

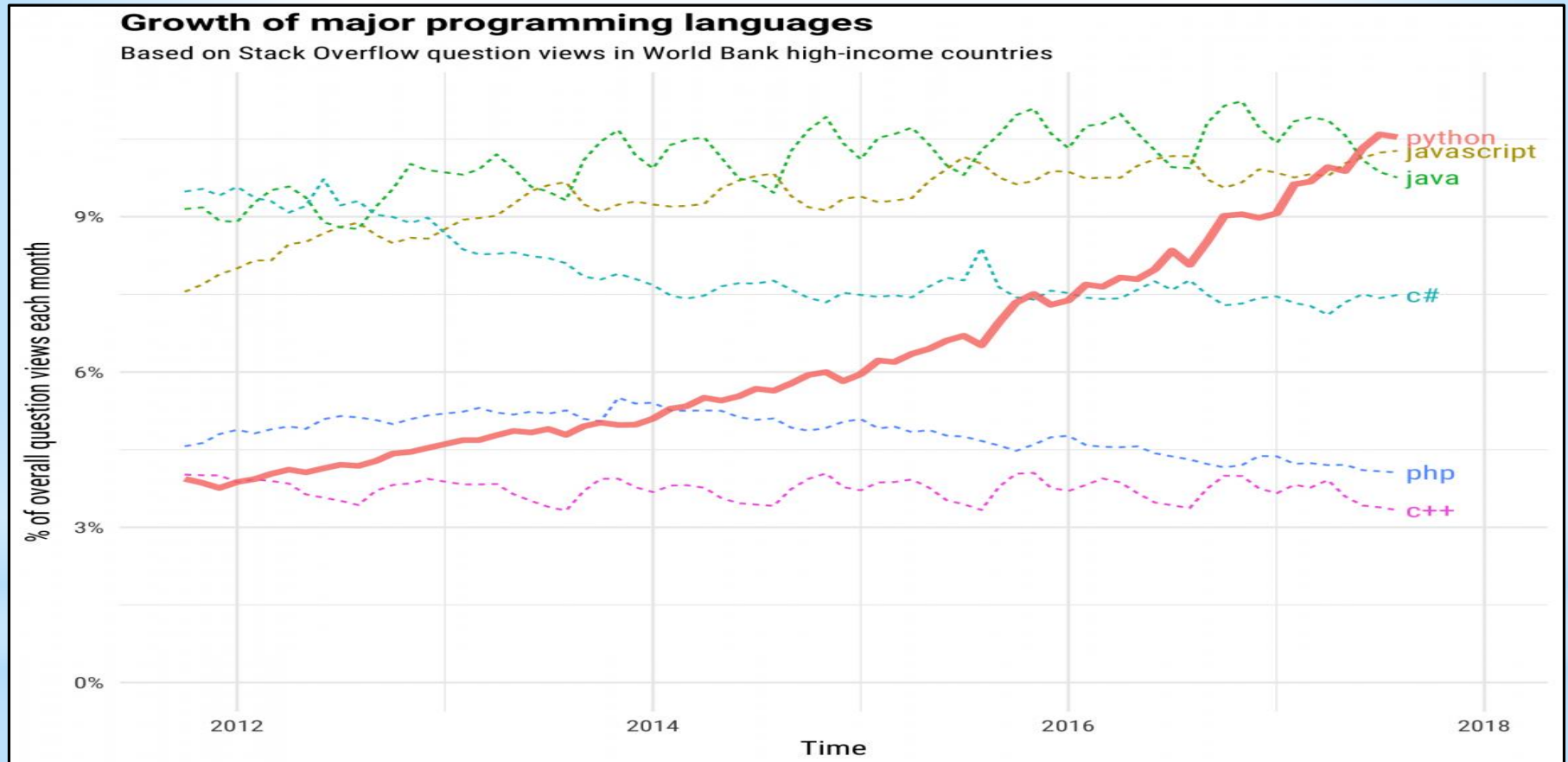
머신러닝/딥러닝을 위한

파이썬(Python)

– Data Type –

overview – Incredible Growth of Python

- 파이썬은 1991년 귀도 반 로섬(Guido van Rossum)이 만들어진 프로그래밍 언어로서 최근 가장 빠르게 확산되고 있는 추세임






overview – Incredible Growth of Python

➤ 파이썬 확산 배경

- ✓ 문법이 쉽고 직관적이어서 빠르게 배울 수 있음
(Life is too short, You need python)
- ✓ 뛰어난 확장성과 유연성
 - 파이썬은 이전에 웹 개발자와 시스템관리자들이 스크립트를 만들때 주로 사용했지만, 최근엔 다양한 라이브러리를 바탕으로 머신러닝, 딥러닝, 자연어처리와 함께 데이터 과학자 들에게도 폭 넓게 사용되고 있음
 - TensorFlow, Caffe, Keras 등의 유명 Machine Learning/Deep Learning 프레임워크에서 기본적으로 파이썬 API를 제공함

install – python

- 아나콘다 설치 (<https://www.anaconda.com/distribution/#download-section>)

 Windows |  macOS |  Linux

Anaconda 2019.10 for Windows Installer

Python 3.7 version

Download

64-Bit Graphical Installer (462 MB)
32-Bit Graphical Installer (410 MB)

Python 2.7 version

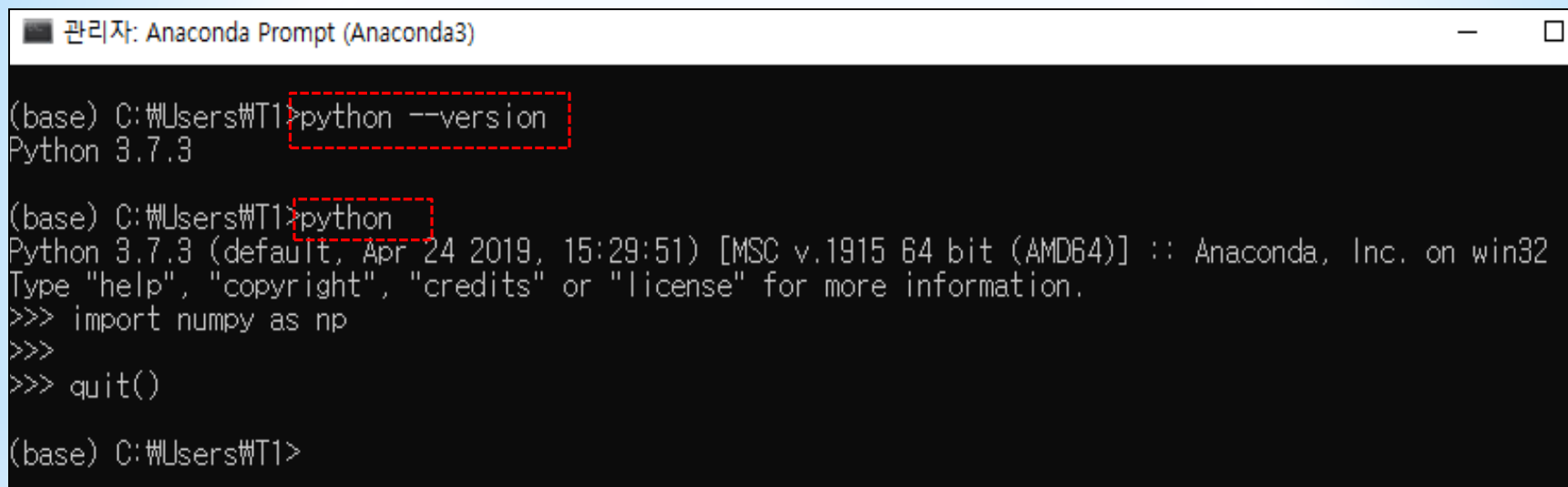
Download

64-Bit Graphical Installer (413 MB)
32-Bit Graphical Installer (356 MB)

※ 이전 버전의 아나콘다 리스트 (<https://repo.continuum.io/archive/index.html>)

install – python

- 터미널에서 `python --version` 명령으로 파이썬 버전 확인
- 터미널에서 `python` 입력으로 인터프리터 실행



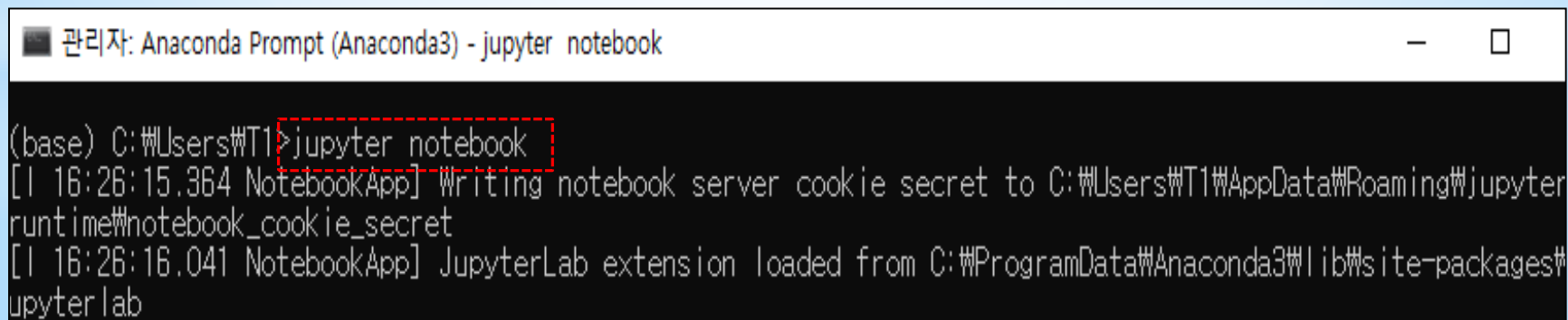
```
관리자: Anaconda Prompt (Anaconda3)

(base) C:\Users\T1>python --version
Python 3.7.3

(base) C:\Users\T1>python
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>>
>>> quit()

(base) C:\Users\T1>
```

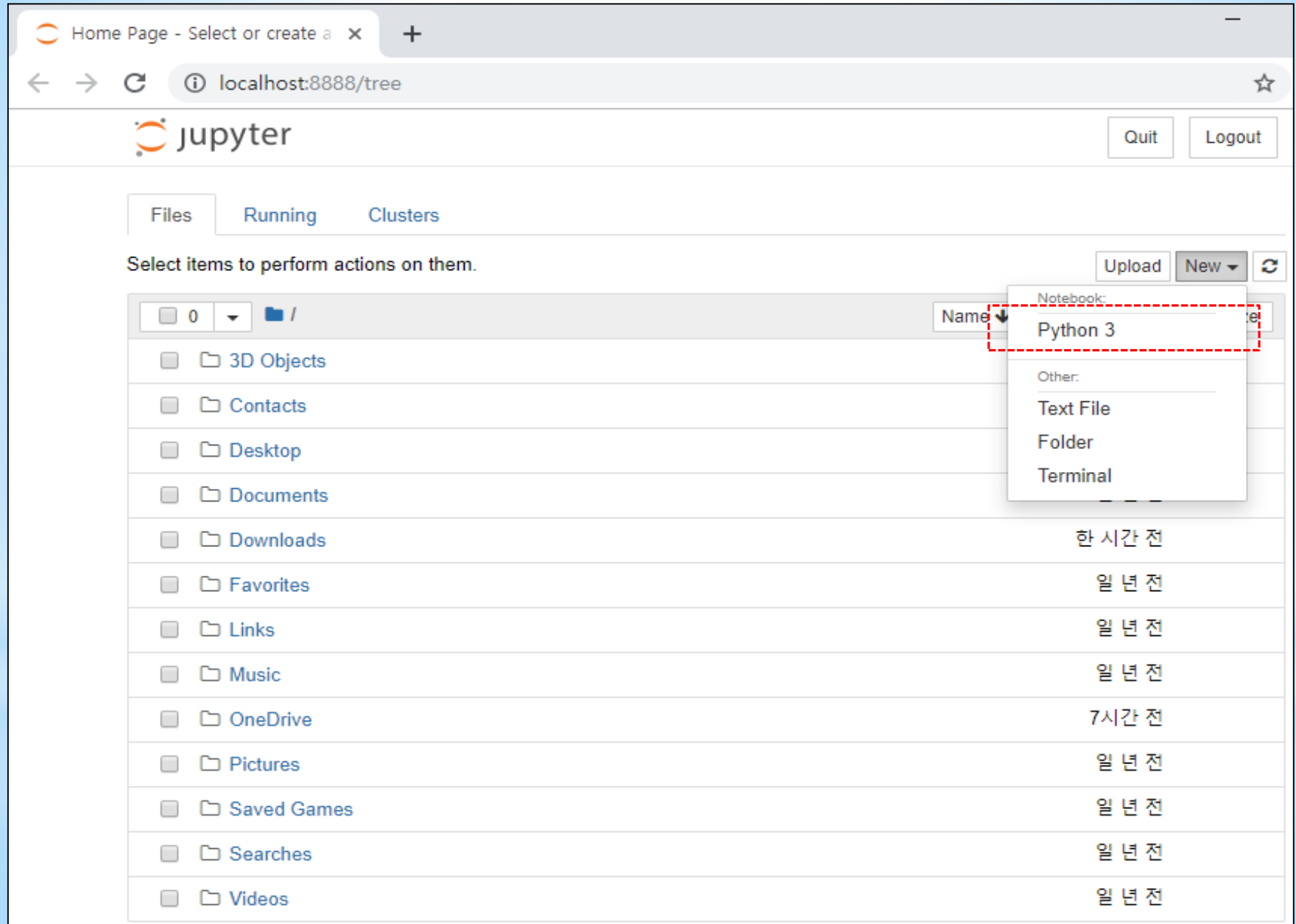
- 터미널에서 `jupyter notebook` 실행을 통하여 개발환경 실행



```
관리자: Anaconda Prompt (Anaconda3) - jupyter notebook

(base) C:\Users\T1>jupyter notebook
[I 16:26:15.364 NotebookApp] Writing notebook server cookie secret to C:\Users\T1\AppData\Roaming\jupyter
runtime\notebook_cookie_secret
[I 16:26:16.041 NotebookApp] JupyterLab extension loaded from C:\ProgramData\Anaconda3\lib\site-packages\
jupyterlab
```

jupyter notebook 실행 후 code 실행 및 comment 입력



The screenshot displays the JupyterLab web interface in a browser window. The address bar shows 'localhost:8888/tree'. The interface includes a top bar with the Jupyter logo, 'Quit', and 'Logout' buttons. Below this is a tabbed interface with 'Files', 'Running', and 'Clusters' tabs. The 'Files' tab is active, showing a file browser with a list of folders and files. A dropdown menu is open, showing options for creating a new notebook or file. The 'Python 3' option is highlighted.

Home Page - Select or create a x +

localhost:8888/tree

jupyter

Quit Logout

Files Running Clusters

Select items to perform actions on them.

Upload New

0 /

Name

Notebook:

Python 3

Other:

Text File

Folder

Terminal

Name	Last Modified
3D Objects	
Contacts	
Desktop	
Documents	
Downloads	한 시간 전
Favorites	일 년 전
Links	일 년 전
Music	일 년 전
OneDrive	7시간 전
Pictures	일 년 전
Saved Games	일 년 전
Searches	일 년 전
Videos	일 년 전

데이터타입 - list

- 리스트(list)는 다른 프로그래밍 언어의 '배열(array)' 과 비슷한 성질을 갖는 데이터타입

```
a = [10, 20, 30, 40, 50] # list 생성
```

- 리스트(list) 인덱스는 0 부터 시작하며, 파이썬에서는 마이너스(-) 인덱스를 지원하여 리스트의 마지막부터 역순으로 값을 참조할 수 있음
(머신러닝 코드에서 슬라이스와 함께 자주 사용됨)

index	0	1	2	3	4
value	10	20	30	40	50
index	-5	-4	-3	-2	-1

```
a = [ 10, 20, 30, 40, 50 ]
```

```
# 0, 1, 2... 인덱스는 리스트 처음부터 시작  
# -1, -2, ... 인덱스는 리스트 마지막부터 시작
```

```
print("a[0] ==", a[0], ", a[2] ==", a[2], ", a[4] ==", a[4])  
print("a[-1] ==", a[-1], ", a[-2] ==", a[-2], ", a[-5] ==", a[-5])
```

```
a[0] == 10 , a[2] == 30 , a[4] == 50  
a[-1] == 50 , a[-2] == 40 , a[-5] == 10
```

데이터타입 - list

- 리스트(list) 각 요소의 데이터타입을 다르게 해서 생성할 수 있으며, 리스트 안에 또 다른 리스트를 포함할 수도 있음

```
b = [ 10, 20, "Hello", [True, 3.14] ]   # list 생성
```

index	0	1	2	3
value	10	20	"Hello"	[True, 3.14]
index	-4	-3	-2	-1

```
b = [ 10, 20, "Hello", [True, 3.14] ]
```

```
# b[3] 또는 b[-1] 값은 [True, 3.14] 리스트 전체임
```

```
print("b[0] ==", b[0], ", b[2] ==", b[2], ", b[3] ==", b[3])  
print("b[-1] ==", b[-1], ", b[-2] ==", b[-2], ", b[-4] ==", b[-4])
```

```
b[0] == 10 , b[2] == Hello , b[3] == [True, 3.14]  
b[-1] == [True, 3.14] , b[-2] == Hello , b[-4] == 10
```

```
print("b[3][0] ==", b[3][0], ", b[3][1] ==", b[3][1])  
print("b[-1][-1] ==", b[-1][-1], ", b[-1][-2] ==", b[-1][-2])
```

```
b[3][0] == True , b[3][1] == 3.14  
b[-1][-1] == 3.14 , b[-1][-2] == True
```


데이터타입 - list

- 빈 리스트(list) 생성 후 append method를 이용하여 데이터 추가 (머신러닝 코드에서 정확도 계산, 손실함수 값 저장하기 위해 사용)

```
c = [ ]  
c.append(100), c.append(200), c.append(300)  
print(c)  
[100, 200, 300]
```

- 파이썬 리스트(list)에는 콜론(:)을 이용한 ‘슬라이싱’ 기능이 있음. 슬라이싱을 이용하면 범위를 지정해 부분 리스트를 얻을 수 있음 (머신러닝을 위해서는 반드시 알아야 하는 기능임)

index	0	1	2	3	4
value	10	20	30	40	50
index	-5	-4	-3	-2	-1

```
a = [ 10, 20, 30, 40, 50 ]
```

```
# a[0:2] => 인덱스 0부터 2-1 까지, a[1:] => 인덱스 1부터 끝까지  
# a[:3] => 인덱스 처음부터 3-1 까지, a[:-2] => 인덱스 처음부터 -2-1 까지  
# a[:] => 인덱스 처음부터 끝까지
```

```
print("a[0:2] ==", a[0:2], ", a[1:] ==", a[1:])  
print("a[:3] ==", a[:3], ", a[:-2] ==", a[:-2])  
print("a[:] ==", a[:])
```

```
a[0:2] == [10, 20] , a[1:] == [20, 30, 40, 50]  
a[:3] == [10, 20, 30] , a[:-2] == [10, 20, 30]  
a[:] == [10, 20, 30, 40, 50]
```

[참고] glob 라이브러리

- glob는 파일들의 목록을 뽑을 때 사용하는 유용한 라이브러리

glob.glob('*') ⇒ 현재 디렉토리의 모든 파일 목록을 리스트로 리턴

glob.glob('train/*') ⇒ train 디렉토리의 파일 목록을 리스트로 리턴

```
import glob
```

```
all_files = glob.glob('train/*')
```

```
print(all_files)
```

```
print(type(all_files))
```

```
txt_files = glob.glob('train/*.txt')
```

```
print(txt_files)
```

```
print(type(txt_files))
```







```
['train\\Agent.py', 'train\\cat.6.jpg', 'train\\cat.9.jpg', 'train\\dog.6316.jpg', 'train\\dog.6331.jpg', 'train\\hello.txt']
```

```
<class 'list'>
```

```
['train\\hello.txt']
```

```
<class 'list'>
```

train 디렉토리

이름	유형
 Agent.py	PY 파일
 cat.6	JPG 파일
 cat.9	JPG 파일
 dog.6316	JPG 파일
 dog.6331	JPG 파일
 hello	텍스트 문서

데이터타입 - tuple

- 튜플(tuple)은 리스트(list)와 거의 비슷하며 다른 점은 다음과 같음
 - 리스트는 []으로 둘러싸지만 튜플은 ()으로 둘러싼다.
 - 리스트 내의 원소를 변경할 수 있지만 튜플은 변경할 수 없다.

index	0	1	2	3	4
value	10	20	30	40	50
index	-5	-4	-3	-2	-1

```
a = ( 10, 20, 30, 40, 50 )
```

```
print("a[0] ==", a[0], ", a[-2] ==", a[-2], ", a[:] ==", a[:])
```

```
print("a[0:2] ==", a[0:2], ", a[1:] ==", a[1:])
```

```
a[0] == 10 , a[-2] == 40 , a[:] == (10, 20, 30, 40, 50)
```

```
a[0:2] == (10, 20) , a[1:] == (20, 30, 40, 50)
```

```
a[0] = 100    # a[0] 값을 100 으로 변경하려고 하기 때문에 에러발생
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-6-f5f926370770> in <module>()  
----> 1 a[0] = 100
```

```
TypeError: 'tuple' object does not support item assignment
```

데이터타입 - dictionary

- 딕셔너리(dictionary)는 다른 프로그래밍 언어의 '해시(hash)' 또는 '맵(map)'과 구조가 비슷
- 딕셔너리는 키(key)와 값(value)을 한 쌍으로 해서 데이터를 저장함

key	value
"KIM"	90
"LEE"	85
"JUN"	95

```
score = { "KIM":90, "LEE":85, "JUN":95 }    # dictionary 생성
```

```
print("score['KIM'] ==", score['KIM'])    # 원소 접근
```

```
score['KIM'] == 90
```

```
score['HAN'] = 100    # 새 원소 추가
```

딕셔너리는 입력한 순서대로 데이터가 들어가는 것이 아니므로 주의해야함

```
print(score)
```

```
{'KIM': 90, 'LEE': 85, 'HAN': 100, 'JUN': 95}
```

딕셔너리 key, value, (key, value)

```
print("score key ==", score.keys())
```

```
print("score value ==", score.values())
```

```
print("score items ==", score.items())
```

```
score key == dict_keys(['KIM', 'LEE', 'HAN', 'JUN'])
```

```
score value == dict_values([90, 85, 100, 95])
```

```
score items == dict_items([('KIM', 90), ('LEE', 85), ('HAN', 100), ('JUN', 95)])
```

데이터타입 - string

- 파이썬 문자열(string)은 홑따옴표('') 또는 쌍따옴표("")를 사용해서 생성
- 문자열 내의 각각의 값 또한 문자열로 인식되며, split() / join() / strip() 함수 등은 머신러닝 코드에서 문자열 데이터 전처리(pre-process) 하기 위해 자주 사용됨

index	0	1	2	3	4	5
value	`A`	`7`	`3`	``,`	`C`	`D`
index	-6	-5	-4	-3	-2	-1

```
a = 'A73,CD'
```

```
a[1] # a[1] 은 숫자 7이 아닌 문자열 7
```

```
'7'
```

```
a = a + ', EFG' # + 연산자 사용
```

```
a
```

```
'A73,CD, EFG'
```

```
# split() 메서드는 특정 separator를 기준으로 문자열을 분리하여 list 리턴
```

```
b = a.split(',')
```

```
print(b)
```

```
['A73', 'CD', ' EFG']
```

데이터타입 - string

- join(리스트 변수)는 구분자(delimiter) 문자열과 문자열 list 요소를 연결하여 새로운 문자열로 만들어 주는 함수

```
delimiter = '-'  
new_str = delimiter.join(['apple', 'orange', 'grape'])  
print(new_str)
```

apple-orange-grape

```
delimiter = ', '  
new_str = delimiter.join(['apple', 'orange', 'grape'])  
print(new_str)
```

apple, orange, grape

- strip()은 양쪽 공백을 모두 제거하는 함수

```
str = '   Hello   '  
print(str.strip())  
print(str)
```

Hello
Hello

```
str = ', Hello.. '  
print(str.strip(',.. '))  
print(str)
```

Hello
, Hello..

useful function – type(), len()

- type(data)는 입력 data의 데이터타입을 알려주는 함수
- len(data)은 입력 데이터의 길이(요소의 개수)를 알려주는 함수.

```
a = [ 10, 20, 30, 40, 50 ]
b = ( 10, 20, 30, 40, 50 )
c = { "KIM":90, "LEE":80 }
d = 'Seoul, Korea'
e = [ [100, 200], [300, 400], [500, 600] ]

print(type(a), type(b), type(c), type(d), type(e))

print(len(a), len(b), len(c), len(d), len(e))

<class 'list'> <class 'tuple'> <class 'dict'> <class 'str'> <class 'list'>
5 5 2 12 3
```

[예제 1] 화면에 John's House을 출력하라. (중간에 '가 있음에 주의)

[예제 2] 다음의 print 문을 통해 \n, \t 기능을 확인하시오

```
print('Hello.\nBasic\t\tLecture')
```

[예제 3] license_plate 문자열이 다음과 같을때, 뒤 4자리만 출력하시오

```
license_plate = "24가 2210"
```

[예제 4] string = "홀짝홀짝홀짝" 에 대하여, '홀홀홀' 만 출력하시오

[예제 5] 다음의 Error 원인과 해결방안을 찾으시오

```
list_A = [ 1, 2, 3, 4, 5 ]

len = len(list_A)
print(len)
5

list_B = [ 1, 2, 3 ]

len = len(list_B)
print(len)

-----
-
TypeError                                Traceback (most recent call last)
<ipython-input-4-43528205170e> in <module>
----> 1 len = len(list_B)
      2
      3 print(len)

TypeError: 'int' object is not callable
```

[예제 6] 아래 코드의 실행 결과를 예상하시오

```
str = 'Hello'
str[0] = 'K'
print(str)
```

[예제 7] 슬래시 ('/')로 구분되어 다음의 문자열에 대해서,

```
string = "Hello/Machine/Deep/Learning"
```

[7.1] 슬라이싱만을 이용하여 다음 리스트를 만드시오

```
list_A = [ 'Hello', 'Machine', 'Deep', 'Learning' ]
```

[7.2] 슬라이싱과 list append method를 이용하여 다음 리스트를 만드시오

```
list_B = [ 'Hello', 'Machine', 'Deep', 'Learning' ]
```

[7.3] string의 split method를 이용하여 다음 리스트를 만드시오

```
list_C = [ 'Hello', 'Machine', 'Deep', 'Learning' ]
```

[예제 8] 변수 A에 'hello world'라는 문자열이 저장되어 있을때 슬라이싱 기능과 문자열 + 연산자만을 이용하여 A 값을 'hi world'로 변경하기

[예제 9] 문자열 `str = "AB,CDE:FG,WX,Y".split(',')` 에서, 다음을 구하시오

(9.1) `str`

(9.2) `str[-2]`

(9.3) `str[0][:-1]`

(9.4) `str[2:]`

(9.5) `str[:][:]`

[예제 10] 다음과 같이 `files` 정의 되었을 때, `new_var` 값을 구하시오

```
files = 'tmp/cat.9.jpg:python.1.py:temp/hello.6316.txt:test.6331.docx'
```

```
new_var = [ files.split(':')[2].split('.')[1].strip() ]
```