3.3-cp35-cp35m-macosx_10_6_intel.macosx_10_9_intel.macosx_10_9_x86_64.ma
cosx_10_10_intel.macosx_10_10_x86_64.whl
Collecting ipywidgets (from ipython[notebook])
  Using cached ipywidgets-4.1.1-py2.py3-none-any.whl
Collecting notebook (from ipython[notebook])
  Using cached notebook-4.1.0-py2.py3-none-any.whl
Collecting ipython-genutils (from traitlets->ipython[notebook])
  Using cached ipython_genutils-0.1.0-py2.py3-none-any.whl
Collecting path.py>=6.2 (from pickleshare->ipython[notebook])
  Using cached path.py-8.1.2-py2.py3-none-any.whl
Collecting ptyprocess>=0.5 (from pexpect->ipython[notebook])
  Using cached ptyprocess-0.5.1-py2.py3-none-any.whl
```

iPython(Jupyter) 설치

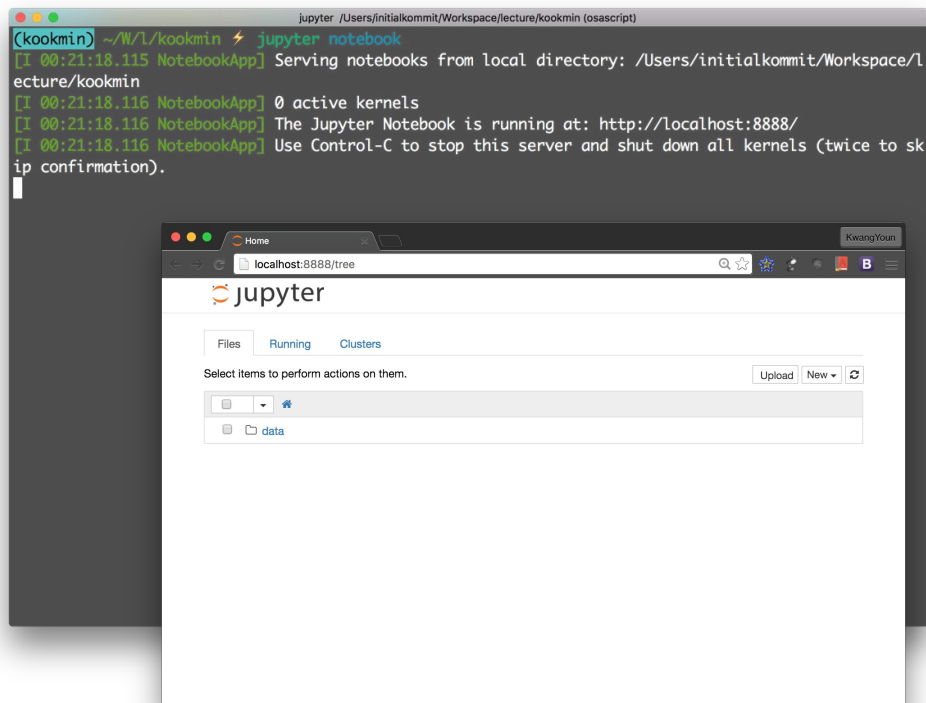
- workon 가상환경이름
- pip install ipython[notebook]

- 먼저 가상 환경에 들어가서 ipython을 설치하도록 하겠습니다.

- 가상 환경에 들어가지 않고 가상 환경이 아닌 자신의 노트북, 로컬 환경에 설치해도 전혀 상관 없습니다. 그러나 우리는 연습도 할겸 가상 환경에 들어가 만들겠습니다.

- 그런데 대괄호([])에 notebook이라는 것이 있습니다. 이는 ipython이 shell으로써뿐만 아니라 웹 브라우저에서도 볼 수 있는 notebook의 기능을 뜻합니다. 매우 강력하고 편한 기능입니다.

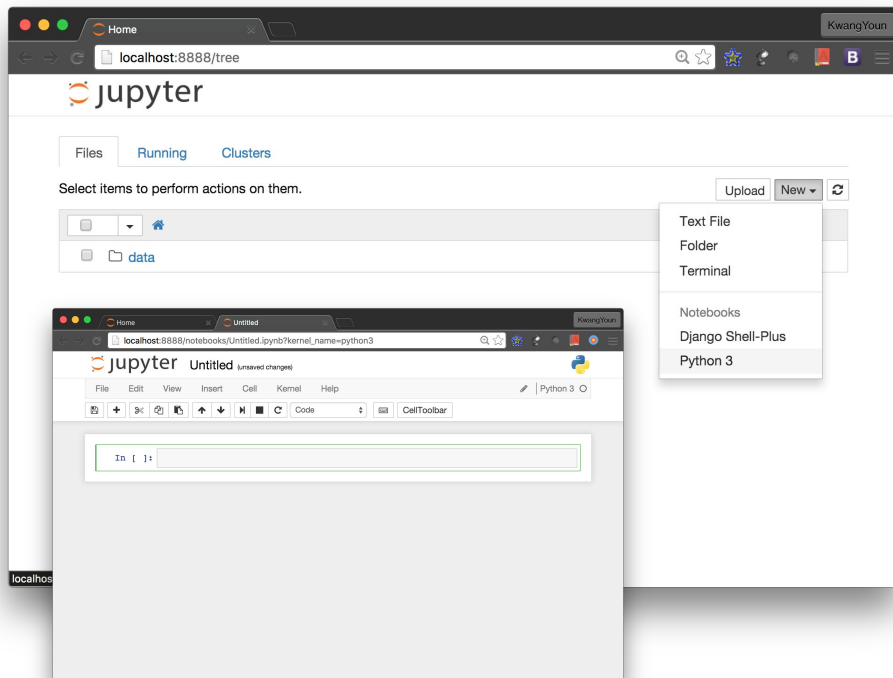
iPython



iPython(Jupyter) Notebook 실행

- jupyter notebook or ipython notebook
- 위와 같이 입력 후 실행하면 몇 가지 메시지가 뜨고 기본 브라우저가 뜨는 것을 볼 수가 있습니다.
- Jupyter Notebook는 Server-Client 방식으로 실행을 합니다. 그래서 브라우저 주소창에도 localhost:8888 라는 주소가 뜨는 것을 볼 수가 있고, 이 서버를 Ctrl-C를 눌러 멈추면 브라우저에서 실행되고 있는 Jupyter에 서버가 꺼졌다는 경고 메시지가 뜨게 됩니다.

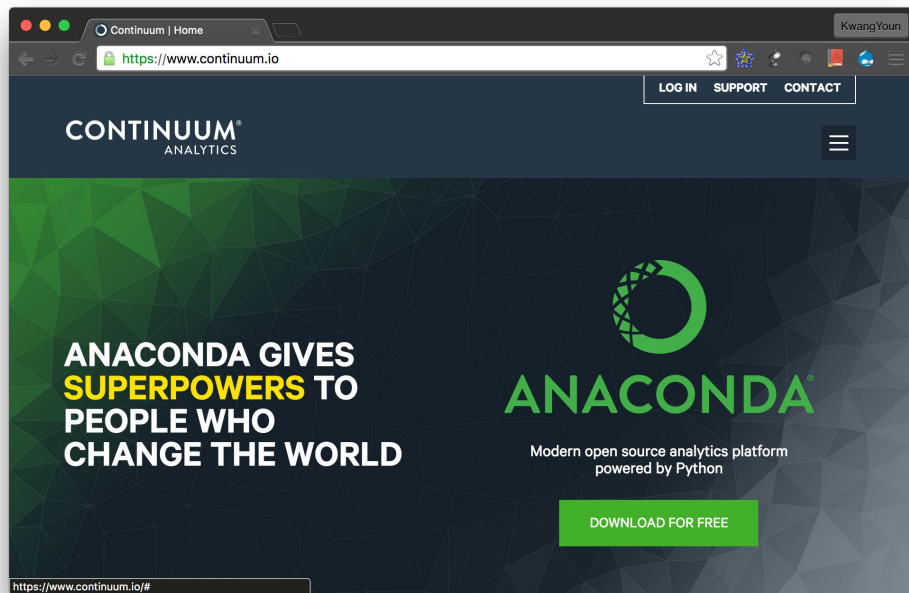
iPython



iPython(Jupyter) Notebook 실행

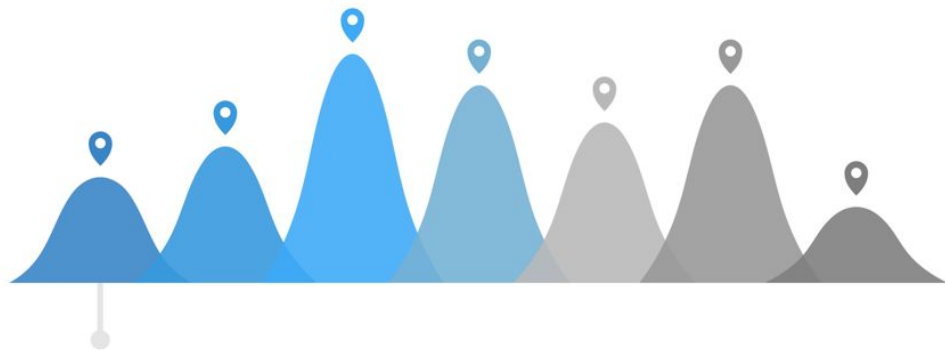
- 브라우저 우측 상단에 **New** 버튼을 클릭하면 **Python 3**을 선택하게 되어있습니다. 이를 좀 더 정확히 표현하면 커널이라고 부릅니다.
- **Python 3** 커널을 클릭하면 새로운 화면이 뜨고 이제부터 본격적으로 **Python**으로 무엇인가 분석할 수 있는 환경이 마련된 것입니다.

iPython



우리가 알아야할 또 하나 - Anaconda

- 우리는 이전에 Python을 설치하고 pip으로 패키지를 따로 설치해봤습니다. 그러나 어떤 패키지의 경우 의존성이 있을 수도 있어 의존성 패키지를 또 설치해줘야 하는 등의 불편한 점들이 있을 수 있습니다.
- 이런 불편함 등을 해결하기 위해 만들어진 것이 Anaconda라는 배포판 패키지입니다.
- Python과 약 200개가 넘는 패키지를 한 번에 설치됩니다.
- conda라는 Package Manager가 따로 있어 패키지를 쉽게 관리할 수 있습니다.
- 그러나 우리는 필요한 패키지 한 땀 한 땀 설치하며 알아가도록 하겠습니다.



개발 환경

개발 환경에 많은 시간 할애한 이유

넘어야 할 산은 많고 우리는 **Python**을 공부하러 왔는데 이제 겨우 개발 환경 하나 넘었습니다. 그러나 개발, 일 등을 하다보면 본질을 추구하고 싶어도 비본질적인 것이 발목을 잡는 경우가 많습니다. 저 또한 그랬고, 처음에 공부하시는 많은 분들이 그랬습니다. 본질에 집중하기 위해 본질을 뒷받침해주는 개발 환경에 대해 잘 알고 있을 필요가 있습니다. 개발 환경에 필히 익숙해지시길 부탁드립니다.

02

Python Primer

Kookmin University
Graduate School of Business Administration
MBA in Big Data Analytics

드디어 Python에 대해 본격적으로 알아보겠습니다. 아주 기본적인 것부터 다뤄보며 조금씩 맛을 보도록 하겠습니다. 지금부터는 슬라이드가 아니라 iPython(Jupyter) Notebook에서 살펴보도록 하겠습니다.