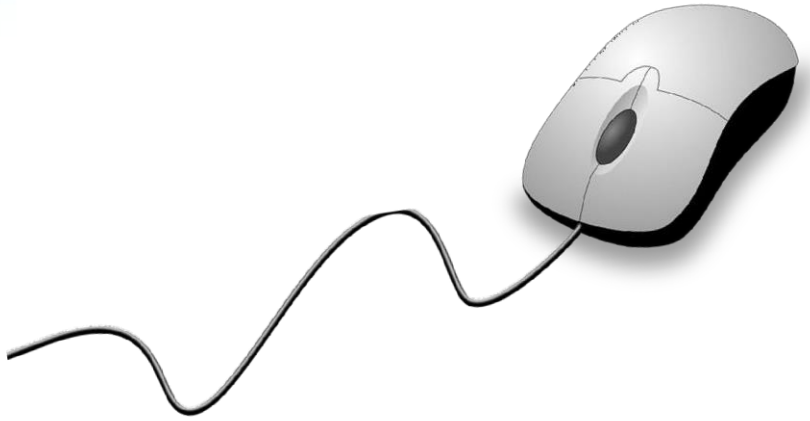
공개SW 솔루션 설치 & 활용 가이드

시스템SW > SW공학도구



python™

제대로 배워보자

How to Use Open Source Software

Open Source Software Installation & Application Guide

OPENUP

오픈소스 소프트웨어 통합지원센터

Open Source Software Support Center

CONTENTS

1. 개요

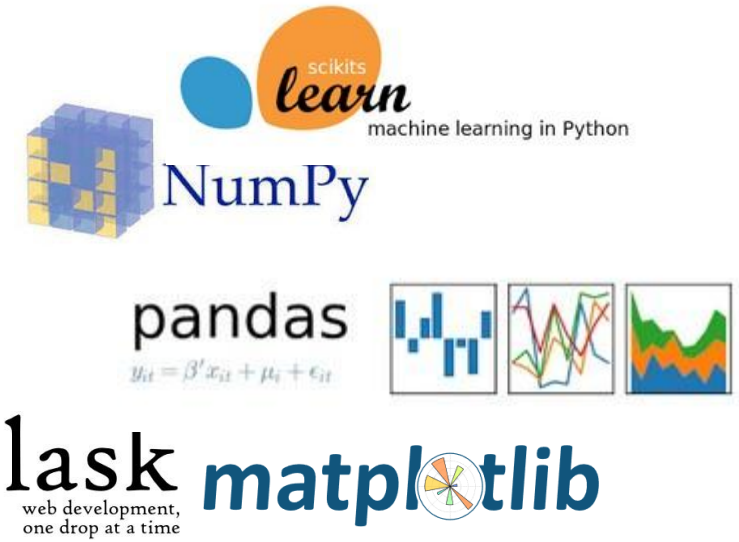
소개	• Guido van Rossum이 1991년에 처음 발표한 고수준 프로그래밍 언어 • 인터프린터 언어로 코드의 가독성 강조 • C++ 또는 Java와 같은 언어를 사용되는 것보다 적은 코드를 사용해서 표현 • 작고 큰 규모로 프로그램을 작성할 수 있도록 많은 라이브러리를 제공 • 동적 시스템 및 메모리 관리 기능을 갖추고 있으며, 객체 지향, 명령형, 함수형 프로그래밍 및 절차 스타일을 비롯한 여러 프로그래밍 패러다임 지원		
주요기능	• 수치해석, 데이터 분석 및 시각화 • 머신러닝 및 딥러닝 지원 • 웹 애플리케이션 프레임웍 지원		
대분류	• 시스템 SW	**소분류**	• SW공학도구
라이선스형태	• Python Software Foundation License	**사전설치 솔루션**	
운영제제	• Windows, macOS, Linux 등	**버전**	• Python 3 : 3.6.3 • Python 2 : 2.7.14
특징	• Dynamic typing • 객체 멤버에 접근이 용이 • 모듈, 클래스, 객체와 같은 언어의 요소가 내부에서 접근할 수 있고, 리플렉션을 이용한 기술 사용		
보안취약점	• 취약점 ID : CVE-2017-1000158 • 심각도 : 9.8 CRITICAL(V3) • 취약점 설명 : 원격 공격자는 사용자가 특수하게 조작된 파일을처리하도록유도하여임의코드를실행 • 대응방안 : 2.7.14 이상으로 업그레이드 • 참고 경로 : https://security.gentoo.org/glsa/201805-02		
개발회사/커뮤니티	• Python Software Foundation		
공식 홈페이지	• https://www.python.org		

1. 개요

- Python 2와 Python 3 선택하기
  - Python은 1990년 처음 탄생한 이후에 많은 발전하였고, 버전 1의 주요 버전인 1.5에서 버전 2로 발전하면서 많은 새로운 개념과 기능이 추가되면서도 하위 호환성을 유지하였다.
  - 1 버전의 잘못된 문제나 버리고 싶은 문제들도 호환성이라는 이름으로 유지하였지만, 버전 3은 기존 잘못되거나 비효율적인 것들을 정리하고 새롭게 시작한 버전이다.
  - 즉 버전 3부터는 하위 호환성을 유지하지 않는다.
  - Python 2 버전에서 작성된 프로그램을 3버전에서 완벽히 실행할 수 없다.
  - 2 버전도 2.7을 기준으로 더 이상 새로운 버전이 발표되지 않는다.
    - ➢ 보안 및 버그로 인하여 2.7.x 버전으로 업데이트는 진행되나, 기능상 업데이트는 없다.
  - Python 3 버전으로 버전이 계속 업데이트 되고 있다고 해서 버전 3으로만 권장할 수가 없다.
    가장 대표적인 이유가 생태 환경이다.
  - Python은 수 많은 외부 모듈과 함께 동작하는데 많은 모듈이 새로운 Python 버전을 지원하지 않았기 때문에 아직 기존의 많은 모듈이 Python 2를 기준으로 작성되어 있다.
  - 현재는 많은 모듈들이 버전 3로 이식되어 있어 본 문서는 Python 3버전으로 Python을 시작 할 것을 권장한다.

2. 기능요약

- 확장성이 뛰어나고, 유지 및 관리하기가 용이
- 언어를 학습하기 쉽고, 가독성이 우수
- 병렬 처리, 람다 함수 지원
- 다양한 기능을 제공하는 라이브러리 제공
  - ➢ 수치해석, 기계학습 지원
    - numpy, scipy, scikit-learn, pandas
  - ➢ Deep Learning 지원
    - Tensorflow, Theano, Kears, Pytorch
  - ➢ 데이터 시각화
    - matplotlib, seaborn, bokeh
  - ➢ Web Framework 지원
    - Django, Flask
  - ➢ Gui 프로그래밍 지원
    - PyQt, wxpython, PyGTK



<다양한 기능을 지원하는 라이브러리>

3. 실행환경

- 지원 OS
  - Windows 32bit / 64bit
  - MacOS 및 iOS
  - Linux (RedHat, CentOS, Debian, Ubuntu, SuSE 등)
  - BSD (FreeBSD, OpenBSD)
  - Unix (HP-UX, Solaris)
- Python 은 IDLE(Integrated DeveLopment Environment) 프로그램과 함께 사용되는 경우가 많다. 대표적인 프로그램으로는 Pycharm, Pydev(Eclipse), Spyder, Canopy 등이 있다.
- IDLE 프로그램은 무료 버전과 유료 버전이 존재하므로 기업에서 사용할 때 라이선스 확인해야 한다.

4. 설치 및 실행

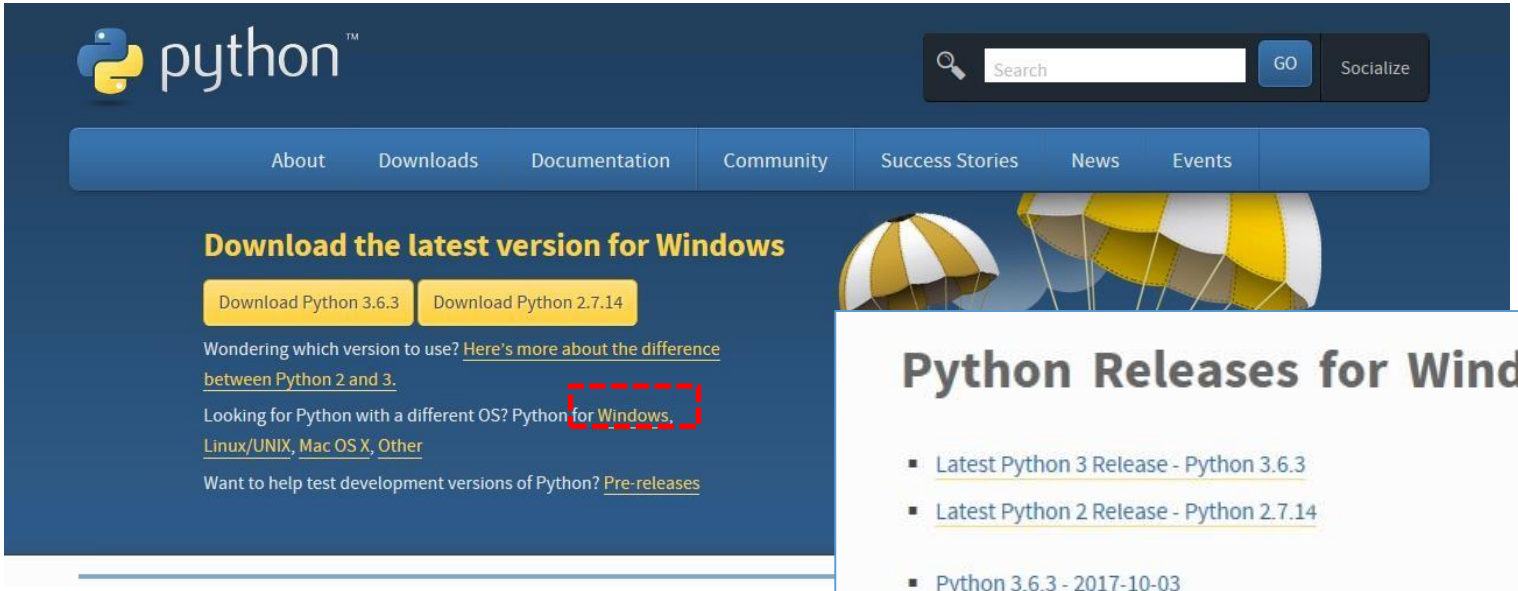
세부 목차

1. Windows 설치
    1. Python.org에서 설치
    2. Anaconda 설치
2. Linux(Ubuntu) 설치 및 초기 설정
3. Jupyter 환경 설정 및 실행

4. 설치 및 실행

4.1.1 Windows 설치(1/9)

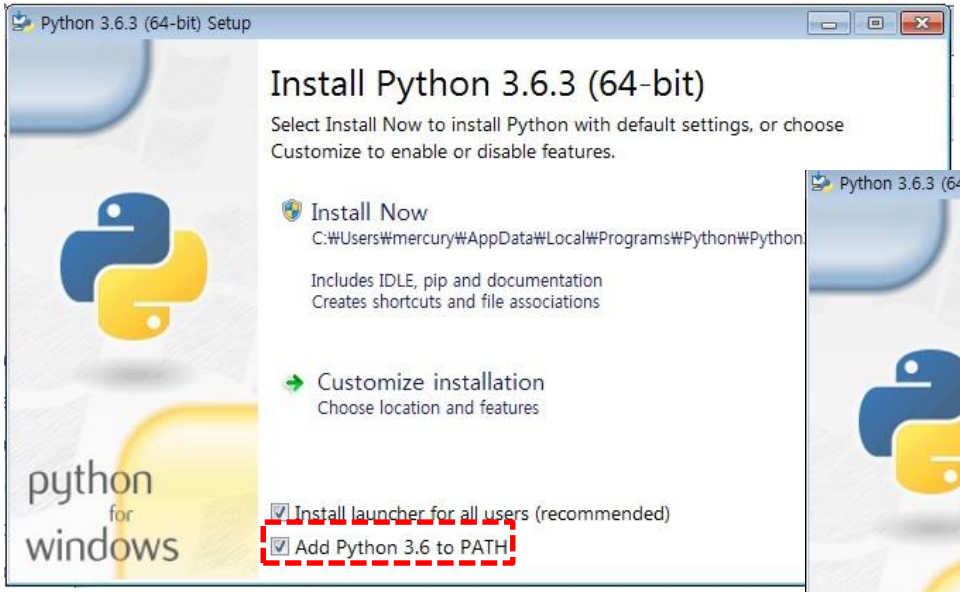
- Python 홈페이지(https://www.python.org/download)에 접속하여 Windows 링크를 클릭한다. 사용자의 컴퓨터 환경(32bit or 64bit)에 적합한 버전을 선택한다.



4. 설치 및 실행

4.1.1 Windows 설치(2/9)

- python-3.6.3-amd64(64bit용) 설치파일 실행한다.
- Add Python 3.6 to Path를 체크한다.



<Python 설치 화면 >

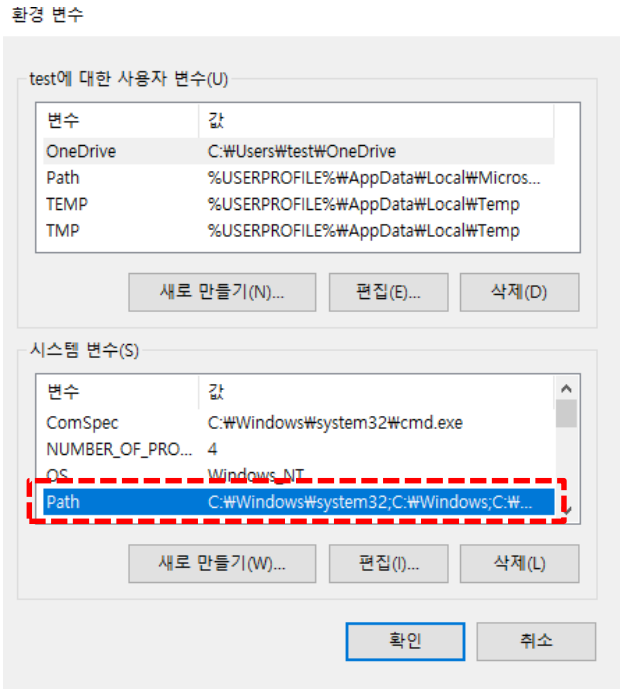
<정상적으로 설치된모습>

* 설치 시에 기본값으로 Add Python 3.6 to Path의 체크박스의 값이 체크되어 있지 않다. 해당 값을 체크하지 않으면 cmd(명령 프롬프트) 창에서 Python 관련 명령어가 실행되지 않기 때문에 체크를 권장한다.

4. 설치 및 실행

4.1.1 Windows 설치(3/9)

- Add Python 3.6 to PATH 체크하지 못 했을 경우 환경 변수의 Path을 직접 추가한다.
  - Python이 설치된 경로 추가(예시)
    - ✓ C:₩Users₩[사용자계정이름]₩AppData₩Local₩Programs₩Python₩Python36
    - ✓ C:₩Users₩[사용자계정이름]₩AppData₩Local₩Programs₩Python₩Python36₩Scripts



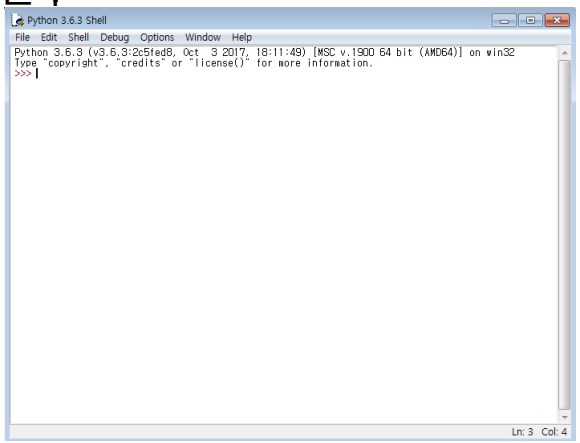
Python이 설치된 폴더와 그 하위 폴더인 Scripts 폴더가 path에 추가되어야 한다.

* Windows 환경에서 환경 변수 추가 하는 방법이 어렵게 느껴진다면 Python을 삭제 후 재설치 시에 체크를 권장
* 컴퓨터마다 설치되는 경로가 위의 예시와 같이 동일할 수는 없으니 참고 삼아 설정하시기 권장

4. 설치 및 실행

4.1.1 Windows 설치(4/9)

- 윈도우 시작버튼 ❼ Python 3.6 폴더 ❼ IDLE (Python 3.6 64-bit) 실행한다



- 명령 프롬프트 실행을 실행한 후 아래 명령어들이 정상적으로 동작해야 한다.

C:₩Windows₩system32₩cmd.exe

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:₩Users₩test>python
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()

C:₩Users₩test>pip

Usage:
  pip <command> [options]
```

python 종료 할 때 quit()를 실행

4. 설치 및 실행

4.1.1 Windows 설치(5/9)

- 패키지 설치하기
  1. 소스로 배포되는 패키지 설치
     1. 압축 소스 파일(*.tar.gz, *.tgz, *.tar.bz2) 다운로드
     2. 내려받은 파일을 압축 해제
     3. 패키지 설치
        - python setup.py install

ex) github에서 Source Code를 받아 아래 명령어를 실행하여 설치한다.

```
C:\test\requests-master>python setup.py install
running install  ru
nning bdist_egg  r
unning egg_info
.....

Installed c:\users\test\appdata\local\programs\python\python36\lib\site-packages\chardet-3.0.4-py3.6.egg
Finished processing dependencies for requests==2.18.4
```

* 패키지에 따라서는 Visual C++ Build Tools가 필요할 수 있다. 아래 url에서 다운로드 받아 설치한다.
  (http://landinghub.visualstudio.com/visual-cpp-build-tools)

4. 설치 및 실행

4.1.1 Windows 설치(6/9)

- 패키지 설치하기

2. pip를 이용하여 배포되는 패키지 설치

1. pip 사용법(cmd창에서 사용)
   - 패키지 설치
     - ✓ pip install [module]
   - 패키지 업그레이드
     - ✓ pip install [module] –upgrade
   - 패키지 삭제
     - ✓ pip uninstall [module]
   - 패키지 목록
     - ✓ pip list
   - 설치된 패키지출력
     - ✓ pip freeze
   - 최신 버전이 아닌 패키지 표시
     - ✓ pip list –outdated
   - 검색어를 포함한 패키지 목록
     - ✓ pip search [module]

2. pip를 사용한 whl 파일 설치
   - 패키지 설치
     - ✓ pip install (module.whl)
     - ✓ pip install (url주소)

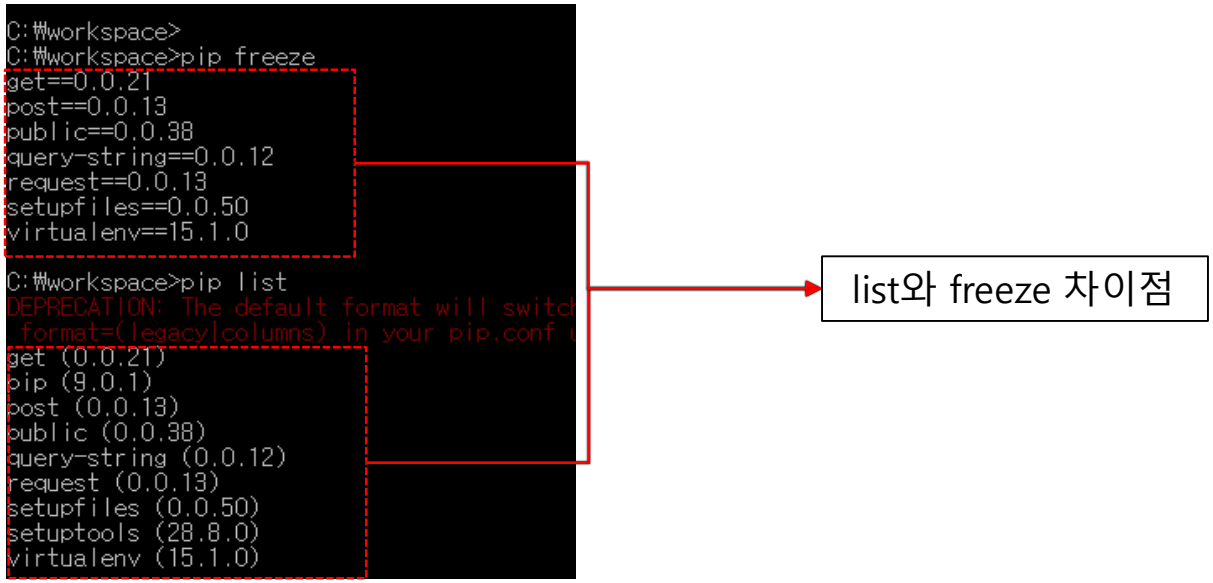
```
C:₩temp>pip install request
Collecting request
  Downloading request-0.0.13.tar.gz
Collecting get (from request)
  Downloading get-0.0.21.tar.gz
Collecting post (from request)
  Retrying (Retry(total=4, connect=None, read=None, redirect=None)) after connection broken by 'ReadTimeoutError("HTTPS
onnectionPool(host='pypi.python.org', port=443): Read timed out. (read timeout=15)",)': /packages/37/d0/0958a06b98d9f4e
c74862daa71b1657f999cfca18326c3bc9db9abdf2da/post-0.0.13.tar.gz
  Downloading post-0.0.13.tar.gz
Collecting setupfiles (from request)
  Downloading setupfiles-0.0.50.tar.gz
Collecting query_string (from get->request)
  Downloading query_string-0.0.12.tar.gz
Collecting public (from query_string->get->request)
  Downloading public-0.0.38.tar.gz
Installing collected packages: setupfiles, public, query-string, get, post, request
  Running setup.py install for setupfiles ... done
  Running setup.py install for public ... done
  Running setup.py install for query-string ... done
  Running setup.py install for get ... done
  Running setup.py install for post ... done
  Running setup.py install for request ... done
Successfully installed get-0.0.21 post-0.0.13 public-0.0.38 query-string-0.0.12 request-0.0.13 setupfiles-0.0.50

C:₩temp>
```

<request 패키지 설치 예시>

4. 설치 및 실행

4.1.1 Windows 설치(7/9)

- list와 freeze의 차이점 및 설치된 패키지 목록 백업하기
  - list와 freeze차이점
    - ✓ list는 설치된 패키지 목록을 출력한다.
    - ✓ Freeze는 설치된 패키지 목록을 백업하여 저장할 수 있다.
      - 백업된 파일을 사용하여 새롭게 설치된 환경에서 일괄로 설치 가능하다.



  - freeze를 사용하여 패키지 목록 백업하기
    - ✓ pip freeze > requirements.txt
  - 목록으로 부터 패키지 일괄 설치하기
    - ✓ pip install -r requirements.txt

4. 설치 및 실행

4.1.1 Windows 설치(8/9)

- 가상환경 설정하기

1. virtualenv를 사용하여 가상환경 만들기
   - Python으로 프로젝트 및 실습을 진행하다 보면 실행 환경에 따라서 패키지 버전을 상이하게 설치해야 경우가 있다. 이럴 경우에 사용하는 툴이 virtualenv 이다.
   - virtualenv는 독립된 Python 환경을 만들어 주는 툴이다. Python 프로젝트에 필요한 패키지를 사용하기 위해 필요한 모든 파일을 포함하는 별도의 폴더를 생성한다.

1.1 virtualenv 설치하기

ex) pip install virtualenv

```
C:₩temp>pip install virtualenv
Collecting virtualenv
  Using cached virtualenv-15.1.0-py2.py3-none-any.whl
Installing collected packages: virtualenv
Successfully installed virtualenv-15.1.0
```

2. virtualenv 환경 만들기
   - 예) workspace 폴더 안에서 virtualenv를 사용하여 myid라는 가상환경 생성

```
C:₩workspace>virtualenv myid
Using base prefix 'c:₩₩users₩₩gen2₩₩appdata₩₩local₩₩programs₩₩python₩₩python36'
New python executable in C:₩workspace₩myid₩Scripts₩python.exe
Installing setuptools, pip, wheel...done.
```

4. 설치 및 실행

4.1.1 Windows 설치(9/10)

- 가상환경 설정하기

1.3 가상환경 실행하기

방법 1 : call [가상환경이름/Scripts/activate]

ex) call ./myid/Scripts/activate

가상환경이 정상적으로 실행이 되면 command 앞에 가상계정 이름이 출력

```
C:\workspace> call ./myid/Scripts/activate
(myid) C:\workspace>
```

방법 2 : 가상환경으로 생성된 폴더의 Scripts 안의 activate를 실행

ex) activate

```
C:\workspace\myid\Scripts> activate
(myid) C:\workspace\myid\Scripts>
```

```
test@ubuntu:~$ source ./myid/bin/activate
(myid) test@ubuntu:~$
```

ubuntu에서 사용 방법

4. 가상환경에서 나오기

✓ 가상환경을 해제시에는 deactivate를 실행 (ubuntu 동일)

```
C:\workspace\myid\Scripts> deactivate
C:\workspace\myid\Scripts>
```

tip) freeze에서 백업한 패키지 목록 일괄 설치하기

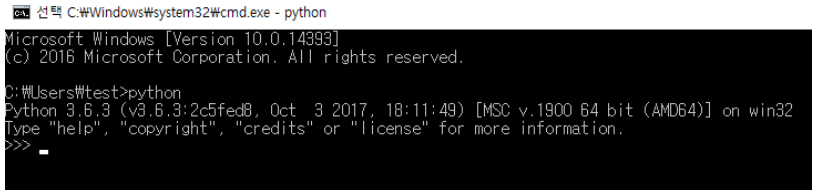
```
(myid) C:\workspace> pip install –r requirements.txt
Collecting get==0.0.21 (from –r requirements.txt (line 1))
```

4. 설치 및 실행

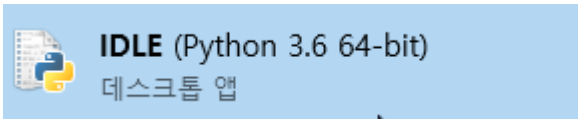
4.1.1 Windows 설치(10/10)

- Python 실행하기

1. 콘솔에서 Python 실행하기
   1. 시작 버튼 -> cmd -> python 실행

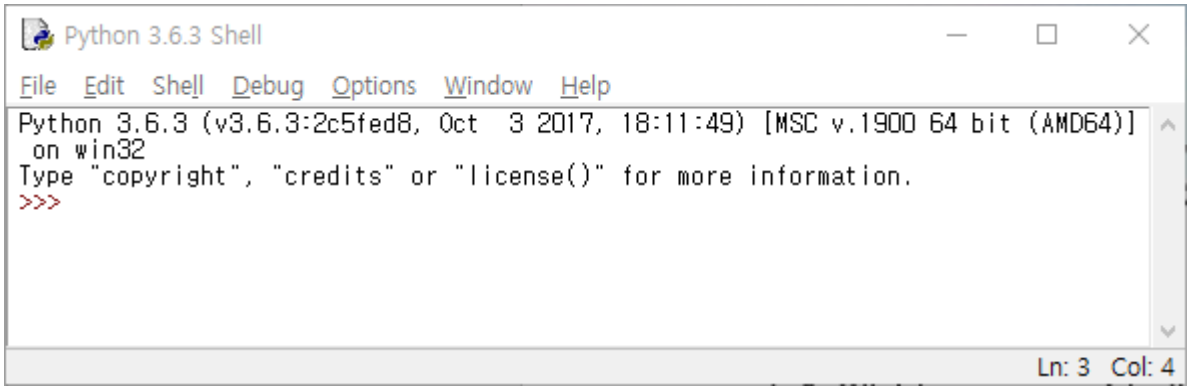


2. Python IDLE 환경 실행하기
   ✓ 기본적인 Python IDLE를 실행



Tip

- Python이 설치된 폴더안에 파일을 실행
  [Python 설치폴더]\Lib\idlelib\idle.pyw C:
\> python C:\python36\Lib\idlelib\idle.pyw

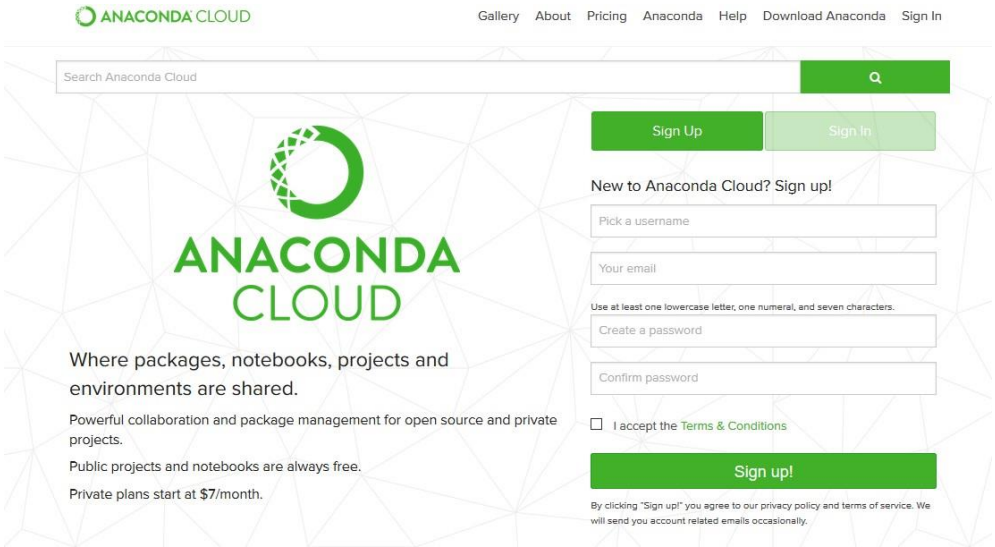


스크립트 작성시에는 File -> New File를 실행하여 코드를 작성한다.

4. 설치 및 실행

4.1.2 Anaconda 설치(1/7)

- Anaconda 설치하기
  - ➢ Anaconda는 패키지 관리 및 배포를 단순화하는 것을 목표로하는 대규모 데이터 처리, 예측 분석 및 과학 컴퓨팅을 위한 Python 및 R 프로그래밍 언어의 배포판이다.
  - ➢ Anaconda를 설치하면 데이터 분석 및 처리하는 패키지가 기본 내장되어 있어 사용자가 개별적으로 설치할 필요가 없어진다.
    - ✓ 분석 및 수집에 필요한 720개 넘는 패키지가 설치되어 있다.
  - ➢ Python의 용도가 분석 및 기계학습이라면 Anaconda 설치를 권장한다.

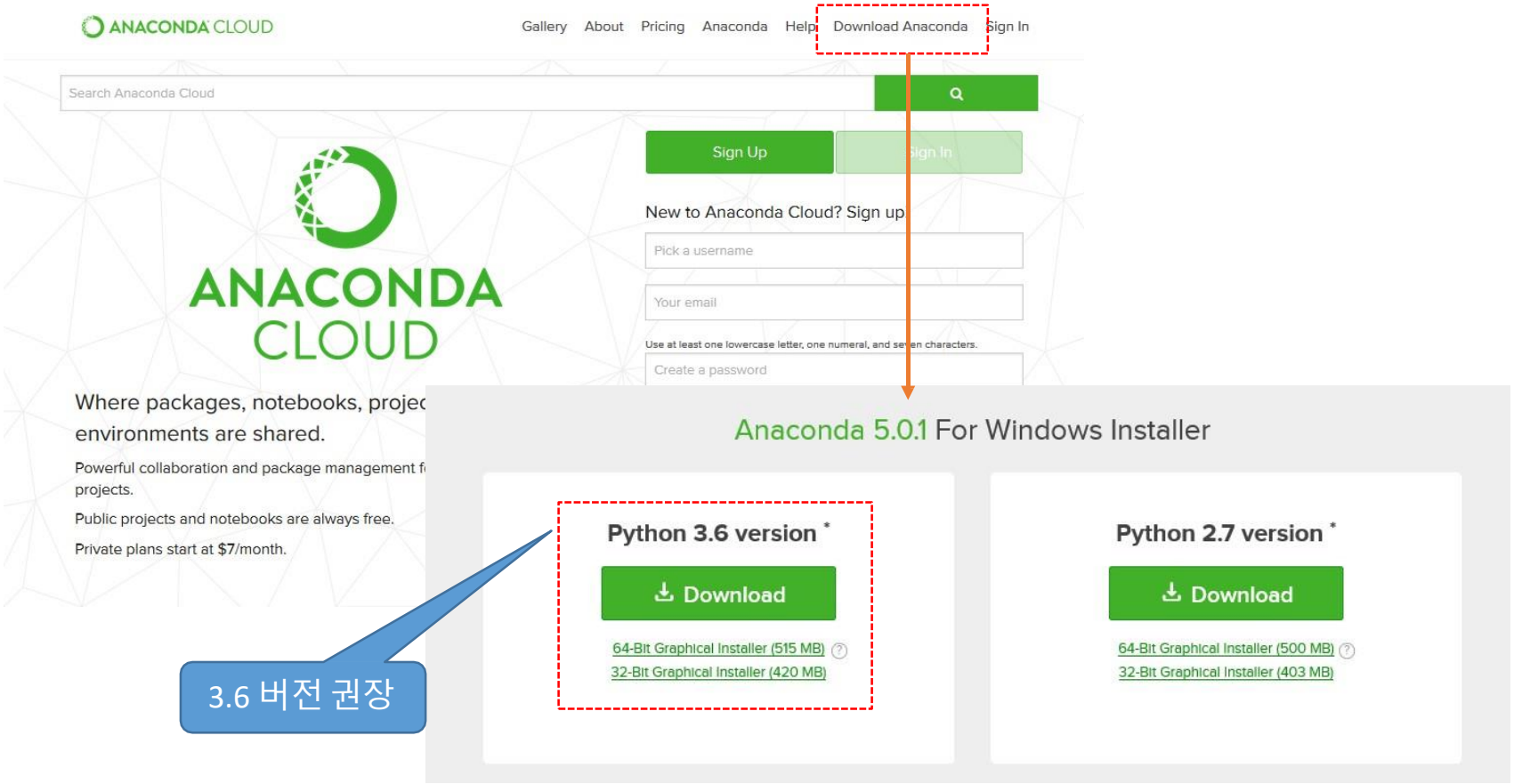


<https://www.anaconda.com>

4. 설치 및 실행

4.1.2 Anaconda 설치(2/7)

- Anaconda 설치하기
  - https://www.anaconda.com

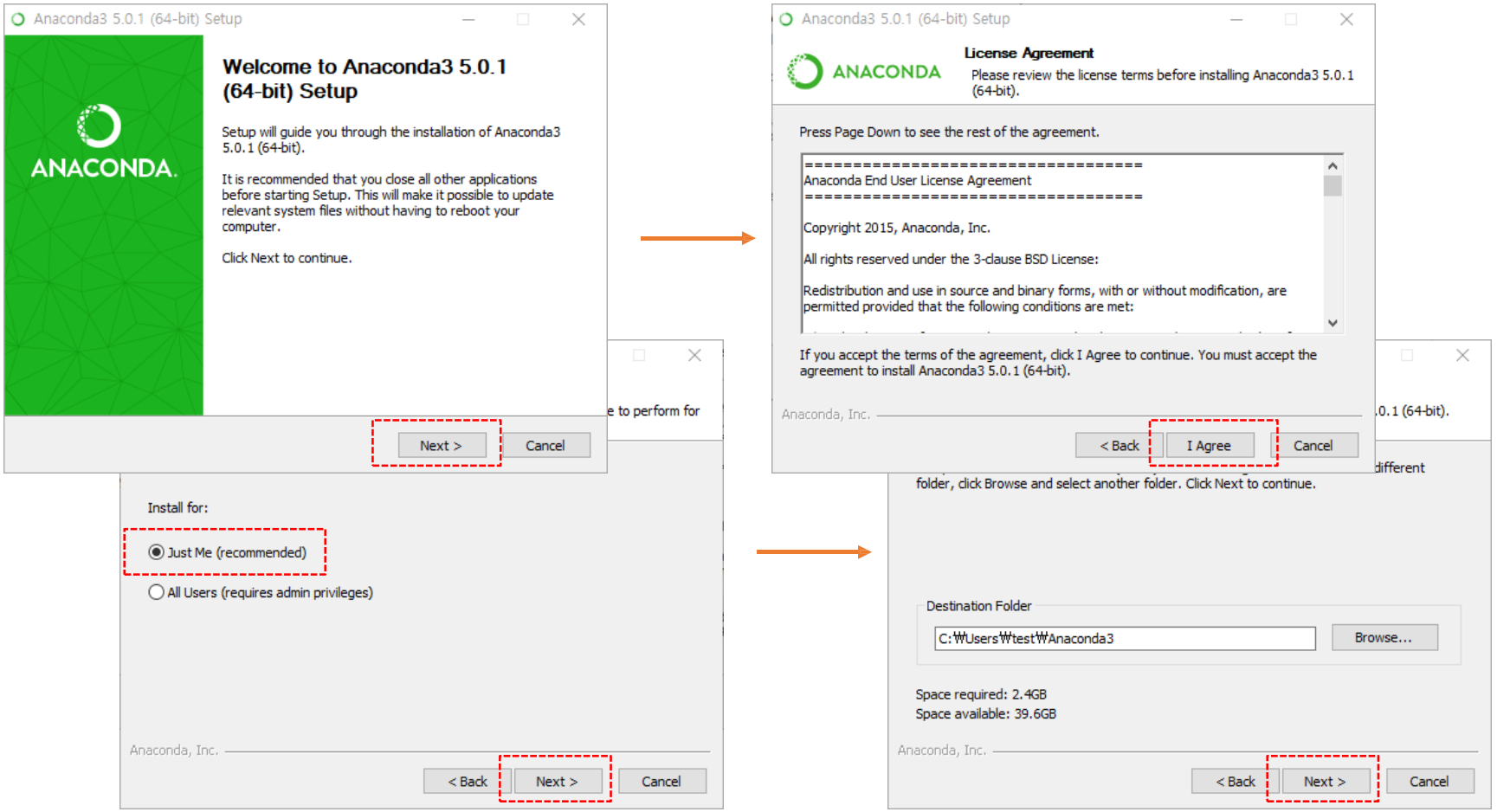


<ANACONDA 설치파일>

4. 설치 및 실행

4.1.2 Anaconda 설치(3/7)

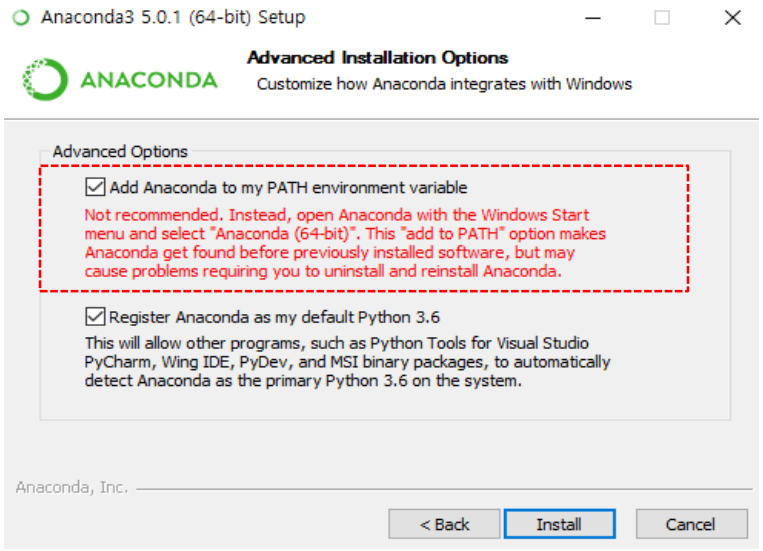
- Anaconda 설치하기

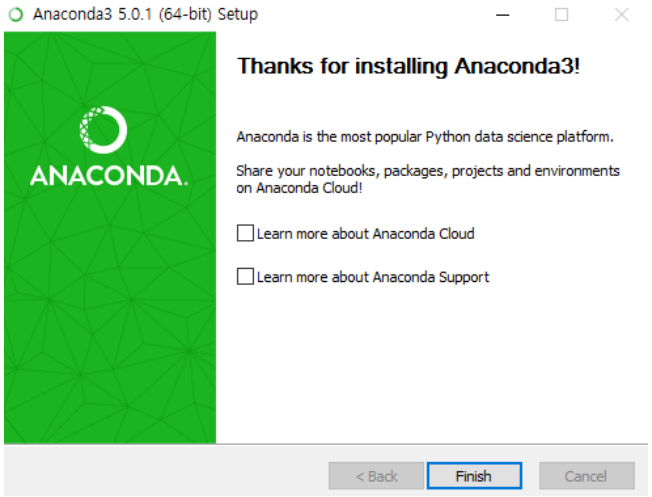


4. 설치 및 실행

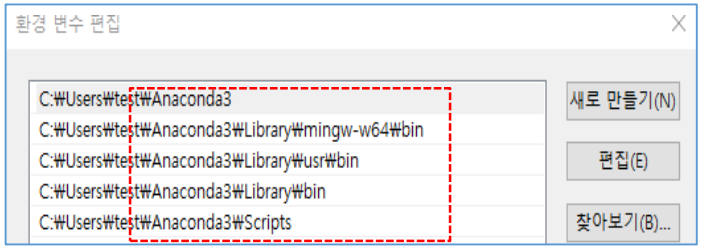
4.1.2 Anaconda 설치(4/7)

- Anaconda 설치하기
  - 아래 체크박스를 체크하여 자동으로 path 설정이 된다.





- 설치가 완료되면 Finish 버튼을 눌러 종료한다.
- Python 설치시 처럼 PATH를 체크하지 않으면 사용자가 직접 path 설정을 진행해서 사용해야 한다.
- 필자 컴퓨터의 Anaconda 환경변수



4. 설치 및 실행

4.1.2 Anaconda 설치(5/7)

- conda 사용법
  - Anaconda에는 Conda라는 패키지 관리 및 환경을 관리 할 수 있는 프로그램이 있다.
  - 사용자가 다른 버전의 패키지 및 관련된 모든 필수 라이브러리를 설치할 수 있도록 한다.
  - 콘솔창(cmd)에서 conda를 실행하면 된다. beautifulsoup4라는 패키지를 예를 들어 설명하겠다. beautifulsoup4 위치에 원하는 패키지 명을 적으시면 된다.

- conda –version : conda 버전 확인하기
- conda list : 현재 환경에서 설치된 패키지 및 버전 확인하기
- conda search beautifulsoup4 : 특정 패키지를 conda로 설치 가능한지 확인하기
- conda install beautifulsoup4 : 패키지 설치하기
- conda remove beautifulsoup4 : 패키지 삭제하기
- conda update beautifulsoup4 : 최신버전으로 패키지 업데이트 하기
- conda update –all : 모든 패키지를 업데이트 하기
- conda update conda : conda 자체를 업데이틑 하기

4. 설치 및 실행

4.1.2 Anaconda 설치(6/7)

- conda 가상환경 만들기
  - Anaconda에는 virtualenv처럼 가상 환경을 conda로 생성할 수 있다.
  - 콘솔창(cmd)에서 conda를 실행한다.

예제) 'test'라는 가상 환경 사용자 만들기

- conda create --name test python=3.6 jupyter spyder
  - ✓ test 뒤에 python 버전 설정 및 설치할 패키지 목록을 적어주면 가상환경 생성할 때 자동으로 설치가 된다.
  - ✓ 위의 예제는 python 3.6버전과 jupyter와 spyder라는 패키지를 설치하라는 의미다.

가상 사용자 만들기 (다른 계정 사용자 환경 복사 - root 사용자 환경 복사)

- conda create --name test --clone root

가상 환경 지우기

- conda env remove –n test

가상 환경 실행하기

- activate test

가상 환경 끝내기

- deactivate test

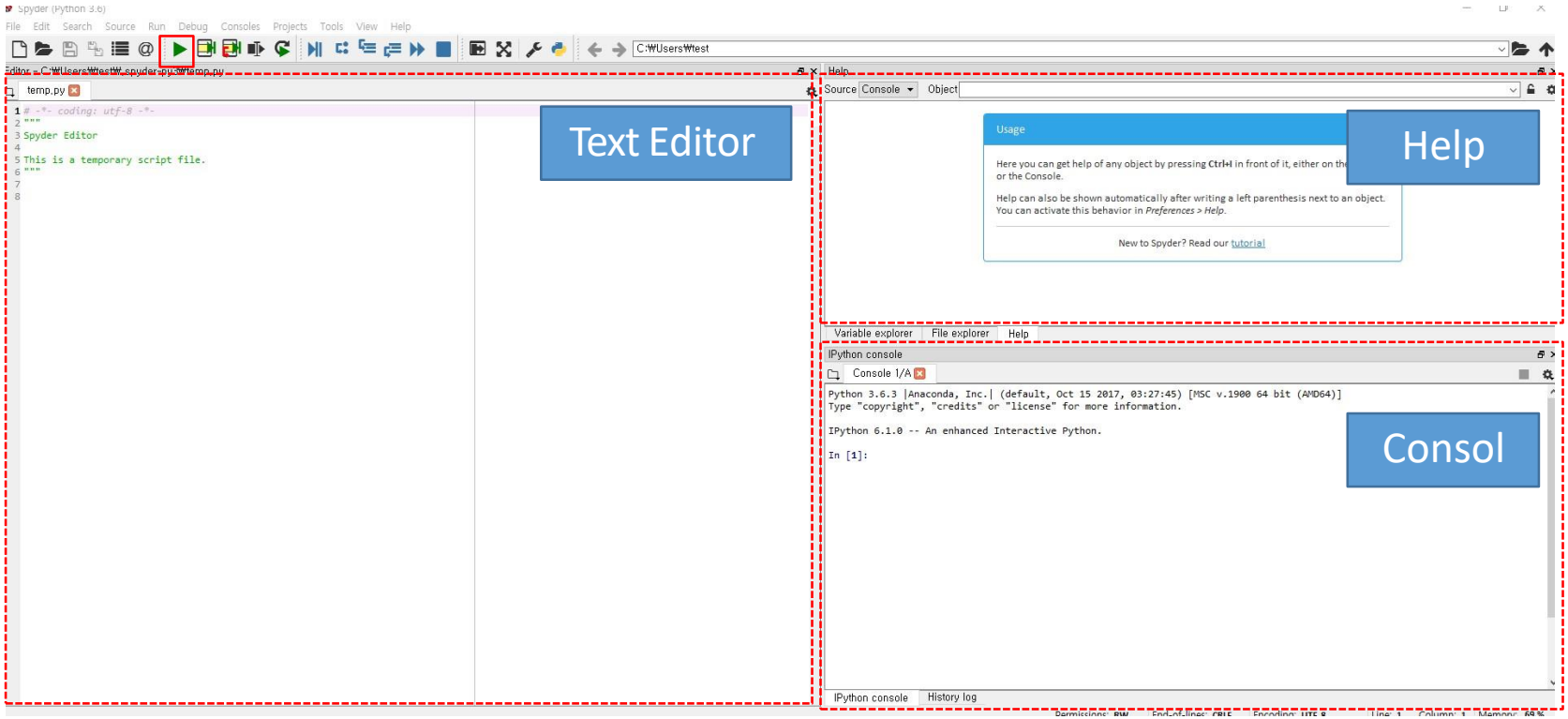
가상 환경 리스트 확인하기

- conda env list

4. 설치 및 실행

4.1.2 Anaconda 설치(7/7)

- IDLE 프로그램 실행하기
  - Anaconda는 Spyder라는 IDLE 프로그램이 내장되어 있다.
  - 시작 프로그램 메뉴의 Spyder를 실행하거나 cmd창에서 spyder라고 실행한다.



  - Text Editor에 Python의 코드를 작성 후에 Play 버튼을 눌러 코드를 전체 실행

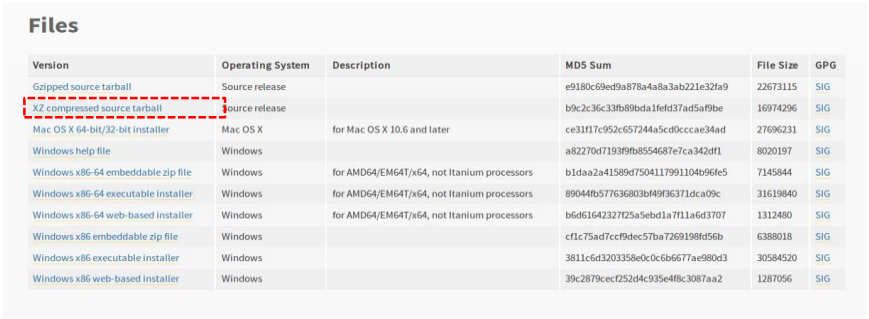
4. 설치 및 실행

4.2.1 Linux(Ubuntu) 설치(1/3)

- Ubuntu 16.04.3 버전을 기준으로 설치 방법에 대해서 설명한다.
- Ubuntu를 기본 설치할 경우 Python이 설치 되어있다.

```
gen2@ubuntu:~$ python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>

gen2@ubuntu:~$ python3
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- 최신버전의 Python을 설치하는 방법에 대해서 2가지 방법으로 설명한다.
- Python 홈페이지(https://www.python.org/download)에 접속하여 XZ compressed source tarball Version을 다운로드 받는다.



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		e9180c69ed9a878a4a8a3ab221e32fa9	22673115	SIG
XZ compressed source tarball	Source release		b9c2c36c33fb89bda1fefd37ad5af9be	16974296	SIG
Mac OS X 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	ce31f17c952c657244a5cd0cccae34ad	27696231	SIG
Windows help file	Windows		a82270d7193f9fb8554687e7ca342df1	8020197	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64, not Itanium processors	b1daa2a41589d7504117991104b96fe5	7145844	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64, not Itanium processors	89044fb577636803bf49f36371dca09c	31619840	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64, not Itanium processors	b6d61642327f25a5ebd1a7f11a6d3707	1312480	SIG
Windows x86 embeddable zip file	Windows		cf1c75ad7ccf9dec57ba7269198fd56b	6388018	SIG
Windows x86 executable installer	Windows		3811c6d3203358e0c0c6b6677ae980d3	30584520	SIG
Windows x86 web-based installer	Windows		39c2879cecf252d4c935e4f8c3087aa2	1287056	SIG

* Python-3.6.3.tar.xz 파일이 저장된다.

4. 설치 및 실행

4.2.1 Linux(Ubuntu) 설치 방법 1번째(2/3)

1. 다운로드 받은 압축파일의 압축을 해제한다.

```
tar –xvf Python-3.6.3.tar.xz
```

2. 압축해제한 폴더로 이동하여 configure 스크립트를 실행한다.

```
cd Python-3.6.3
./configure
```

3. sudo make를 실행한다.

```
sudo make
```

4. 마지막으로 sudo make install를 실행하여 설치를 완료한다.

```
sudo make install
```

- 4번째 명령어를 실행하고 다음과 같은 오류가 발생한다면 아래 패키지를 설치한다.

```
zipimport.ZipImportError: can’t decompress data; zlib not available
Makefile:1079: recipe for target ‘install’ failed
```

- make: *** [install] Error 1

```
sudo apt-get install zlib1g-dev
```

5. Python3를 실행한다.

```
test@ubuntu:~$ python3
Python 3.6.3 (default, Oct 6 2017, 08:44:35)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

4. 설치 및 실행

4.2.1 Linux(Ubuntu) 설치 방법 2번째(apt-get 사용)(3/3)

- 홈페이지에서 다운로드 받지 않고 apt-get를 이용하여 설치하는 방법이다.
- Ubuntu apt-get으로 Python3.6 버전을 설치한다. 아래와 같이 명령어를 실행하여 설치한다.

```
sudo add-apt-repository ppa:jonathonf/python-3.6
sudo apt-get update
sudo apt-get install python3.6
```

- 설치가 완료되면 python3.6를 실행한다.

```
test@ubuntu:~$ python3.6
Python 3.6.3 (default, Oct 6 2017, 08:44:35)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- Ubuntu의 apt-get의 기능을 통해서 설치를 하는 것을 처음 시작하는 사용자들에게 추천한다.
- 가상환경 설정
  - ➢ Windows 설치 방법 참조

* PPA(Personal Package Archive)
런치패드에서 제공하는 우분투의 공식 패키지 저장소에 없는 서드파티 소프트웨어를 위한 개인용 소프트웨어 패키지 저장소다. 단순히 소프트웨어의 패키지를 저장하는 것뿐만이 아닌 해당 소프트웨어의 업데이트 기능도 제공한다.

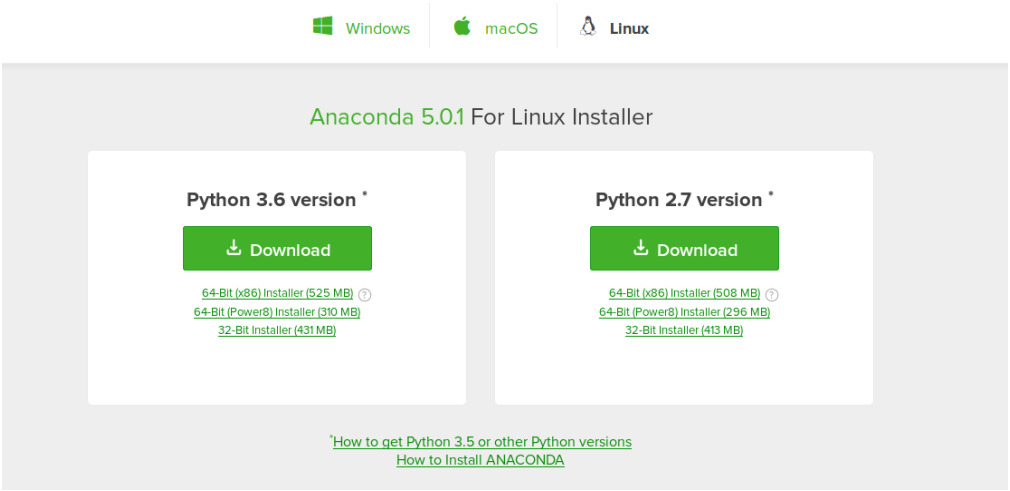
* 런치패드(Launchpad)
사용자가 소프트웨어를 개발할 때 사용하는 웹사이트 혹은 웹 애플리케이션

출처: 위키피디아

4. 설치 및 실행

4.2.2 Ubuntu에 Anaconda 설치(1/3)

- Windows 경우처럼 Anaconda 설치에 대해서 알아보자.
- anaconda 홈페이지에서 설치할 컴퓨터 환경에 맞는 버전을 다운로드 받는다.



1. 다운로드 한 폴더로 이동한다.

```
test@ubuntu:~$ cd ~/Downloads
test@ubuntu:~/Downloads$ ls  An
aconda3-5.0.1-Linux-x86_64.sh
```

2. 다운로드 받은 파일을 실행 권한 부여하고 실행한다.

```
test@ubuntu:~/Downloads$ chmod +x ./Anaconda3-5.0.1-Linux-x86_64.sh
test@ubuntu:~/Downloads$ ./ Anaconda3-5.0.1-Linux-x86_64.sh
```

4. 설치 및 실행

4.2.2 Ubuntu에 Anaconda 설치(2/3)

3. Enter 버튼을 눌러 다음을 진행한다.

```
Welcome to Anaconda3 5.0.1

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

4. License 문장이 정상적으로 출력될 것이다.

```
Anaconda End User License Agreement
==================================

Copyright 2015, Anaconda, Inc.

All rights reserved under the 3-clause BSD License:

Redistribution and use in source and binary forms, with or without modification,
 are permitted provided that the following conditions are met:
.....

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WA
RRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
--More--
```

5. q버튼을 누르고 동의서에서 나옵니다. 그 뒤에 yes를 입력하여 동의하고 진행한다.

```
Do you accept the license terms? [yes|no]
[no] >>> yes
```

출처: 위키피디아

4. 설치 및 실행

4.2.2 Ubuntu에 Anaconda 설치(3/3)

6. 설치할 위치를 선택한다. 기본 위치로 설치하겠다. 그 뒤 바로 설치되는 모습을 볼 수 있다.

```
Anaconda3 will now be installed into this location:
/home/gen2/anaconda3

  - Press ENTER to confirm the location
  - Press CTRL-C to abort the installation
  - Or specify a different location below

[/home/gen2/anaconda3] >>>
```

7. bashrc에 PATH를 추가할 것인지 확인한다. yes를 입력한다.

```
Do you wish the installer to prepend the Anaconda3 install location
to PATH in your /home/gen2/.bashrc ? [yes|no]
[no] >>> yes
```

* .bashrc 파일을 확인하면 아래와 같이 경로가 추가된 것을 확인할 수 있다. 위에서 no라고 하였다면 PATH 설정이 되지 않는다.

```
# added by Anaconda3 installer
export PATH="/home/gen2/anaconda3/bin:$PATH"
```

8. bashrc 설정 파일을 적용하여 anaconda가 정상적으로 적용 되었는지 확인한다.

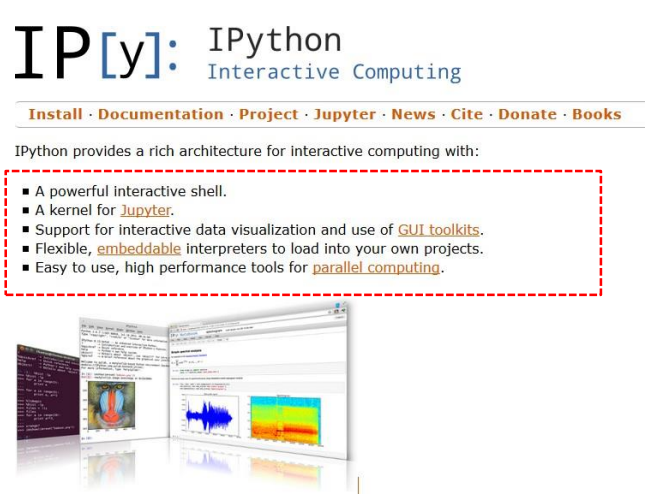
```
test@ubuntu:~/Downloads$ source ~/.bashrc
test@ubuntu:~/Downloads$ python
Python 3.6.3 |Anaconda, Inc.| (default, Oct 13 2017, 12:02:49)
[GCC 7.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()
```

* Spyder도 정상적으로 실행이 되어야 한다.

4. 설치 및 실행

4.3 IPython 및 Jupyter 실행하기(1/4)

- IPython은 2001년에 Fernando Perez가 파이썬을 과학 플랫폼에 쉽게 사용할 수 있도록 커맨드라인 인터페이스로 개발했다.
- IPython은 대화형 Notebook을 2011년에 발표하였고, 웹 브라우저에서 실행되며 코드, 텍스트, 수학식, 도표, 그래프, 대화형 그래픽 콘트롤러 등과 같은 인터페이스를 제공한다.
- Notebook은 Python이외의 언어(R, Julia, Ruby)에서도 사용되는 인터페이스가 되었고, 과학 분야 뿐만 아니라 교육, 소프트웨어 문서, 책 집필등에도 널리 사용되게 되었다.
- 2014년에 Jupyter라는 새로운 프로젝트로, Notebook을 일반 목적으로 사용할 수 있도록 확장하였다.
- Jupyter는 언어에 독립적으로 다양한 커널과 연동된다. 이제는 IPython은 Jupyter Notebook의 Python 커널 이름이다. (Jupyter Notebook의 R 커널의 이름은 IR, Julia는 IJulia등으로 불린다.)



- 강력한 대화형 Shell 제공
- Jupyter의 커널
- 대화식 데이터 시각화 및 GUI 툴킷 지원
- 프로젝트에 IPython 인터프린터를 포함하기 유연함
- 병렬 컴퓨팅에 사용하기 쉬운 고성능 도구

4. 설치 및 실행

4.3 IPython 및 Jupyter 실행하기(2/4)

- Ipython 설치하기
  - ✓ Anaconda를 설치했다면 Ipython은 기본적으로 설치되어 있다.
  - ✓ Python.org에서 파일을 다운받아서 설치하신 분은 아래처럼 Ipython과 Jupyter를 설치해야 한다.

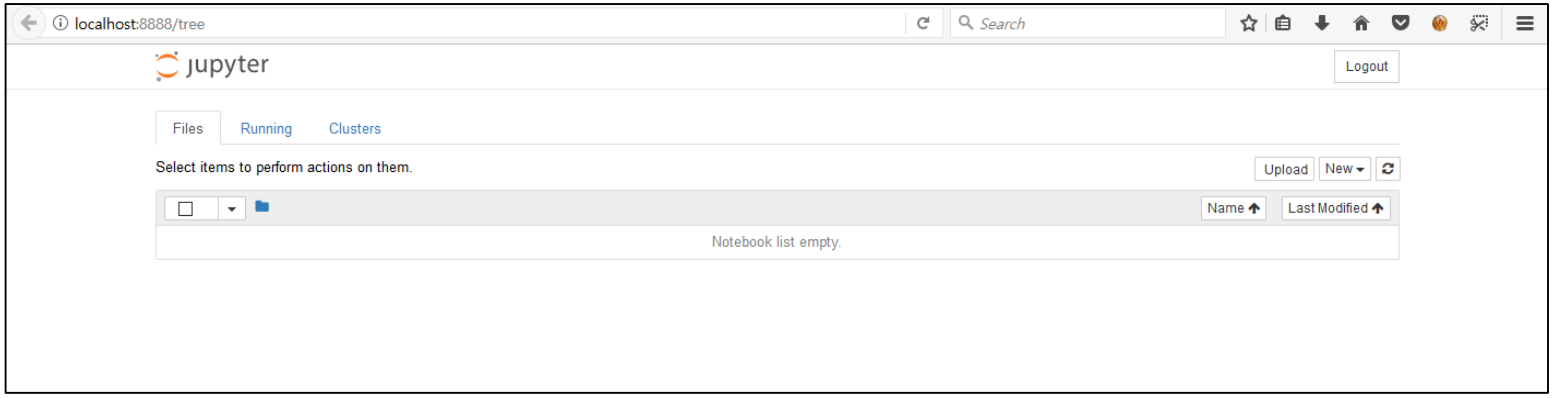
```
C:\test> pip install ipython
```

- Ipython을 설치하고 다음에는 Jupyter를 설치한다.

```
C:\test> pip install jupyter
```

- 정상적으로 Ipython과 Jupyter가 설치되었다면 아래 명령어를 통해서 Jupyter를 구동한다.

```
C:\test> jupyter notebook
```



<Jupyter 구동 모습>

4. 설치 및 실행

4.3 IPython 및 Jupyter 실행하기(3/4)

- Jupyter Notebook의 특징
  - ✓ Coding 한 결과를 실시간으로 확인이 가능하다.
  - ✓ 자동완성 기능이 있다.
  - ✓ 다양한 언어를 지원(R, Scala, Julia)한다.
  - ✓ Markdown를 지원하여 문서화가 가능하다.
  - ✓ Web 접근이 가능하면 접속 가능하다.
- Jupyter Notebook 구동하는 방법에 대해서 알아보자.
  - ✓ Jupyter를 구동할 때 시작 위치가 중요하다. 시작 위치란 작업 폴더의 위치다.
  - ✓ 처음 작업 위치 설정하는 방법은 2가지 있다.

1. 작업할 폴더에서 Jupyter 구동하기 (test라는 폴더에서 구동한 예시)

```
C:\test> jupyter notebook
```

2. Jupyter 옵션으로 폴더 지정하기

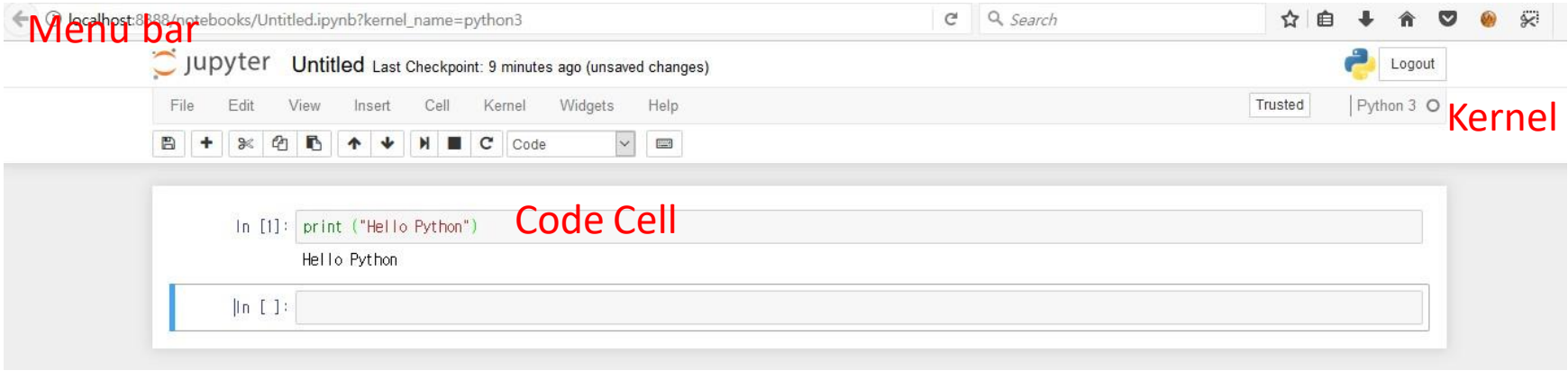
   --notebook-dir 은 옵션으로 처음 시작 경로를 설정할 때 사용한다

```
C:\> jupyter notebook --notebook-dir="c:/test"
```

4. 설치 및 실행

4.3 IPython 및 Jupyter 실행하기(4/4)

- Jupyter Notebook 실행 화면 및 단축키



- Ctrl + Enter : 현재 Cell 실행
- Shift + Enter : 아래의 Cell를 선택하고 Cell 실행
- Alt + Enter : Cell 실행하고 아래에 새 Cell 추가
- Ctrl + s : notebook 저장
- Line Number 보기

➢ Command Mode (cell에서 esc 누르면 모드 전환)

- y : code cell로 전환 / m: Markdown cell
- a : 현재 Cell 이전에 새로운 Cell 만들기 / b : 현재 Cell 다음에 새로운 Cell 만들기
- dd : 현재 cell 삭제
- l : 라인번호 보이기
- h : 단축키 목록 보이기

4. 기능소개

세부 목차

5. 기능소개

5.1 Python 변수 만들기 및 화면 출력

- 변수명 만들기
  - 변수명을 만들 때는 영어 소문자, 대문자, 숫자, _으로 구성하면 된다. 하지만 단 숫자로는 시작할 수 없다.
  - Python은 대소문자를 구별한다.
  - Python에서 사용되는 키워드 및 예약어는 사용이 불가하다.
  - 변수명 예시

```
사용 가능한 변수명
a, a1, my_name, your_job, MyName, _private,_private_member

변수명이 될 수 없는 변수명
1abc, @file, fil@e, %x
```

  - 아래는 예약어 리스트다. 변수명으로 사용할 수 없다.

```
False,None,True,and,as,assert,break,class,continue,def,del,elif,else,except,finally,for,from,
global,if,import,in,is,lambda,nonlocal,not,or,pass,raise,return,try,while,with,yield
```

- 화면 출력하기 (print)
  - Python에서 화면을 출력할 때 사용하는 함수는 print() 이다.

```
>>> print ("Hello Python")
Hello Python
```

- Sequence형 자료형이란?
  - Sequence형 자료형은 여러 객체를 저장할 수 있는 순서를 갖는 연속된 자료형이다.
  - 자료형의 종류는 아래와 같다.

List, Tupe, String

  - Sequence형 자료형에는 공통적인 연산이 있다.
    - 인덱싱(indexing)
    - 슬라이싱(slicing)
    - 연결하기
    - 반복하기
    - 멤버십 테스트
    - 길이정보
  - Sequence형 자료형의 색인의 처음은 항상 0부터 시작한다. 아래는 문자열 예시다.

문자열	a	p	p	l	e
Index	0	1	2	3	4

5. 기능소개

5.2 Python 자료형(튜플)(2/7)

- Python의 자료 구조는 다른 언어에 비해 단순하지만 매우 강력하다.
- 튜플, 리스트, 사전, 세트라는 자료 구조를 가지고 있다.
  - 튜플(Tuple)
    - 1차원의 크기를 가지고 있는 변경이 불가능한 Sequence 자료형이다.
    - 값 사이에 콤마(,)를 사용하여 튜플을 생성한다.

```
>>> tuple_tmp = 1, 2, 3
>>> tuple_tmp
(1, 2, 3)
```

- 괄호 ( ) 를 사용하면 값을 묶어서 정의할 수 있다.

```
>>> tuple_tmp2 = (1, 2, 3), (4, 5)
>>> tuple_tmp2
((1, 2, 3), (4, 5))
```

- 모든 Sequence형 자료형은 튜플 메서드를 사용하여 튜플로 변환할 수 있다.

```
>>> tuple([1,2,3])
(1, 2, 3)
>>> tuple("Python String")
('P', 'y', 't', 'h', 'o', 'n', ' ', 'S', 't', 'r', 'i', 'n', 'g')
```

문자열을 튜플로 변경

- 대괄호 [ ] 을 사용하면 튜플의 각 원소에 접근할 수 있다. Sequence 자료형의 색인의 시작은 0이다.

```
>>> tuple_tmp[0]
1
>>> tuple_tmp[1]
2
```

5. 기능소개

2. Python 자료형(튜플)(3/7)

- 튜플(Tuple)

  - 튜플 안의 원소는 서로 다른 자료형 또는 객체를 담을 수 있다.

```
>>> f = lambda x: x**2
>>> tuple_tmp3 = (f, [1,2], "python")
>>> tuple_tmp3
(<function <lambda> at 0x0000022D3455B9D8>, [1, 2], 'python')
```

f는 lambda 함수의 객체이며, [1,2]는 리스트 “python”은 문자열

  - 한번 생성된 튜플은 각 원소를 변경하는 것은 불가능하다.

```
>>> tuple_tmp1 = 1,2,3,4,5
>>> tuple_tmp1[1]
2
>>> tuple_tmp1[1] = 4
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

처음 생성된 튜플의 원소의 값을 변경할 경우 Error가 발생

  - 다른 Sequence형 자료형처럼 +, * 연산자를 사용하여 이어붙일 수도 반복할 수도 있다.

```
>>> tuple_tmp1 + (6, 7)
(1, 2, 3, 4, 5, 6, 7)
>>> tuple_tmp1 * 3
(1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
```

  * 객체는 복사되지 않으며, 그 객체에 대한 참조만 복사가 된다.

  - 대입연산자를 사용하면 오른쪽에 있는 변수에서 값이 분리되는데 이를 unpacking이라고 한다.

```
>>> a, b, c, d, e = tuple_tmp1
>>> a
1
>>> a
2
```

tuple_tmp1의 원소의 값들이 a, b, c, d, e에 분리되어 저장

5. 기능소개

2. Python 자료형(리스트)(4/7)

- 리스트(list)
  - 튜플과 같은 Sequence형 자료형이나 크기나 원소 값을 변경할 수 있다.
  - 대괄호 [ ] 나 list() 함수를 사용해서 생성할 수 있다.

```
>>> list_tmp = [1, 2, 3, 4, 5]
>>> list_tmp
[1, 2, 3, 4, 5]
```

  - 튜플을 리스트 형식으로 list함수를 사용하면 변환 할 수 있다.

```
>>> list_tmp2 = list(tuple_tmp1)
>>> list_tmp2
[1, 2, 3, 4, 5]
```

- 리스트도 튜플과 같이 +, * 연산자를 사용하여 이어붙일 수도 반복할 수도 있다.

```
>>> list_tmp + list_tmp  [
1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
>>> list_tmp * 4
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

- 리스트의 메서드를 사용하면 원소 추가 및 삭제를 할 수 있다.

```
>>> list_tmp.append(6)
>>> list_tmp
[1, 2, 3, 4, 5, 6]
>>> list_tmp.remove(2)
>>> list_tmp
[1, 3, 4, 5, 6]
```

5. 기능소개

2. Python 자료형(사전)(5/7)

- 사전(Dicationary)
  - 사전은 Python에서 중요한 자료형 중 하나입니다. Key-Value 형식으로 데이터를 저장한다.
  - 중괄호 {} 나 dict() 함수를 사용해서 생성할 수 있다.
  - 아래는 사전의 중요 내용이다.
    - 키(Key) – 값(Value)을 원소라고 부른다.
    - Dictionary은 중괄호 안에 쉼표로 구분된 원소를 넣어 생성한다.
    - Dictionary_이름 = { 키_1 : 값_1, 키_N : 값_N}
    - 매핑형에서는 key를 이용해 value에 접근한다.
    - 내부적으로 해시 기법을 사용하기 때문에 검색 속도가 빠르다.
    - key 값은 중복을 허용하지 않는다.
  - 사전 생성 및 Key값을 사용해서 Value값을 호출하는 예시다.

```
>>> contacts = {"철수" : "02-543-11111", "영희" : "02-543-2222"}
>>> contacts
{'철수': '02-543-11111',  ' 영희': '02-543-2222'}
```

철수, 영희가 Key이고 각각의 전화번호가 Value값이다.

```
>>> print (contacts['철수'])
02-543-11111
```

Key를 통해서 Value값에 빠르게 접근할 수 있다.

5. 기능소개

2. Python 자료형(집합)(6/7)

- 집합(Set)
  - 집합은 여러 값을 순서 없이 그리고 중복 없이 모아 놓은 자료형이다.
  - Python에서는 set과 frozenset 두 가지 집합 자료형을 제공한다.
  - Set은 변경 가능한 집합, frozenset은 변경 불가능한 집합이다.
  - 반복 가능한(Iterable) 객체로부터 집합을 만들 수 있지만 모든 데이터가 집합의 원소로 사용 할 수 있는 것은 아니다. Hashable이면서 변경 불가능한 자료형만이 집합의 원소로 사용 가능하다. 페이지 아래 예제를 참고하기 바란다.
  - 중괄호 { }와 set() 함수를 사용하여 생성할 수 있다.

```
>>> a = set()
>>> b = {1, 2, 3}
>>> a
set()
>>> b
{1, 2, 3}
```

```
>>> a = [1,2]
>>> b = [3, 4]
>>> c = {a,b}
Traceback (most recent call last):  F
ile "<stdin>", line 1, in <module>  T
ypeError: unhashable type: 'list'
```

```
>>> a = (1,2)
>>> b = (3,4)
>>> c = {a, b}
>>> c
{(1, 2), (3, 4)}
```

* 왼쪽 예제는 원소의 값이 변경 가능한 리스트로 집합을 생성할 때 Error가 발생한다.

* 오른쪽 예제는 한번 생성하면 값 변경이 불가능한 튜플로 집합을 생성하면 정상적으로 동작한다.

2. Python 자료형(집합)(7/7)

- 집합(Set)
  - 집합은 수학의 집합처럼 연산이 가능하다.
    - | 연산자를 사용하여 합집합 연산을 한다.

```
>>> a = {1, 2, 3, 4, 5, 6}
>>> b = {4, 5, 6, 7, 8, 9}
>>> a | b
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

- & 연산자를 사용하여 교집합 연산을 한다.

```
>>> a & b
{4, 5, 6}
```

- \- 연산자를 사용하여 차집합 연산을 한다.

```
>>> a - b
{1, 2, 3}
```

- ^ 연산자를 사용하여 대칭 차집합 연산을 한다

```
>>> a ^ b
{1, 2, 3, 7, 8, 9}
```

5. 기능소개

3. Python 도움말 얻기

- dir() 함수와 help를 사용하여 객체가 가지고 있는 메서드 및 도움말을 얻는 방법을 알아보자. dir() 함수를 사용하면 객체가 가지고 있는 메서드 리스트가 출력된다.

```
>>> dir(list)
['_add_', '_class_', '_contains_', '_delattr_', '_delitem_', '_dir_', '_doc_', '_eq_', '_format_', '_ge_', '_getattribute_', '_getitem_', '_gt_',
'_hash_', '_iadd_', '_imul_', '_init_', '_init_subclass_', '_iter_', '_le_', '_len_', '_lt_', '_mul_', '__ne_', '_new_', '_reduce_', '_reduce_ex_',
'__repr ', ' reversed ', ' rmul ', ' setattr ', ' setitem ', ' sizeof ', ' str ', ' subclasshook ', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert',  'pop', 'remove', 'reverse', 'sor
t']
>>> dir(tuple)
['_add_', '_class_', '_contains_', '_delattr_', '_dir_', '_doc_', '_eq_', '_format_', '_ge_', '_getattribute_', '_getitem_', '_getnewargs_', '_gt_',  '_hash_', '_init_', '_init_subclass_', '_iter_', '_le_', '_len_', '
lt_', '_mul_', '_ne_', '_new_', '_reduce_', '_reduce_ex_', '_repr_', '_rmul_',
'__setattr ', ' sizeof ', ' str ', ' subclasshook ', 'count', 'index']
>>> dir(dict)
['_class_', '_contains_', '_delattr_', '_delitem_', '_dir_', '_doc_', '_eq_', '_format_', '_ge_', '_getattribute_', '_getitem_', '_gt_', '_hash_',  '_init_', '_init_subclass_', '_iter_', '_le_', '_len_', '_lt_', '_ne
', '_new_', '_reduce_', '_reduce_ex_', '_repr_', '_setattr_', '_setitem_',
'__sizeof ', ' str ', ' subclasshook ', 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values']
>>> dir(set)
['_and_', '_class_', '_contains_', '_delattr_', '_dir_', '_doc_', '_eq_', '_format_', '_ge_', '_getattribute_', '_gt_', '_hash_', '_iand_', '_init_',
'_init_subclass_', '_ior_', '_isub_', '_iter_', '_ixor_', '_le_', '_len_', '_lt_', '_ne_', '_new_', '_or_', '_rand_', '_reduce_', '_reduce_ex_',
'__repr ', ' ror ', ' rsub ', ' rxor ', ' setattr ', ' sizeof ', ' str ', ' sub ', ' subclasshook ', ' xor__', 'add', 'clear', 'copy', 'difference',  'difference_update', 'discard', 'inte
rsection', 'intersection_update', 'isdisjoint', 'issubset', 'issuperset', 'pop', 'remove', 'symmetric_difference',  'symmetric_difference_update', 'union', 'update'
]
```

* 출력 결과에서 가 붙지 않은 문자열이 메서드의 이름이다. (예: append, clear, copy....)

```
>>> help(dict.update)
Help on method_descriptor:
update(...)
   D.update([E, ]**F) -> None. Update D from dict/iterable E and F.
   If E is present and has a .keys() method, then does: for k in E: D[k] = E[k]  I
   f E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v  I
   n either case, this is followed by: for k in F: D[k] = F[k]
```

* help() 함수를 사용하여 dict 객체의 update의 도움말을 출력한다.

- Python 자료형을 아래와 같이 정리하면 아래와 같다.
  - List
    - ✓ 내용을 변경 할 수 있다.
    - ✓ 리스트 함수를 사용하여 항목을 삽입하거나 삭제 가능하다.
  - Tuple
    - ✓ 메모리에서 빠르게 동작한다.
    - ✓ 항목을 삽입하거나 삭제 불가하다.
  - Dictionary
    - ✓ 항목을 가리키는 고유한 이름을 가지고 있다.(Key)
    - ✓ 인덱스로 key 이름을 사용한다.
    - ✓ Value에는 문자열, 정수, 실수, Tuple과 List가 사용 가능하다.
    - ✓ 콜론으로 Key와 Value을 구분하고 Key 값은 중복이 되면 안된다.
    - ✓ key 값을 중복으로 입력은 가능하지만 어느 Value을 가져올지는 모른다.
    - ✓ Hash table을 사용하여 빠른 속도로 동작한다.

5. 기능소개

5.5 Python 제어문(if와 Code Block)(1/3)

- Python 인터프리터는 사용자가 입력한 순서대로 순차적으로 명령문을 하나씩 실행한다. 코드를 작성하다 보면 순차적인 흐름을 바꾸어 특정 조건일 때만 실행을 해야 할 경우가 생긴다. 특정 조건을 가진 명령문을 조건문이라고 한다.

- if문의 사용법은 아래와 같다.

```
if 조건식:
    명령문
elif 조건식:
    명령문
else:
    명령문
```

조건이 참이라면 해당 줄의 명령문이 실행후 종료

위의 조건이 거짓이면 elif 조건식을 검사 조건이 참이면 실행 후 종료

위의 조건들이 다 거짓일 경우 else문 다음 명령문이실행

Code Block

```
a = 12
if a > 15 :
    print ("15보다 큽니다.")
elif a > 10 :
    print ("10보다 큽니다.")
else:
    print ("10보다 작습니다.")

>> 10보다 큽니다.
```

a는 12로 할당되고 조건문의 첫번째는 a가 15보다 큰가를 조사한다. 거짓이기 때문에 다음 블럭에 있는 명령문은 실행되지 않고 다음 조건으로 넘어간다. 두번째 조건이 참이기 때문에 조건문 다음 명령문이 실행되고 조건문은종료

- Python의 특징 중 하나는 Python은 조건문을 사용할 때 해당 조건에 영향 받는 명령문들을 Code Block이라는 간격으로 인식한다. 위의 예제에서 빨간색 박스는 공백(혹은 탭)을 의미한다. 조건식의 다음 차례의 명령문이 공백문자를 가지고 있다면 해당 명령문은 위의 조건문에 종속되어 있다. 조건이 참일 경우 명령문은 실행되고 거짓일 경우 실행되지 않는다.

* elif : else if의 줄인 말

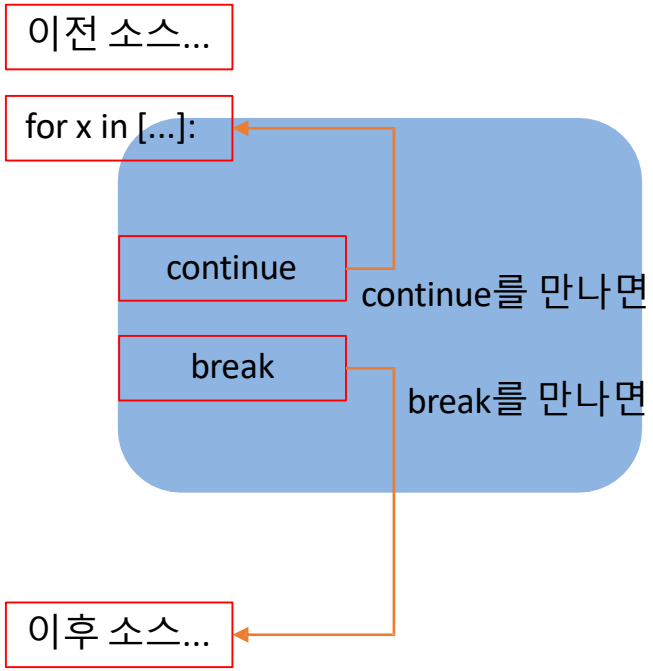
5. 기능소개

5.5 Python 제어문(for)(2/3)

- 흐름을 제어하는 또 다른 방법은 같은 부분을 반복해서 실행 하는 것이다. 이를 반복문이라고 한다.
- 반복문은 사용자가 얼마큼 반복할 것인지 정할 수 있다.
- For는 가장 많이 사용하는 반복문이며, 형식은 아래와 같다.

```
for <target> in <object> :
    명령문
```

- <object>는 sequence형 데이터여야 한다.
- <object>의 각 항목은 <target>에 치환되어 명령문 실행한다.
- 반복의 횟수는 <object>의 크기에 결정된다.
- for 문의 내부 블록에서 continue를 만나면 시작 부분으로 이동, break를 만나면 for문의 내부 블록을 종료한다.
- for를 사용할 수 있는 객체는 아래와 같다.
  - ✓ 문자열, 리스트, 튜플, 사전, range, 반복이 가능한 객체



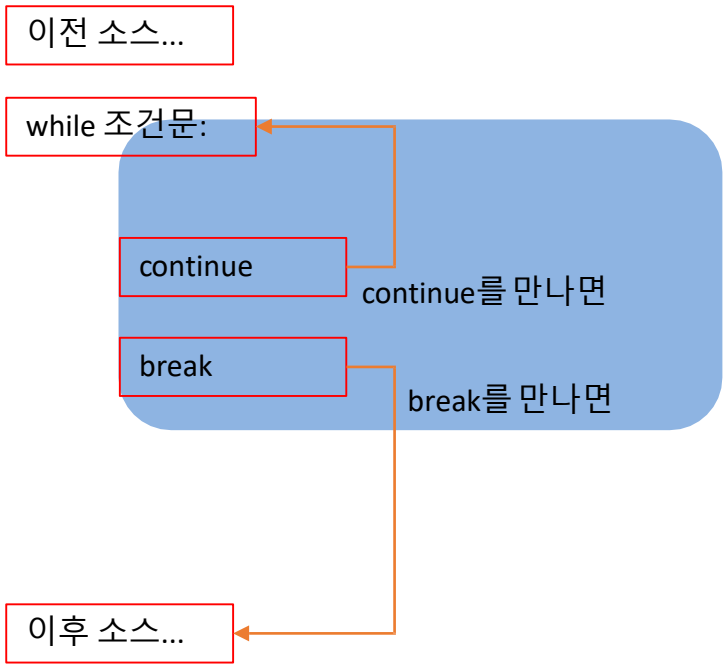
* 조건문에는 반드시 조건식 뒤에 : 를 붙인다.

5. 기능소개

5.5 Python 제어문(while)(3/3)

- for 문과 같은 반복문이며, 조건문이 거짓으로 판별되기 전까지 계속 반복된다.
- 시작 부분의 <조건>을 검사해서 결과가 참 이면 내부 블록이 반복적으로 실행된다.

```
while 조건:
        명령문
```



- 반복문에서 continue를 만나면 만나면 시작 부분으로 이동
- break를 만나면 while 문의 내부 블록을 종료
- while 문은 조건이 잘못 설정되면 무한루프에 빠질수 있다. 아래 코드는 while 문의 조건은 항상 참이기 때문에 계속해서 무한으로 반복할 것이다.

```
i = 0
while i < 1:
    print(i)
```

* i는 항상 0이기 때문에 조건은 항상 참이다.

6. 활용예제

세부 목차

1. 윤년계산 코드작성
2. 시각화
3. 기계학습

6. 활용예제

6.1 윤년계산 코드작성

- 제어문을 사용해서 간단한 예제를 작성해보다.
- 아래는 윤년 계산 하는 알고리즘이다.

1. 서력 기원 연수가 4로 나누어떨어지는 해는 윤년으로 한다.(1988년, 1992년, 1996년, 2004년, 2008년, 2012년 ...)
2. 이 중에서 100으로 나누어떨어지는 해는 평년으로 한다.(1900년, 2100년, 2200년, 2300년, 2500년 ...)
3. 그중에 400으로 나누어떨어지는 해는 윤년으로 둔다.(1600년, 2000년, 2400년 ...)

출처) https://ko.wikipedia.org/wiki/%EC%9C%A4%EB%85%84

- 간단하게 배웠던 조건문과 자료형을 사용하여 위의 알고리즘을 Python 코드로 작성하면 아래와 같다.

```
# 아래 2가지 방법 모두 초기 리스트 생성
#leap_year = list()
leap_year = []

for year in range(1900, 2101):
    if(year% 4 == 0 and year % 100 != 0 or year % 400 == 0):
        print (year)
```

range() 함수를 사용하여 1900 ~ 2100까지 Sequence형 숫자를 생성한다.
for문을 사용하여 1900~2100까지 숫자 하나씩 if 조건문을 사용하여 해당 값이 참일 때 출력하는 코드다.
붉은 글씨로 되어 있는 부분이 위 알고리즘을 구현한 코드다.

- #으로 시작하는 구문은 주석문이다. 프로그램에 영향을 미치지 않는다.
- range(시작하는 숫자 : 끝나는 숫자 : 간격) 함수는 사용자가 원하는 범위와 간격만큼 숫자를 만들어 주는 함수다.
- 아래는 help () 함수를 사용하여 range()의 사용법을 출력한 화면이다.

```
>>> help(range)
Help on class range in module builtins:

class range(object)
 | range(stop) -> range object
 | range(start, stop[, step]) -> range object
```

6. 활용예제

6.2 시각화

- 두 번째 예제는 Python으로 하는 Machine Learing 알고리즘을 적용하는 것이다.
- ML(Machine Learning)을 제공하는 라이브러리 중에서 Scikit-Learn이라는 라이브러리가 존재한다.
- 아이리스 데이터를 사용하여 시각화 및 ML에 대한 맛보기을 진행을 해보겠다.(Jupyter 사용)

```
import sklearn as sk
import numpy as np
import matplotlib.pyplot as plt
```

Scikit-learn, numpy, matplotlib를 사용하기 위해서 import 한다.

➢ scikit-learn에 있는 아이리스 데이터를 로드한다.

```
from sklearn import datasets
iris = datasets.load_iris()
X_iris, y_iris = iris.data, iris.target
```

datasets를 import하여 아이리스 데이터를 load 시킨다.

➢ 데이터 분리 및 전처리를 위해서 해당 기능을 로드한다.

```
from sklearn.preprocessing import StandardScaler
```

전처리를 위해서 StandardScaler 메서드도 호출한다.

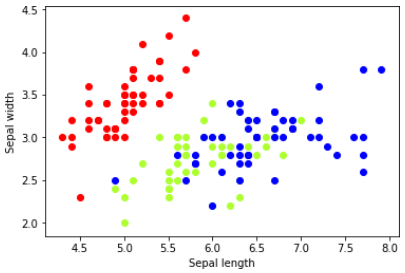
➢ train_test_split 메서드를 사용해서 데이터를 분리한다.

```
X, y = X_iris[:,:2], y_iris
```

➢ 데이터를 시각화하여 출력한다.

```
colors = ['red', 'greenyellow', 'blue']
for i in range(len(colors)):
    px = X[:, 0][y == i]
    py = X[:, 1][y == i]
    plt.scatter(px, py, c=colors[i])
plt.xlabel('Sepal length')  plt.yla
bel('Sepal width')  plt.show()
```

matplotlib의 scatter 메서드를 사용해서 산포도 그래프를 출력하는 코드를 작성한다. 다른 성격의 데이터를 출력하기 위해서 각각의 데이터를 브로드캐스팅하여 px, py에 매핑하여 출력한다.



6. 활용예제

6.2 기계학습 SVM(1/2)

- 앞의 시각화한 데이터를 분석을 위한 Machine Learning(기계학습)을 진행해 보자.
- 예제는 scikit-learn 공식 문서에서 참조했다.
  - 앞의 그림처럼 iris 데이터는 3개의 종류로 분류 되어있다.

<Iris 사진>

출처:위키피디아

make_meshgrid, plot_contours 함수를 생성하여 데이터 형식 및 분류기를 실행한다.

- sepal width (cm), petal length (cm), petal width (cm)의 값에 따라서 'setosa', 'versicolor', 'virginica ' 으로 분류할 수 있다.
- 이전 페이지에서는 데이터를 시각화하므로써 데이터의 분포를 확인한다.
- 데이터를 SVM(Support Vector Machine) 방식으로 Iris 형태를 분류해줄 수 있는 분류기를 만들어 보자
- 아래는 코드는 분류기에 값을 넣고 시각화를 할 수 있는 함수다.
- 코드를 이해하기 위해서는 Numpy 및 선형대수의 지식이 필요하다. 예제에서는 설명을 생략하겠다.

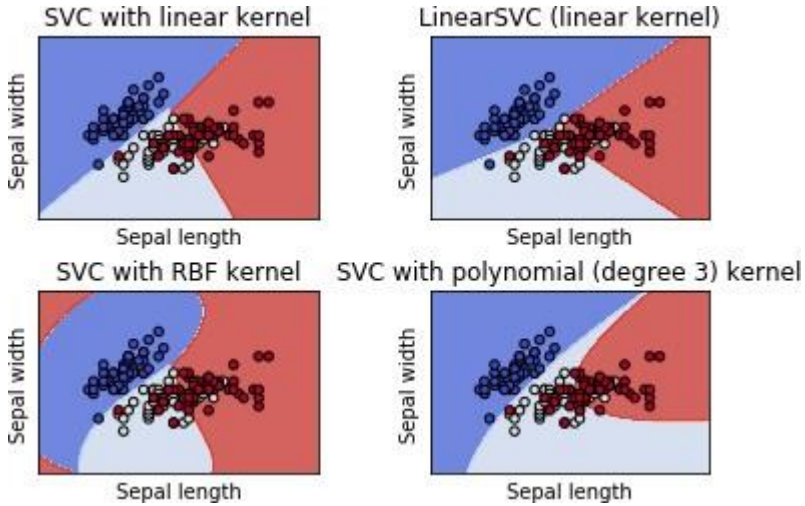
```
def make_meshgrid(x, y, h=.02):
    x_min, x_max = x.min() - 1, x.max() + 1
    y_min, y_max = y.min() - 1, y.max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    return xx, yy

def plot_contours(ax, clf, xx, yy, **params):
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    out = ax.contourf(xx, yy, Z, **params)
    return out
```

6. 활용예제

6.2 기계학습 SVM(2/2)

- 앞의 시각화한 데이터를 분석을 위한 전처리 및 기계학습을 진행해 보자.
- 예제는 scikit-learn 공식 문서에서 참조하였다.
  - SVM을 kernel 형식에 따라서 4가지로 구분하여 분류기 모형을 실행한다. SVM에 대한 자세한 설명은 아래 url에서 학습하기 바랍다.
  - https://en.wikipedia.org/wiki/Support_vector_machine

```
C = 1.0 # SVM regularization parameter
models = (svm.SVC(kernel='linear', C=C),
        svm.LinearSVC(C=C),  svm.SVC(kernel='r
        bf', gamma=0.7, C=C),  svm.SVC(kernel='
        poly', degree=3, C=C))
models = (clf.fit(X, y) for clf in models)

titles = ('SVC with linear kernel', 'LinearSVC (linear kernel)',
         'SVC with RBF kernel',
        'SVC with polynomial (degree 3) kernel')

fig, sub = plt.subplots(2, 2)  plt.subplots_adju
st(wspace=0.4, hspace=0.4)  X0, X1 = X[:, 0],
X[:, 1]
xx, yy = make_meshgrid(X0, X1)

for clf, title, ax in zip(models, titles, sub.flatten()):
    plot_contours(ax, clf, xx, yy,
                  cmap=plt.cm.coolwarm, alpha=0.8)
    ax.scatter(X0, X1, c=y, cmap=plt.cm.coolwarm, s=20, edgecolors='k')
    ax.set_xlim(xx.min(), xx.max())
    ax.set_ylim(yy.min(), yy.max())
    ax.set_xlabel('Sepal length')  ax
    .set_ylabel('Sepal width')  ax.se
    t_xticks(())  ax.set_yticks(())  ax
    .set_title(title)
plt.show()
```



Iris 데이터를 SVM 분류기 4개를 사용하여 각각 모형을 돌려본 결과를 시각화 한 것이다. 각각의 커널의 종류에 따라서 분류 결과가 약간의 차이가 발생하는 것을 확인할 수 있다.

본 예제는 scikit-learn 공식 문서에서 참조했다. 아래 주소에서 전체 소스를 다운로드 받을 수 있다.

http://scikit-learn.org/stable/auto_examples/svm/plot_iris.html

Q Python에서 Lambda식이라는 것은 무엇인가요?

A Python에서 함수가 값이라면, 함수를 값으로 갖는 식이 존재합니다. 이러한 표현식을 람다(Lambda)식이라고 합니다. 그리스 문자로 λ로 표현합니다. 또 익명 함수라고도 불리기도 합니다. 간단히 정리하면 값을 반환하는 단순한 문장으로 이루어진 식이라고 생각하면 됩니다.

예) lambda x : x ** 2 (제곱수를 반환하는 람다식)

Q Python에서 클로저란 무엇인가요?

A 클로저는 아주 유용하고 강력합니다. 클로저는 다른 함수에서 반환되는 동적으로 생성된 함수입니다. 클로저는 클로저를 생성하는 함수가 끝나더라도 생성된 시점의 네임스페이스에 계속해서 접근이 가능합니다.

7. FAQ

Q Dictionary 및 List는 어디에 사용해야 하나요?

A 사전(Dictionary)은 특정한 값으로 다른 값을 '찾을 때' 사용하며 Dictionary는 그렇기 때문에 Lookup table이라고 부르기도 한다고 합니다.
List의 경우에는 차례를 지켜야 하는 순서가 있는 항목에서 값을 이용해서 해당 위치를 찾아도 괜찮을 때 사용하면 됩니다.

Q for 와 while 차이를 알려주세요

A for문은 조건 및 집합의 순환을 위해서 반복할 수 있습니다. while문은 하고 싶은 어떤 종류의 반복적인 구성도 할 수 있습니다. for와 while 둘 다 반복문이라는 점은 같지만 while문은 무한 루프를 일으킬 수 있기 때문에 보통은 for문으로 많은 일을 하게 됩니다.

8. 용어정리

용어	설 명
CPython	C로 작성된 Python을 의미한다. 기본적으로 Python이라고 하면 CPython을 의미한다. Java로 구현된 JPython, .Net용으로 개발된 IronPython, Python으로 구현된 PyPy등이 있다.
JIT	JIT(Just-In-Time) 컴파일 또는 Dynamic Translation(동적 번역) 이라고 하며 적절할 때에 컴파일하는 방식으로 Python의 Numba, PyPy 패키지가 여기에 속한다.
인터프리터 (interpreter)	인터프리터는 프로그래밍 언어의 소스 코드를 바로 실행하는 컴퓨터 프로그램 또는 환경을 말한다. 원시 코드를 기계어로 번역하는 컴파일러와 대비된다.
메서드	클래스 안의 함수를 부르는 이름이다.
전역변수 (global)	해당 프로그램의 전체에서 사용 가능한 변수다.
지역변수 (local)	함수 내부에서 선언된 변수로 함수 안에서만 사용 가능하다. 전역변수와 다르게 선언한 함수 이외에는 사용할 수 없다.
디버깅	프로그램의 버그를 잡는 과정을 의미한다. Python에서는 pdb 라는 Python Debugger 모듈을 제공하고있다.

Open Source Software Installation & Application Guide