

(Ch 13) Python Modules and Packages

- Utilizing Modules
- Python Package
- PyPI and Pip: Automatic Installation

Python Module 만들고 불러보기 [1/2]

모듈에 대해서 자세히 살펴보기 전에 간단한 모듈을 한번 만들어 보자.

```
# mod1.py
def sum(a, b):
    return a + b
```

위와 같이 sum 함수만 있는 파일 mod1.py를 만들고 C:\Python 디렉터리에 저장하자. 이 파일이 바로 모듈이다. 지금까지 에디터로 만들어 왔던 파일과 다르지 않다.

반드시 mod1.py를 저장한 위치로 이동한 다음 이후 예제를 진행해야 한다. 그래야만 대화형 인터프리터에서 mod1.py를 읽을 수 있다. 이제 아래와 같이 따라 해보자.

```
>>> import mod1
>>> print(mod1.sum(3,4))
7
```

Python이 module의 위치를 어떻게 찾을것인가?

Python Module 만들고 불러보기 [2/2]

이번에는 mod1.py 파일에 다음 함수를 추가해 보자.

```
def safe_sum(a, b):  
    if type(a) != type(b):  
        print("더할수 있는 것이 아닙니다.")  
        return  
    else:  
        result = sum(a, b)  
    return result
```

Python이 module의 위치를 어떻게 찾을것인가?

```
from mod1 import sum, safe_sum
```

```
from mod1 import *
```

이런식의 import를 하려면 mod1.py가 있는 directory로 이동하거나
or
mod1.py가 있는곳까지의 file path를 Python에게 알리거나

if __name__ == "__main__": 의 의미

```
# mod1.py
def sum(a, b):
    return a+b

def safe_sum(a, b):
    if type(a) != type(b):
        print("더할수 있는 것이 아닙니다.")
        return
    else:
        result = sum(a, b)
        return result

if __name__ == "__main__":
    print(safe_sum('a', 1))
    print(safe_sum(1, 4))
    print(sum(10, 10.4))
```

다른 PL들은 main() 이 정해져 있지만,
Python은 main() 정해진게 아니고, 지금
실행되는 module이 main의 역할을 함!

직접 mod1.py 파일을
IDLE에서 "run module" click

(__name__ == "__main__") 가 True

더할 수 있는 것이 아닙니다.

None

5

20.4

다른 Python화일에서 mod1.py
파일을 import 한경우
>>> import mod1

(__name__ == "__main__")가 False

if문장의 code는 수행이 안됨

* >>> python mod1.py 혹은
* IDLE에서 run시키는 code가 main() 역할

이때에 special variable __name__ 은 "__main__"을 가지게 된다.

```
18
19
20
21 def sum(a, b):
22     return a+b
23
24 def mult(a, b):
25     return a * b
```

```
In [4]: runfile('C:/Users/hjk/.spyder-py3/
temp.py', wdir='C:/Users/hjk/.spyder-py3')
```

```
In [5]: __name__
Out[5]: '__main__'
```

```
In [6]:
```

클래스나 변수 등을 포함한 모듈

```
# mod2.py
```

```
PI = 3.141592
```

```
class Math:
```

```
    def solv(self, r):  
        return PI * (r ** 2)
```

```
def sum(a, b):  
    return a+b
```

```
if __name__ == "__main__":  
    print(PI)  
    a = Math()  
    print(a.solv(2))  
    print(sum(PI, 4.4))
```

mod2.py file

modtest.py file

Python이 module의 위치를 어떻게 찾을것인가?

```
# modtest.py  
import mod2  
result = mod2.sum(3, 4)  
print(result)
```

mod2.py와 modtest.py가 같은 directory에 있어야 한다

이 파일은 반지름을 계산하는 Math 클래스와 두 값을 더하는 sum 함수 그리고 원주율 값인 PI 변수처럼 클래스, 함수, 변수 등을 모두 포함하고 있다. 파일 이름을 mod2.py로 하고

IDLE에서 "run module" click

```
3.141592  
12.566368  
7.541592
```

다른 Python화일에서 mod1.py 파일을 import 한경우

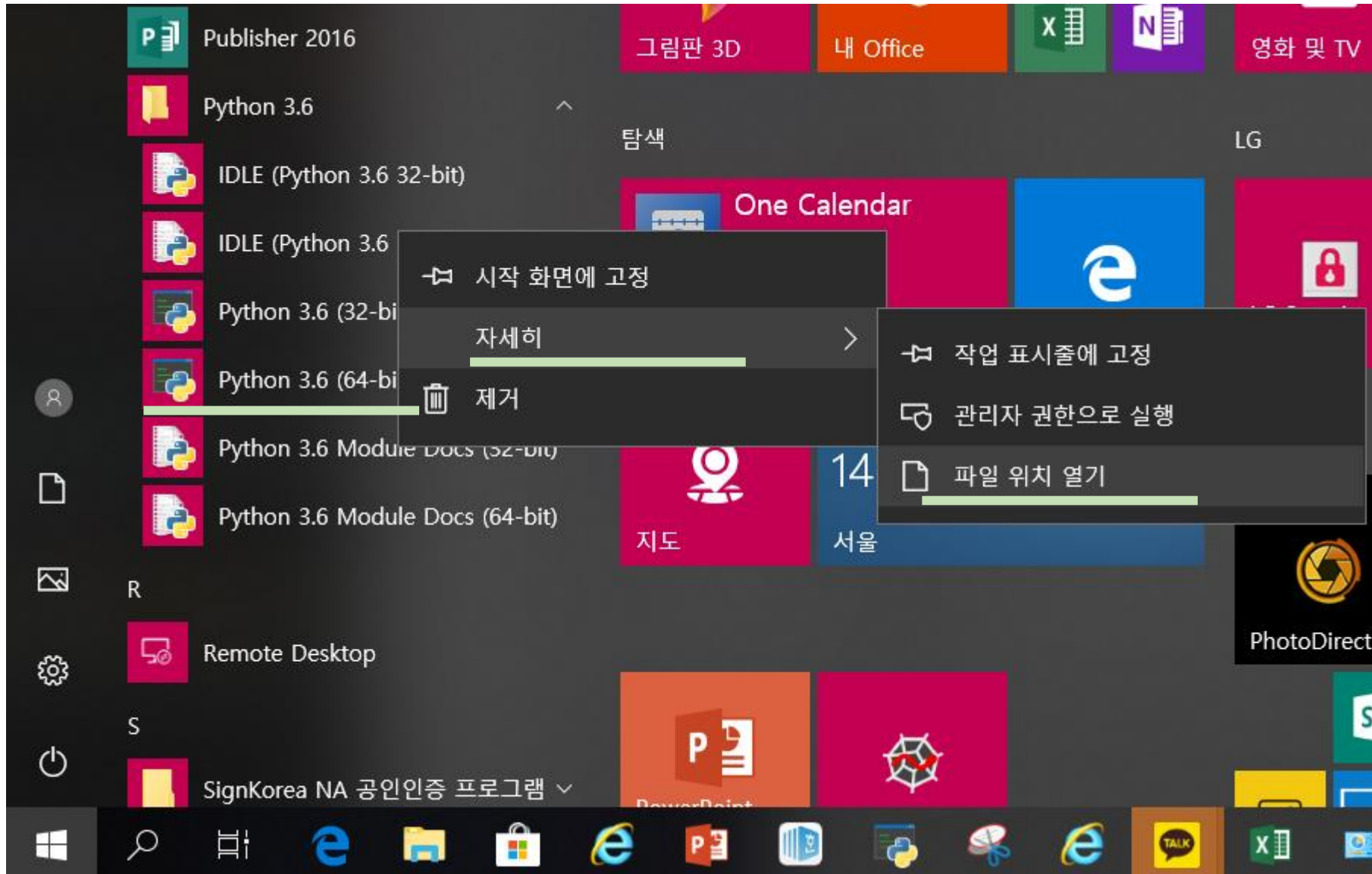
```
>>> import mod2  
>>>
```

`__name__ == "__main__"` 이 거짓이 되므로 아무런 값도 출력되지 않는다.

IDLE에서 동료가 만든 Python File을 실행시키려면 [1/2]

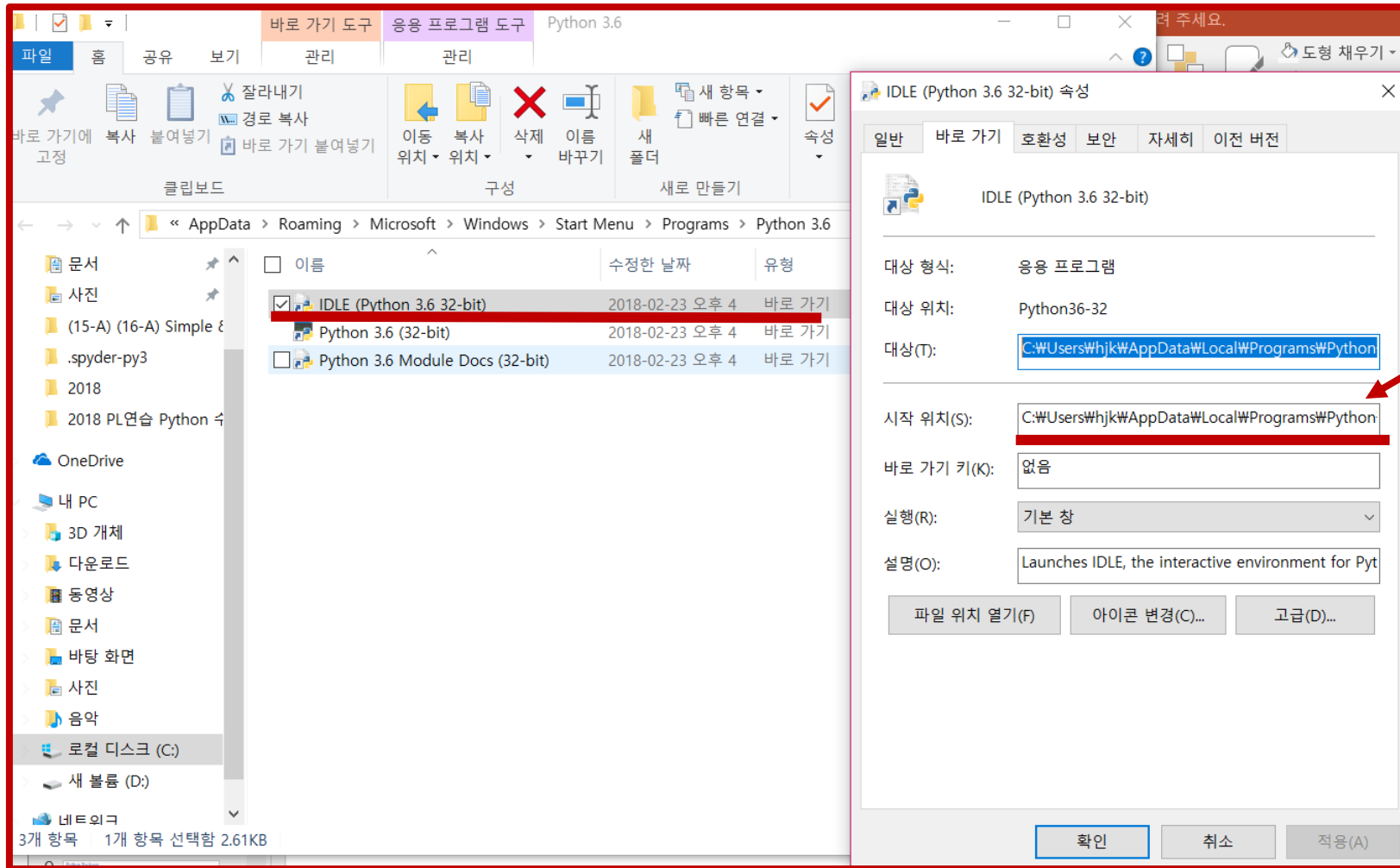
- Window_Key 를 누르고 Python 을 찾아서:

Mouse 오른쪽키 click >> “자세히” click >> “파일위치열기“ click



IDLE에서 동료가 만든 Python File을 실행시키려면 [2/2]

- IDLE 직접가기 Icon이 나오면: Mouse 오른쪽키 click >> “속성(R)” click



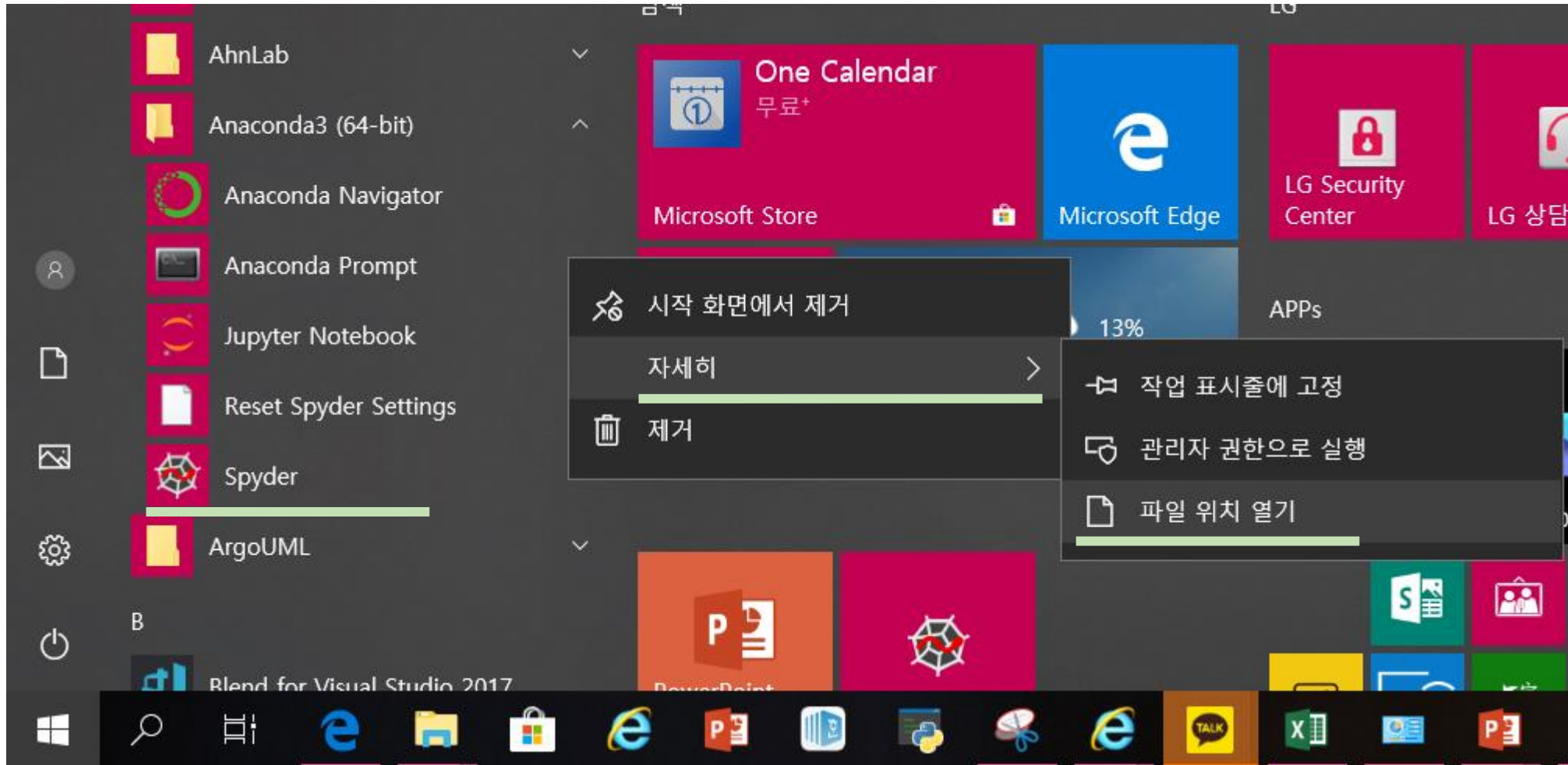
이곳 directory가 Python shell 이 실행되는 장소

`C:\Users\Whjk\AppData\Local\Programs\Python\Python36-32`

Anaconda Spyder 에서 동료가 만든 Python File을 실행시키려면 [1/2]

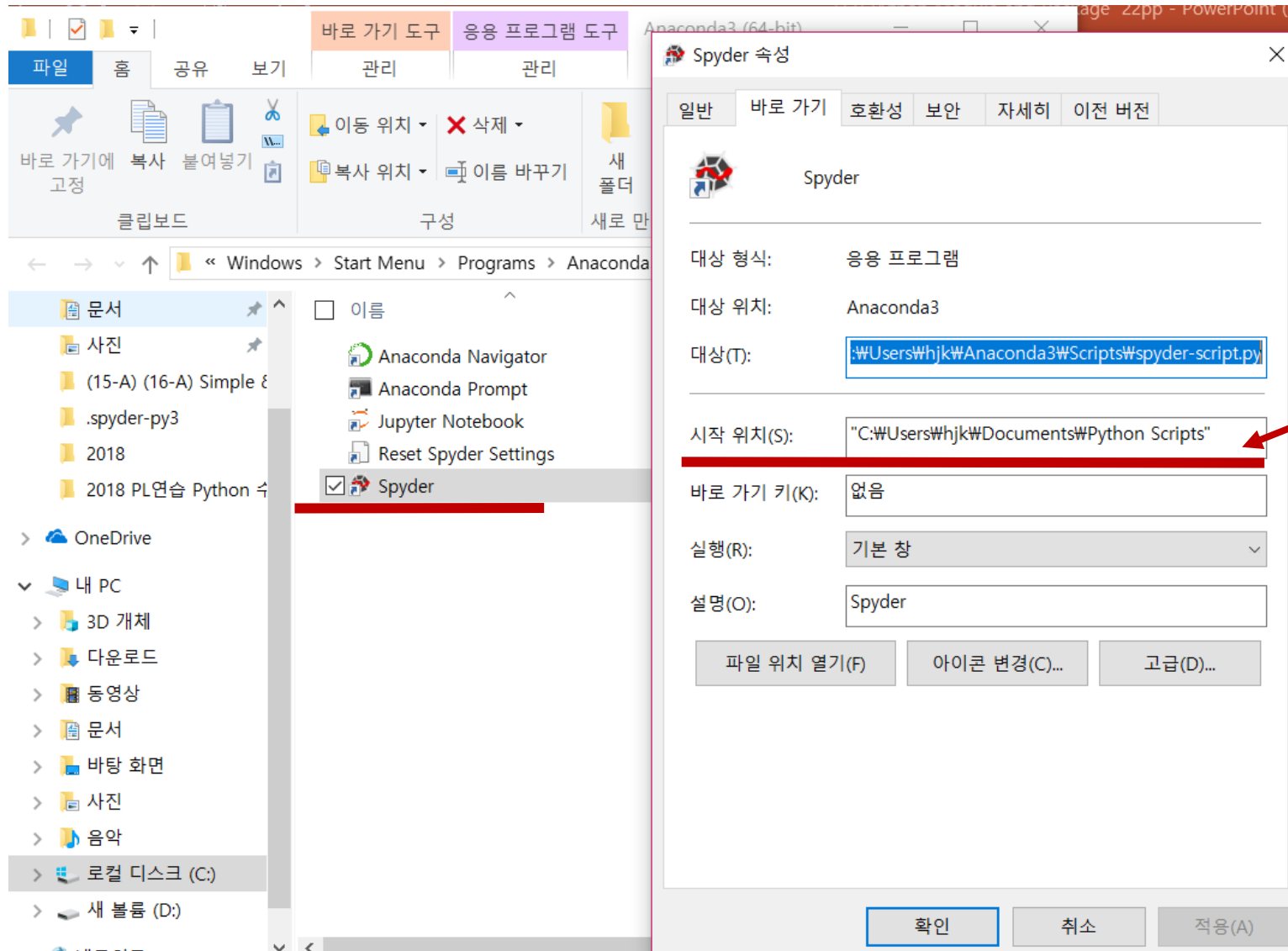
- Window_Key 를 누르고 Python 을 찾아서:

Mouse 오른쪽키 click >> “자세히” click >> “파일위치열기” click



Anaconda Spyder에서 동료가 만든 Python File을 실행시키려면 [2/2]

- Spyder 직접가기 Icon이 나오면: Mouse 오른쪽키 click >> “속성(R)” click



이곳 directory가 Spyder
의 Anaconda Python
Shell이 실행되는장소

sys.path: Python이 py file을 자동적으로 먼저 뒤지는 Directory들

```
Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
Python 3.6.4 (v3.6.4:d48ebeb, Dec 19 2017, 06:04:45) [MSC v.1900
32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
>>> import sys
>>>
>>> sys.path
['', 'C:\\Users\\hjk\\AppData\\Local\\Programs\\Python\\Python36-32\\Lib\\idlelib', 'C:\\Users\\hjk\\AppData\\Local\\Programs\\Python\\Python36-32\\python36.zip', 'C:\\Users\\hjk\\AppData\\Local\\Programs\\Python\\Python36-32\\DLLs', 'C:\\Users\\hjk\\AppData\\Local\\Programs\\Python\\Python36-32\\lib', 'C:\\Users\\hjk\\AppData\\Local\\Programs\\Python\\Python36-32\\lib\\site-packages']
>>>
>>>
>>> |
```

```
IPython console
Console 3/A
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900
Type "copyright", "credits" or "license" for more information.

IPython 6.2.1 -- An enhanced Interactive Python.

In [1]: import sys

In [1]:

In [2]: sys.path
Out[2]:
['',
 'C:\\Users\\hjk\\Anaconda3\\python36.zip',
 'C:\\Users\\hjk\\Anaconda3\\DLLs',
 'C:\\Users\\hjk\\Anaconda3\\lib',
 'C:\\Users\\hjk\\Anaconda3',
 'C:\\Users\\hjk\\Anaconda3\\lib\\site-packages',
 'C:\\Users\\hjk\\Anaconda3\\lib\\site-packages\\win32',
 'C:\\Users\\hjk\\Anaconda3\\lib\\site-packages\\win32\\lib',
 'C:\\Users\\hjk\\Anaconda3\\lib\\site-packages\\Pythonwin',
 'C:\\Users\\hjk\\Anaconda3\\lib\\site-packages\\IPython\\extensions',
 'C:\\Users\\hjk\\.ipython']

In [3]:
```

Python이 설치된 directory 로 python file을 이동하고나서 import 를 했는데, python file 이 있는 곳까지의 **full path** 를 Python에게 알려주면 Python이 설치된 directory로 이동하지 않고 import 가능

1. sys.path.append(모듈을 저장한 디렉터리) 사용하기 먼저 sys 모듈을 불러온다.

>>> import sys

>>> sys.path

['', 'C:\\Windows\\SYSTEM32\\python35.zip', 'c:\\Python35\\DLLs', 'c:\\Python35\\lib', 'c:\\Python35', 'c:\\Python35\\lib\\site-packages']

>>> sys.path.append("C:/Python/Mymodules")

>>> sys.path

['', 'C:\\Windows\\SYSTEM32\\python35.zip', 'c:\\Python35\\DLLs', 'c:\\Python35\\lib', 'c:\\Python35', 'c:\\Python35\\lib\\site-packages', 'C:/Python/Mymodules']

...

2. PYTHONPATH 환경 변수 사용하기

모듈을 불러와서 사용하는 또 다른 방법으로는 PYTHONPATH 환경 변수를 사용하는 방법이 있다.

C:\\Users\\home>set PYTHONPATH=C:\\Python\\Mymodules

C:\\Users\\home>python

Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:54:25) [MSC v.1900 64 bit (AM...
Type "help", "copyright", "credits" or "license" for more information.

>>> import mod2

>>> print(mod2.sum(3,4))

7

Python안에서 하던지

이것을 해줘야

>>> import mod2
혹은
>>> from mod2 import *

CMD창에서 하던지

했을때 Python이 알아서 mod2.py를 찾아감

Python Modules and Packages

- Utilizing Modules
- Python Package
- PyPI and Pip: Automatic Installation

Python Package

패키지(Packages)는 도트(.)를 이용하여 파이썬 모듈을 계층적(디렉터리 구조)으로 관리할 수 있게 해준다. 예를 들어 모듈명이 A.B인 경우 A는 패키지명이 되고 B는 A 패키지의 B 모듈이 된다.

파이썬 패키지는 디렉터리와 파이썬 모듈로 이루어지며 구조는 다음과 같다.

가상의 game 패키지 예

```
game/  
  __init__.py  
  sound/  
    __init__.py  
    echo.py  
    wav.py  
  graphic/  
    __init__.py  
    screen.py  
    render.py  
  play/  
    __init__.py  
    run.py  
    test.py
```

다음 Page에 있는것처럼 code들을 생성하면
가상의 Package가 만들어지는것

- The '__init__.py' files are required to make Python treat the directories as containing packages.
- The '__init__.py' can just be an empty file, but it can also execute initialization code for the package or set the __all__ variable, described later.

'__init__.py' 는 해당director가 part of package라는것을 알려줌

원편의 directory구조처럼 code들을 정리하고 __init__.py를 포함하고
있으면, import game.sound 같은 식으로 package내부의
subdirectory를 dot(.)으로 접근가능하게 해준다!

game, sound, graphic, play는 디렉터리명이고 .py 확장자를 가지는 파일은 파이썬 모듈이다. game
디렉터리가 이 패키지의 루트 디렉터리이고 sound, graphic, play는 서브 디렉터리이다.

Package 기본요소 준비하기 [1/2]

1. `C:/Python`이라는 디렉터리 밑에 `game` 및 기타 서브 디렉터리들을 생성하고 `.py` 파일들을 다음과 같이 만들어 보자(만약 `C:/Python`이라는 디렉터리가 없다면 먼저 생성하고 진행하자).

```
C:/Python/game/__init__.py
```

```
C:/Python/game/sound/__init__.py
```

```
C:/Python/game/sound/echo.py
```

```
C:/Python/game/graphic/__init__.py
```

```
C:/Python/game/graphic/render.py
```

`__init__.py`가 있는곳의 directory
들은 상위 directory 에 있는
Package의 subpackage 이다!

```
import game.sound.echo  
from game.graphic.echo import *  
같은 표현이 가능하다!
```

2. 각 디렉터리에 `__init__.py` 파일을 만들어 놓기만 하고 내용은 일단 비워 둔다.

3. `echo.py` 파일은 다음과 같이 만든다.

```
# echo.py  
def echo_test():  
    print ("echo")
```

Package 기본요소 준비하기 [2/2]

4. render.py 파일은 다음과 같이 만든다.

```
# render.py
def render_test():
    print ("render")
```

5. 다음 예제들을 수행하기 전에 우리가 만든 game 패키지를 참조할 수 있도록 다음과 같이 도스 창에서 set 명령을 이용하여 PYTHONPATH 환경 변수에 C:/Python 디렉토리를 추가한다. 그리고 파이썬 인터프리터(Interactive shell)를 실행하자.

```
C:\> set PYTHONPATH=C:/Python
C:\> python
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:54:25) [MSC v.1900 64 bit (AM...
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Package안의 Function 실행

첫 번째는 echo 모듈을 import하여 실행하는 방법으로, 다음과 같이 실행한다.

```
>>> import game.sound.echo
>>> game.sound.echo.echo_test()
echo
```

```
C:/Python/game/__init__.py
C:/Python/game/sound/__init__.py
C:/Python/game/sound/echo.py
C:/Python/game/graphic/__init__.py
C:/Python/game/graphic/render.py
```

두 번째는 echo 모듈이 있는 디렉터리까지를 from ... import하여 실행하는 방법이다.

```
>>> from game.sound import echo
>>> echo.echo_test()
echo
```

세 번째는 echo 모듈의 echo_test 함수를 직접 import하여 실행하는 방법이다.

```
>>> from game.sound.echo import echo_test
>>> echo_test()
echo
```


__init__.py의 용도

__init__.py 파일은 해당 디렉터리가 패키지의 일부임을 알려주는 역할을 한다. 만약 game, sound, graphic 등 패키지에 포함된 디렉터리에 __init__.py 파일이 없다면 패키지로 인식되지 않는다.

(※ python3.3 버전부터는 __init__.py 파일 없이도 패키지로 인식이 된다(PEP 420 (<https://www.python.org/dev/peps/pep-0420/>)). 하지만 하위 버전 호환을 위해 __init__.py 파일을 생성하는 것이 안전한 방법이다.)

시험 삼아 sound 디렉터리의 **init.py**를 제거하고 다음을 수행해 보자.

```
>>> import game.sound.echo
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named sound.echo
```

sound 디렉터리에 __init__.py 파일이 없어서 임포트 오류(ImportError)가 발생하게 된다.

```
C:/Python/game/__init__.py
C:/Python/game/sound/__init__.py
C:/Python/game/sound/echo.py
C:/Python/game/graphic/__init__.py
C:/Python/game/graphic/render.py
```

__all__ variable 용도

외부의 python 프로그램
에서 import문장을 불렀
을때 →

```
>>> from game.sound import *
>>> echo.echo_test()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'echo' is not defined
```

```
C:/Python/game/__init__.py
C:/Python/game/sound/__init__.py
C:/Python/game/sound/echo.py
C:/Python/game/graphic/__init__.py
C:/Python/game/graphic/render.py
```

이렇게 특정 디렉터리의 모듈을 * 를 이용하여 import할 때에는 다음과 같이 해당 디렉터리의
__init__.py 파일에 __all__ 이라는 변수를 설정하고 import할 수 있는 모듈을 정의해 주어야

```
# C:/Python/game/sound/__init__.py
__all__ = ['echo']
```

여기서 __all__ 이 의미하는 것은 sound 디렉터리에서 * 기호를 이용하여 import할 경우 이곳
에 정의된 echo 모듈만 import된다는 의미이다.

위와 같이 __init__.py 파일을 변경한 후 위 예제를 수행하면 원하던 결과가 출력되는 것을 확인
할 수 있다.

```
>>> from game.sound import *
>>> echo.echo_test()
echo
```

Relative Package: Cascading Function Calls

만약 graphic 디렉터리의 render.py 모듈이 sound 디렉터리의 echo.py 모듈을 사용하고 싶다면

```
# render.py at graphic directory
```

```
from game.sound.echo import echo_test
```

```
def render_test():  
    print ("render")  
    echo_test()
```

```
# echo.py at echo directory
```

```
def echo_test():  
    print ("echo")
```

```
C:/Python/game/__init__.py
```

```
C:/Python/game/sound/__init__.py
```

```
C:/Python/game/sound/echo.py
```

```
C:/Python/game/graphic/__init__.py
```

```
C:/Python/game/graphic/render.py
```

from game.sound.echo import echo_test라는 문장을 추가하여 echo_test() 함수를 사용할 수 있도록 수정했다.

이렇게 수정한 후 다음과 같이 수행해 보자.

```
>>> from game.graphic.render import render_test
```

```
>>> render_test()
```

```
render
```

```
echo
```

Python Modules and Packages



- Utilizing Modules
- Python Package
- PyPI and Pip: Automatic Installation

PyPI: the Python Package Index

- A repository of open software for Python
 - 파이썬 개발자들은 자신들의 개발한 파이썬 모듈들을 PyPI에 upload
 - PyPI에 저장된 모듈들은 누구에게나 공개
 - PyPI 홈페이지에 접속하지 않고 pip을 통해서 손쉽게 원하는 모듈을 다운로드
- You can download extrinsic libraries and packages from PyPI by using **pip**

PyPI - the Python Package Index

The Python Package Index is a repository of software for the Python programming language. There are currently **74662** packages here.
To contact the PyPI admins, please use the [Support](#) or [Bug reports](#) links.

Not Logged In
[Login](#)
[Register](#)
[Lost Login?](#)
Use [OpenID](#) 
[Login with Google](#) 

Status
[Nothing to report](#)

Get Packages

To use a package from this index either "[pip](#) install *package*" ([get pip](#)) or download, unpack and "`python setup.py install`" it.

Package Authors

Submit packages with "`python setup.py upload`". The index [hosts package docs](#). You may also use the [web form](#). You must [register](#). Testing? Use [testpypi](#).

Infrastructure

To interoperate with the index use the [JSON](#), [OAuth](#), [XML-RPC](#) or [HTTP](#) interfaces. Use [local mirroring](#) or [caching](#) to make installation more robust.

Updated	Package	Description
2016-02-16	django-drynk 0.1.1	django-drynk gives you DRY natural keys.
2016-02-16	gbdx-tools 0.1.2	Python tools to order imagery and launch workflows on DigitalGlobe GBDX platform.
2016-02-16	pyextend 0.1.9	the python extend lib
2016-02-16	marketorestpython 0.1.10	Python Client for the Marketo REST API

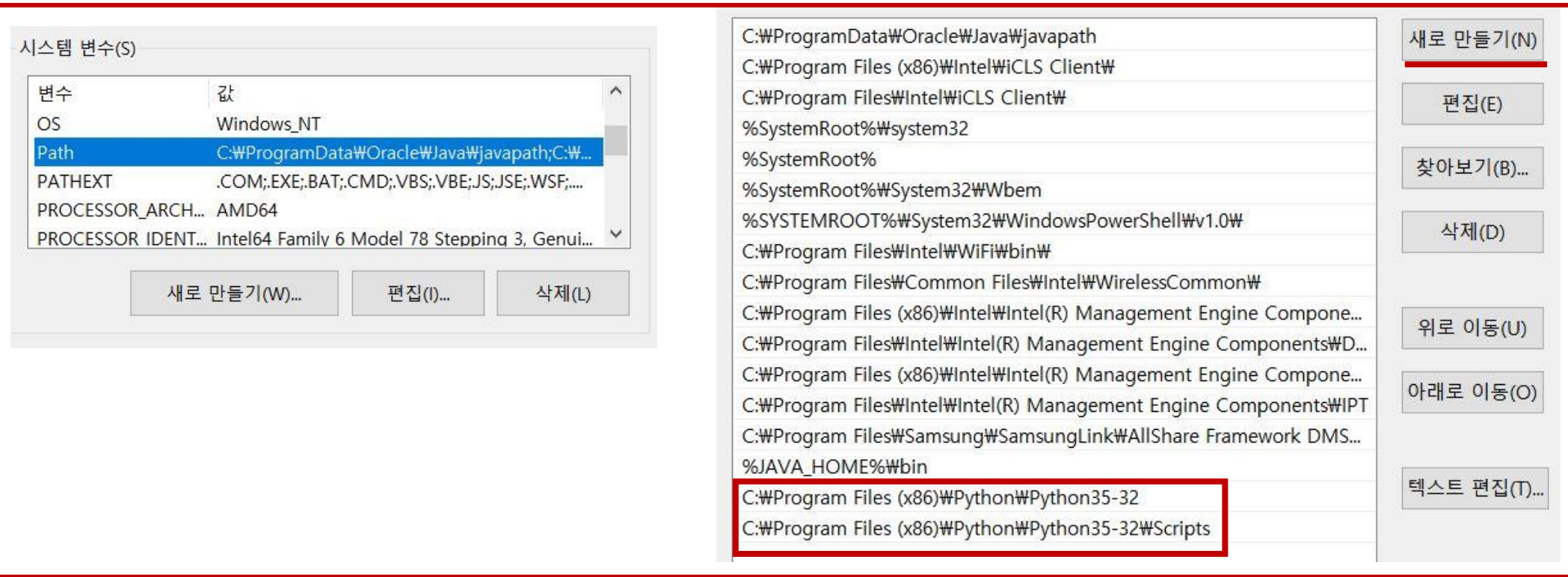
Extrinsic: 고유하지 않은, 비본질적인, 외부의
Intrinsic: 고유한, 본질적인

pip – Package Manager

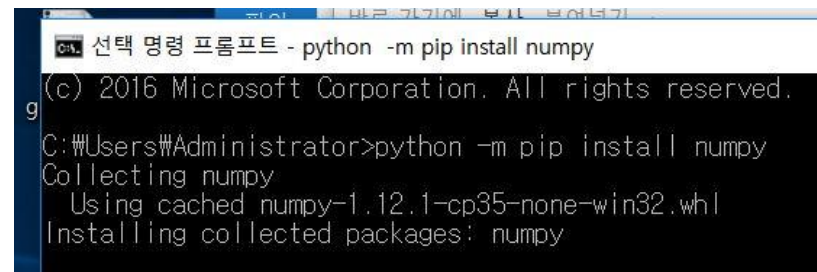
- 먼저 “pip” sw를 pc에 install 해야 한다

- Python 2.7.9 and later, and Python 3.4 and later include pip by default

- MS Window의 “제어판 ” 에 들어가서 “고급 시스템 설정” click하고 “환경변수”를 click하면



지금 부터는 cmd창에서 “python” 을 typing하면 작동을 하는 상태가 된다



```
python -m pip install numpy
```

```
python -m pip install Django
```

이제, pip이 자동으로 numpy or Django를 PyPI site에서 가져다 내PC에 설치해준다

Popular Python Libraries (Packages)

- **DB Binding Libraries**
 - PyMySQL module
 - MySQLdb module
 - SQLAlchemy module
- **Scientific Libraries**
 - SciPy module
- **Web Development Libraries**
 - Django module
 - Flask module
- **Machine Learning Libraries**
 - NumPy module
 - Pandas module
 - Matplotlib module
 - SKLearn Module
 - NLTK module

NumPy

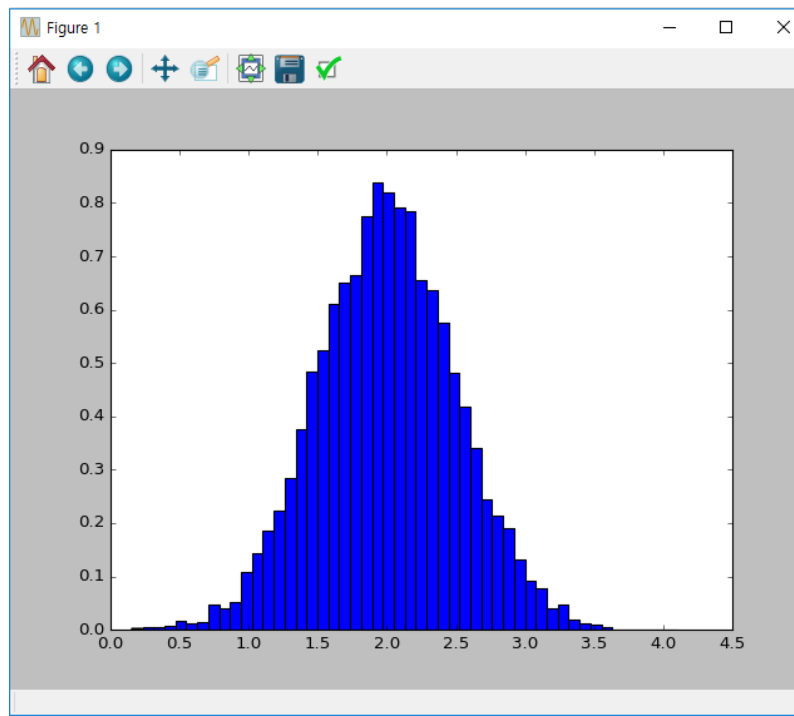
```
>>> import numpy as np
>>> a = np.arange(15).reshape(3, 5)
>>> a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>> a.shape
(3, 5)
>>> a.ndim
2
>>> a.dtype.name
'int64'
>>> a.size
15
>>> type(a)
<type 'numpy.ndarray'>
```

```
>>> b = np.array([6, 7, 8])
>>> b
array([6, 7, 8])
>>> type(b)
<type 'numpy.ndarray'>
```


matplotlib

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

# Build a vector of 10000 normal deviates with variance  $0.5^2$  and
mean 2
mu, sigma = 2, 0.5
v = np.random.normal(mu, sigma, 10000)
# Plot a normalized histogram with 50 bins
plt.hist(v, bins=50, normed=1)
plt.show()
```



NLTK(Natural Language Toolkit)

- <http://www.nltk.org>

NLTK 3.0 documentation

[NEXT](#) | [MODULES](#) | [INDEX](#)

Natural Language Toolkit

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to [over 50 corpora and lexical resources](#) such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active [discussion forum](#).

Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project.

NLTK has been called “a wonderful tool for teaching, and working in, computational linguistics using Python,” and “an amazing library to play with natural language.”

[Natural Language Processing with Python](#) provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more. The book is being updated for Python 3 and NLTK 3. (The original Python 2 version is still available at http://nltk.org/book_1ed.)

TABLE OF CONTENTS

[NLTK News](#)

[Installing NLTK](#)

[Installing NLTK Data](#)

[Contribute to NLTK](#)

[FAQ](#)

[Wiki](#)

[API](#)

[HOWTO](#)

SEARCH

Enter search terms or a module, class or function name.

NLTK(Natural Language Toolkit)

```
import nltk

sentence = "At eight o'clock on Thursday morning Arthur didn't  
feel very good."  
tokens = nltk.word_tokenize(sentence)  
tagged = nltk.pos_tag(tokens)  
print(tokens)  
for tag in tagged:  
    print(tag)
```

```
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning', 'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']  
('At', 'IN')  
('eight', 'CD')  
("o'clock", 'NN')  
('on', 'IN')  
('Thursday', 'NNP')  
('morning', 'NN')  
('Arthur', 'NNP')  
('did', 'VBD')  
("n't", 'RB')  
('feel', 'VB')  
('very', 'RB')  
('good', 'JJ')  
('.', '.')
```