Python

0.start

python.org

Downloads → All Releases → Release version 선택 → os에 맞는 file download

python 2.* : 2020.04 지원 종료

os X 에는 python 2 버전이 설치되어 있음.

- python3, pip3 명령어 사용

0.start

CPython	c언어로 작성된 인터프리터. 일반적인 python을 의미
Stackless Python	c언어의 스택을 사용하지 않는 인터프리터
Jython	JVM 용 인터프리터. JPython이라고도 함
Iron Python	.Net 용 인터프리터
Pypy	Python으로 작성된 인터프리터

0.start

Interactive shell 지원

- REPL (Read Eval Print Loop)

body 대신 들여쓰기

- 2space or 4space or tab

플랫폼에 독립적

python coding style

들여쓰기 : 4 space

주석은 별도의 줄

최상위 수준 함수와 클래스는 빈줄*2

79자를 넘지 않게 줄바꿈(80자 제한)

연산자와 콤마 뒤에 space

독스트링(문서화 문자열) """ 사용

클래스와 메서드는 빈줄*1

0.start

* python 내장 가상환경 모듈 (venv)

python -m venv 가상환경이름

* anaconda 이용

conda create -n 가상환경이름 python=버전 모듈

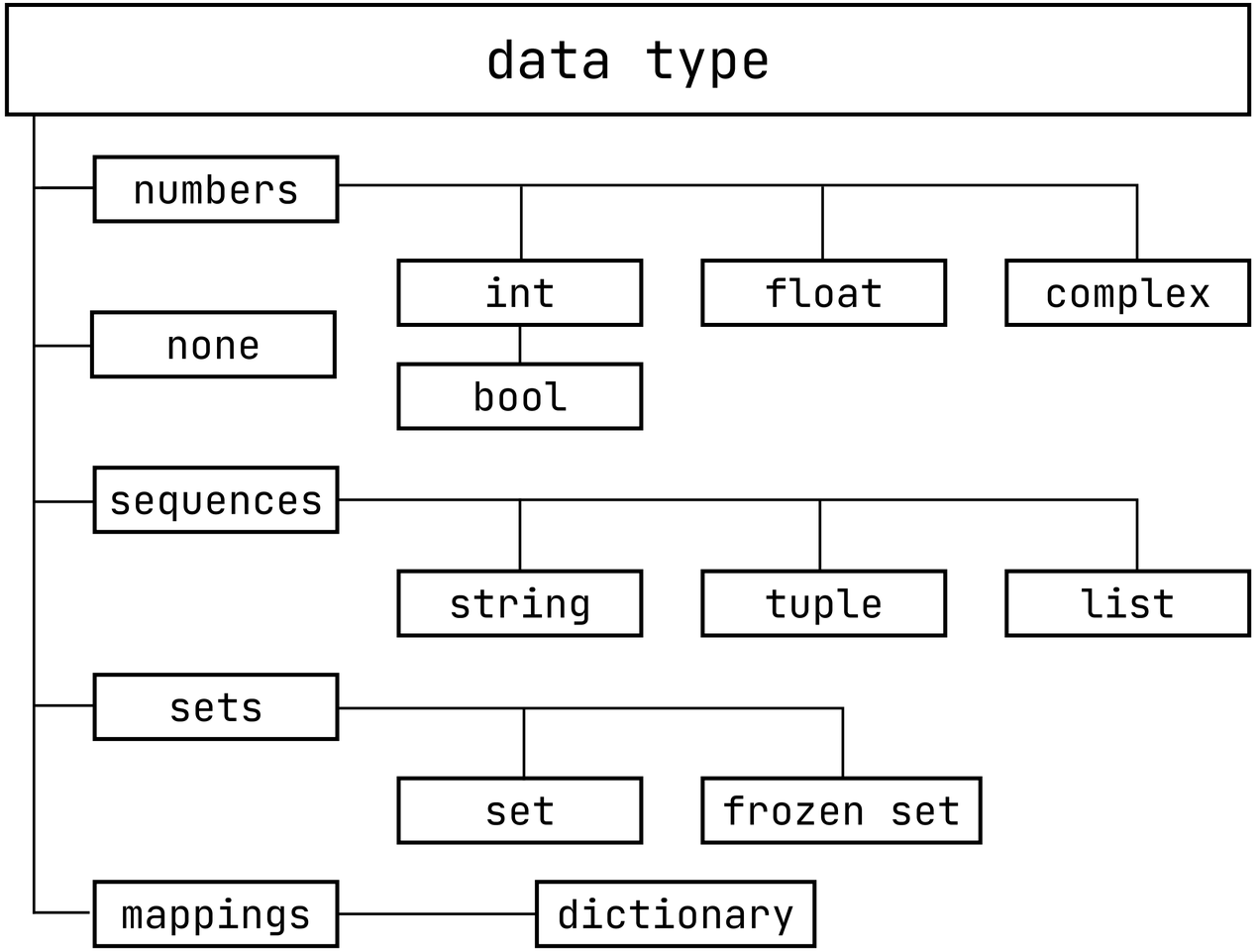
1.type

구조

변수 = 값

* 타입 = 값의 형태

* 모든 타입은 객체(object)로 이루어져 있음



1.type

numbers

int	정수	binary (2)	0b
float	실수	octal (8)	0o
complex	복소수	hex (16)	0x
bool	논리 (True / False)		

* complex는 공백 불가!

1.type

none

None 값이 없음

- 내장 상수 (NoneType 객체)
- singleton으로 생성됨
- 논리적인 값을 확인할 때 False로 취급

* undefined (정의되지 않음) 와 다름!

1.type

객체가 가진 요소(값)이 연속적(sequence) 으로 연결되어 있음

string	single quotation (‘ ‘) double quotation (“ “) *3 raw string escape sequence
list	[] 순서 O, 중복 O
tuple	{} 순서 O, 중복 O, 수정 X

1.type

escape

\n	linefeed
\r	carriage return
\t	tab
\b	backspace
\\	\
\0000	None (Null)
\’	’
\”	”

1.type

set {}

순서 X, 중복 X

집합을 의미 (합집합, 교집합, 차집합, ...)

mutable

frozen set {}

순서 X, 중복 X, 수정 X

immutable

1.type

mapping

dictionary {key: value, key: value, ...}

순서 O, 중복 (key: X, value: O)

key 를 통해 value 에 접근 가능

key 에는 mutable 한 값이 들어갈 수 없음

* python 3.7 이후부터 입력 순서를 가진다

1.type

immutable/mutable

immutable	numbers, string, tuple, frozen set
	값(주소값)이 변하지 않음
	call by value
mutable	list, set, dictionary
	값이 변함
	call by reference

1.type

print

% values

```
print('%d' % 5)
```

format class (str.format)

```
name = 'python'
print('{}'.format(name))
```

formatted string literals

```
print('{name}')
```

1.type

format	
%s	string
%c	character
%f	floating point
%d	10
%b	2
%o	8
%x	16
%%	%

2.operator

산술연산

+		a + b
-		a - b
*		a * b
**	거듭제곱	a ** b
/	나누기 (float)	a / b
//	몫 (int)	a // b
%	나머지	a % b

2.operator

비교연산

==

!=

>

>=

<

<=

* 결과 : True | False

is

is not

and

or

not

in

not in

멤버연산

2.operator

범위연산

`range(n)`	`0 ~ n - 1`
`range(start, end)`	`start ~ end -1`
`range(start, end, step)`	`start, start + step, ..., end - 1`

2.operator

범위연산

```
 +---+---+---+---+---+---+
 | p | y | t | h | o | n |
 +---+---+---+---+---+---+
 0   1   2   3   4   5   6
-6  -5  -4  -3  -2  -1
```

[n]

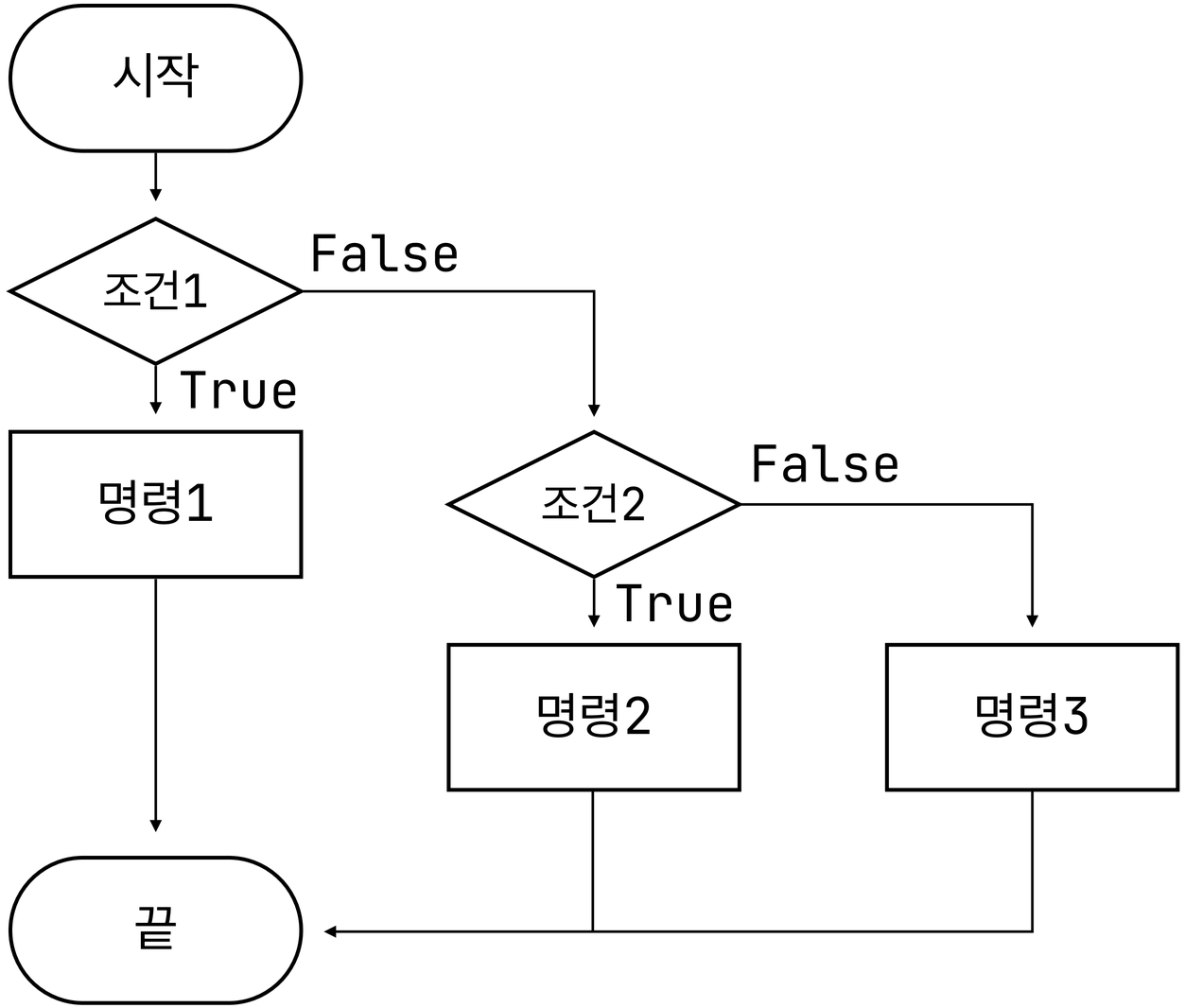
[start, end]

[start, end, step]

```
val = “python”

val[2]              python

val[-2]             python

val[2 : ]           python

val[ : -2]          python

val[2 : 5]          python

val[0 : 6 : 2]      python

val[ : : -2]        nohtyp
```

3.control

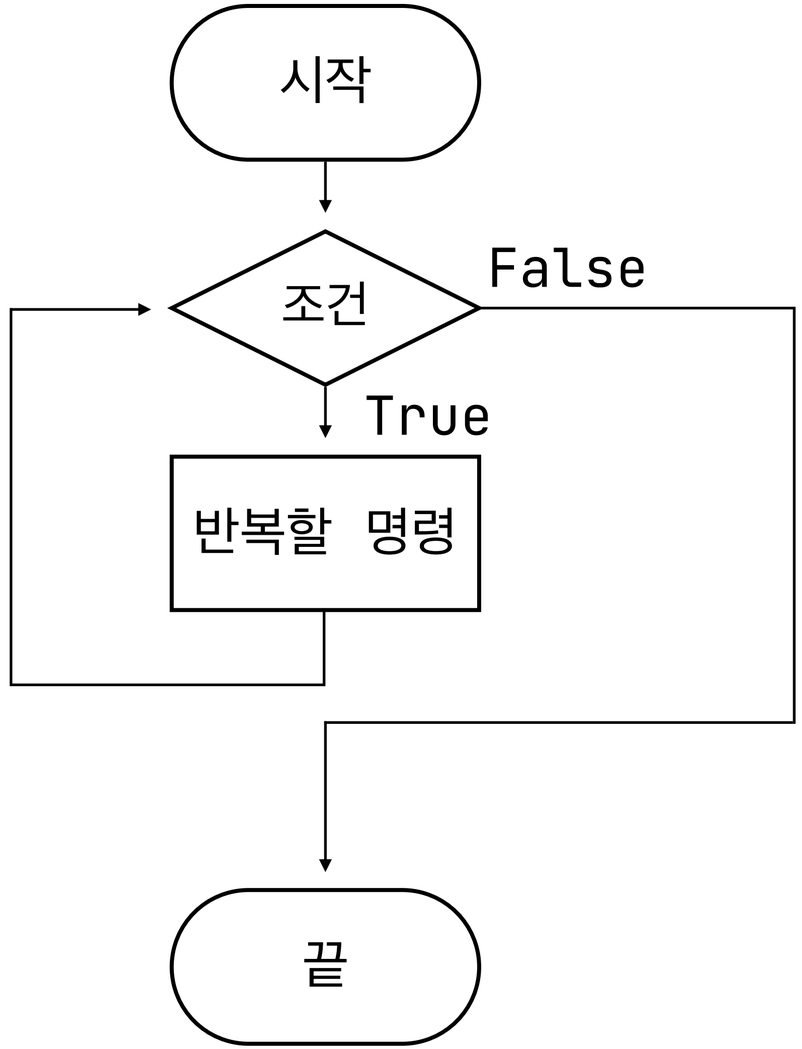

```
if 조건1 :
    명령1
elif 조건2 :
    명령2
else :
    명령3
```



3.control


```
while 조건:
    반복할 명령
```

* do while 없음

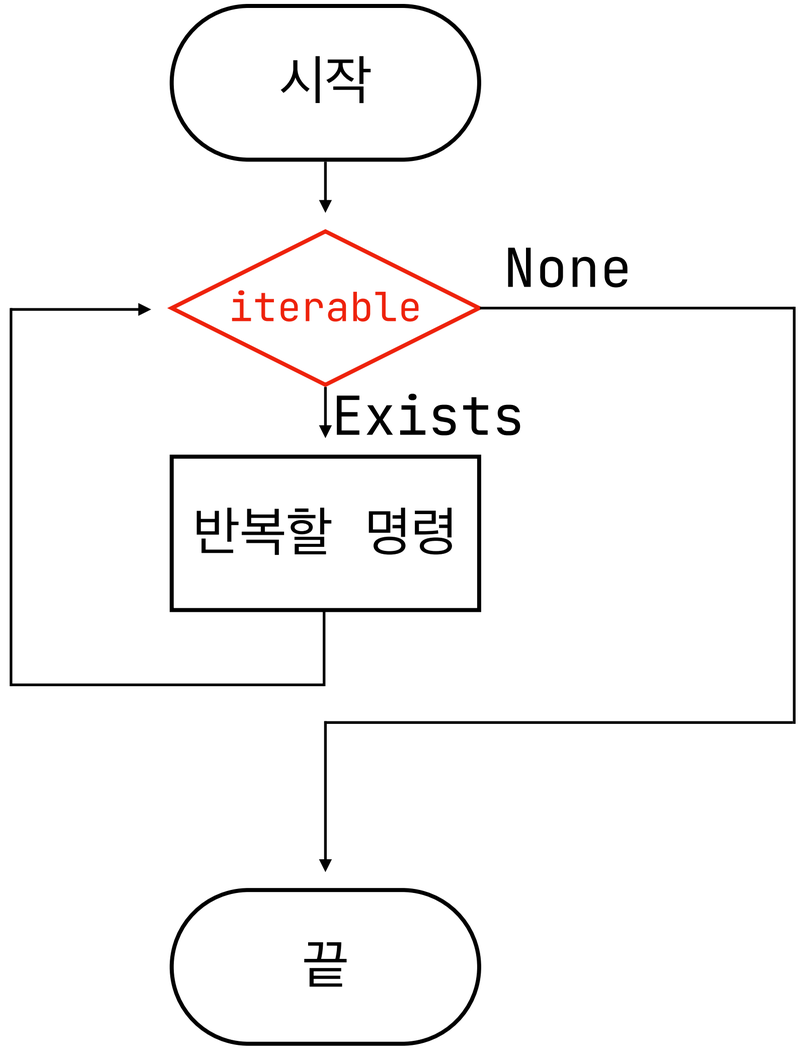


3.control


```
for item in iterable :
    반복할 명령
```

* iterable 한 객체의 요소를 순서대로 변수에 저장

* enumerate : index 와 item return



3.control

iterable

반복가능한 객체

sequences, dict, files,
__iter__() or __getitem__() 구현한 객체 등

iterator

data stream (데이터의 흐름)을 표현한 객체

__next__() or next()를 사용하여 iterator 내부의 값을 호출 가능

더 이상 값이 없을 때는 StopIteration 예외 발생

* Lazy Evaluation

4.function


```
def 함수명(parameter):
    명령
```

*args	여러 개의 arguments를 입력하여 호출 가능
**kwargs	keyword arguments (k=v)를 입력하여 호출 가능

* parameter (매개변수)

  arguments (인수)

4.function

변수 = lambda parameter: 명령

- 익명 함수 표현식

- 함수를 간편한 수식으로 표현

```
def test01(x):
    return x + 10

print(test01(10))
```

```
test02 = lambda x: x + 10

print(test02(10))
```

4.function

closure

```
def 외부함수(param):
    명령어

    def 내부함수(param):
        명령어
```

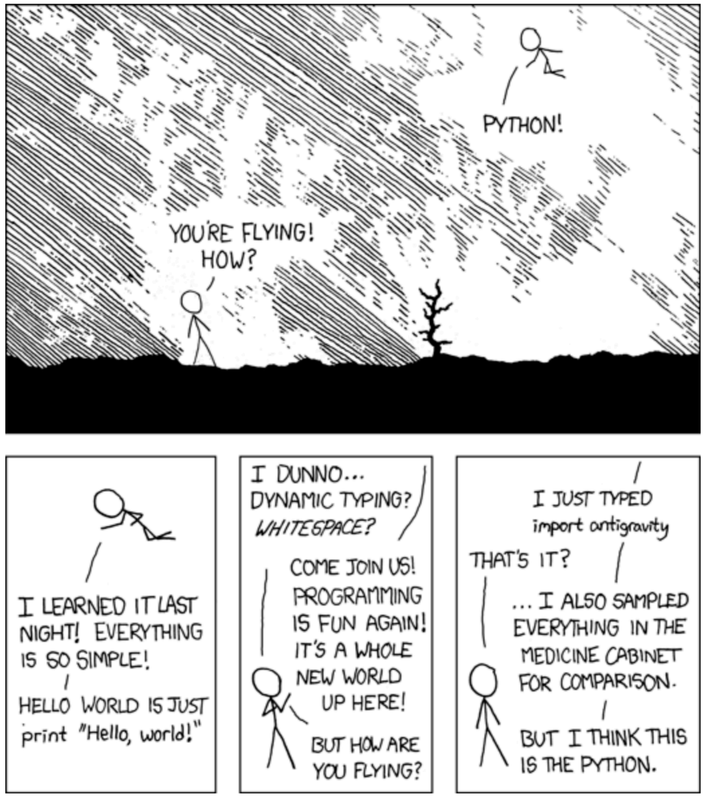
* 내부함수는 외부함수의 변수를 참조 할 수 있다 (수정 불가)
  nonlocal : 수정하고 싶을 때 사용하는 키워드 (변수의 scope 설정)

5.module

Pypi (Python Package Index)

python package (library, module) 관리 프로그램

pip install 패키지명



import antigravity

* 패키지 사용 방법

import 패키지명

import 패키지명 as 별칭

from 패키지명 import 클래스 | 함수 | 변수

6.io


```
file = open('경로', 'mode')

명령

file.close()
```

```
with open('경로', 'mode') as file:
    명령
```

* with open : file 객체 자동 close()

r	읽기
w	덮어쓰기
a	파일 끝에 쓰기
x	새 파일 쓰기 (파일이 있으면 에러)
t	text
b	binary
mode+	모드 연결

7.exception

예외처리 프로그램의 비 정상적 종료 방지

```
try:
    예외가 발생할 수 있는 명령

except 예외:
    해당 예외가 발생할 경우 처리할 명령

finally:
    예외의 발생유무에 관계없이 무조건 처리할 명령
```

8.class

Object Oriented Programming

abstraction	추상화
inheritance	상속
polymorphism	다형성
encapsulation	캡슐화

8.class

구조

constructor — 객체 생성
field 초기화

field — 속성

method — 기능

@staticmethod

@classmethod

@property

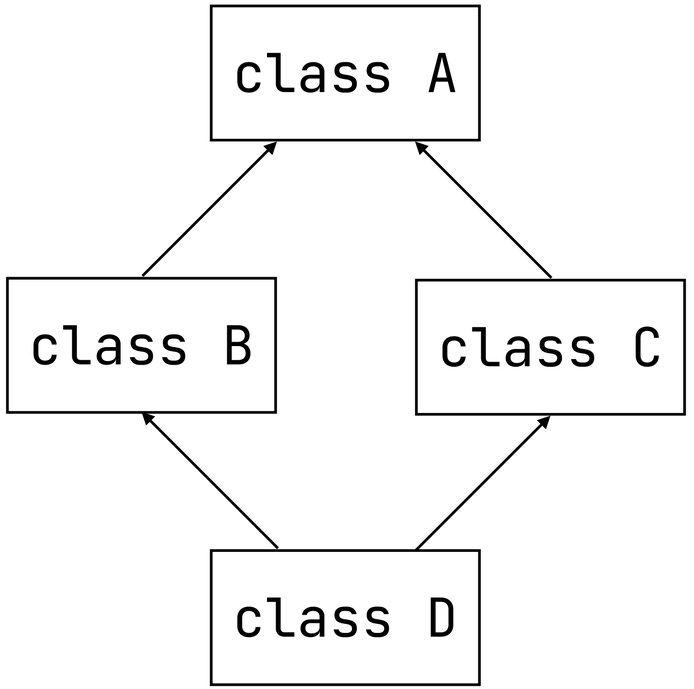
@필드.setter

```
class 클래스 이름:

    변수 # class 변수

    def __init__(self):
        self.변수 # instance 변수

    def 함수명(param):
        명령
```

8.class


```
class 자식클래스(부모클래스):
    ...
```

- diamond inheritance




```
class A:
    ...

clsss B(A):
    ...

clsss C(A):
    ...

clsss D(C, B):
    ...
```

* Method Resolution Order (상속 순서 확인)

```
D.mro()
D.__mro__
```

8.class


```
from abs import *

class 클래스 이름(metaclass=ABCMeta):     # abstract class

    @abstractmethod
    def 함수 이름(param):                 # abstract method
        pass
```

* 추상 클래스는 인스턴스화 불가!

9.decorator

생성

```
def 데코레이터 이름(func):

    def 실제 기능 함수(param):
        func()

    return 실제 기능 함수
```

```
class 데코레이터 이름:

    def __init__(self, func):
        self.func = func


    def __call__(sef):
        self.func()
```

* 클래스나 함수 위에 @데코레이터 로 사용

10.generator

yield 값	next()가 호출될 시 해당 값 return
yield from	다른 generator 호출
next()	generator 안의 값 호출 __next__와 같음

11.thread

동시처리

program	명령어와 데이터의 집합
process	실행되어 메모리에 적재된 program (job, task)
thread	process 내에서 실제로 작업을 수행

```
import threading

import multiprocessing
```

12.coroutine


```
변수 = (yield 변수)

next(coroutine)

coroutine.send()
```

* generator는 next()를 반복 호출하지만, coroutine은 한 번만 호출!

13.async

async def

async for　　__aiter__(), __anext__() 구현

async with　　__aenter__(), __aexit__() 구현

await coroutine | task | future

* ~~@asyncio.coroutine : deprecated~~

14.crawling

scraping

requests	url 요청 및 응답
beautifulsoup	document parsing
selenium	automates browsers webdriver를 사용하여 browser 자동화

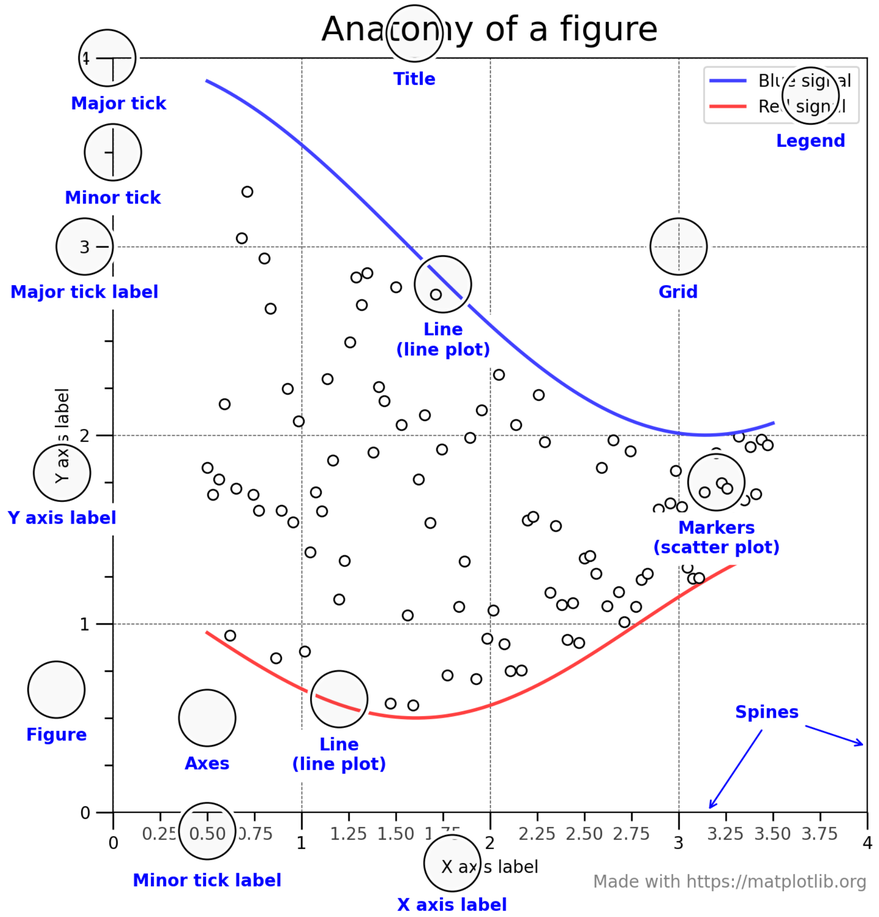
15.visual

matplotlib

```
pip install matplotlib
```

시각화 (그래프) 라이브러리

```
from matplotlib.pyplot as plt

fig = plt.figure()

...

plt.show()
```

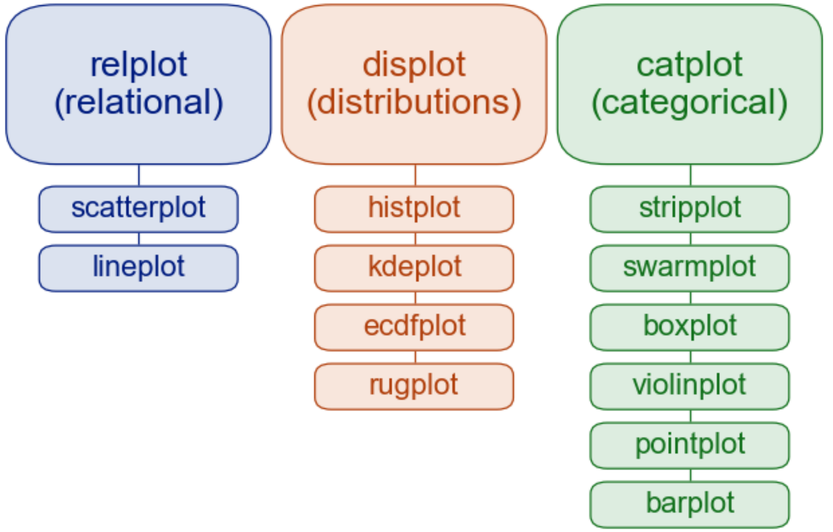


15.visual

pip install seaborn

- matplotlib 기반의 시각화 라이브러리

- datasets 내장




```
import seaborn as sns

sns.load_datasets(name)
```

15.visual

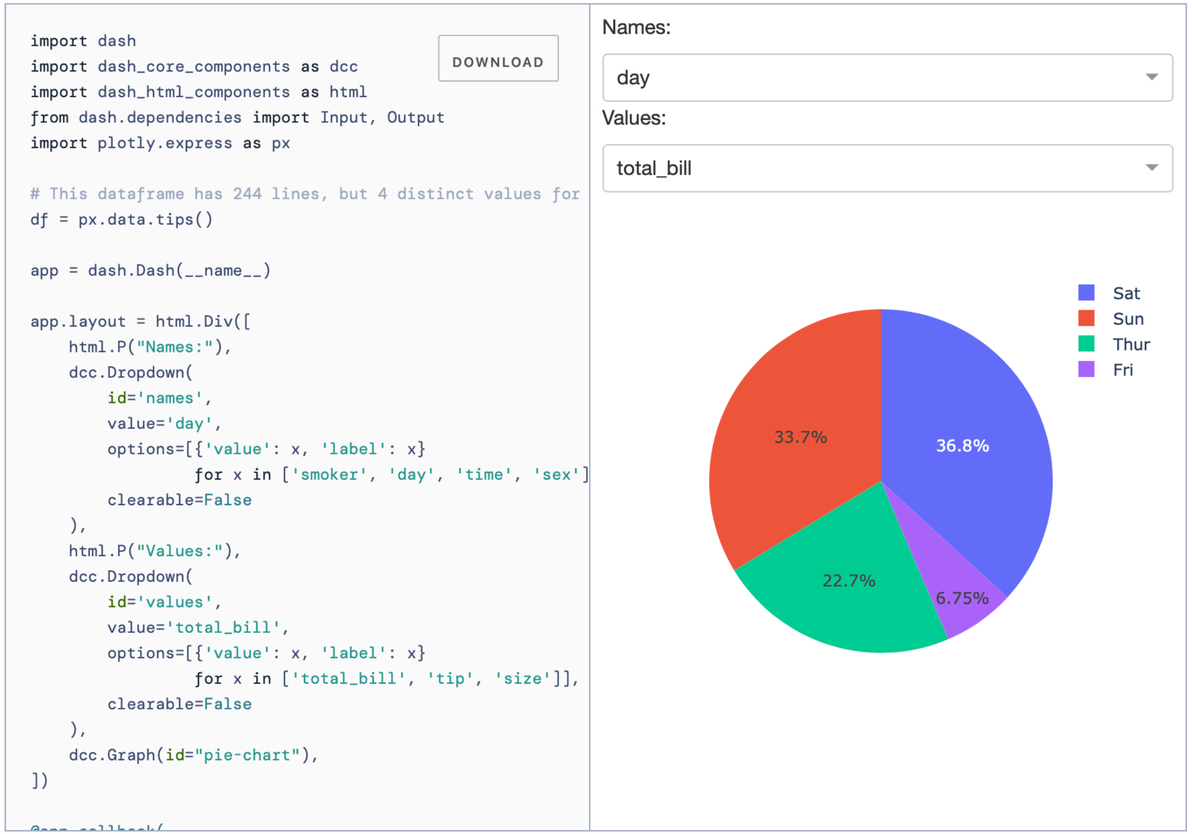

```
pip install plotly
```

- dash 라이브러리와 함께 사용해 interactive dashboard 작성 가능

```
import plotly.express as px
```

```
import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output
import plotly.express as px

# This dataframe has 244 lines, but 4 distinct values for
df = px.data.tips()

app = dash.Dash(__name__)

app.layout = html.Div([
    html.P("Names:"),
    dcc.Dropdown(
        id='names',
        value='day',
        options=[{'value': x, 'label': x}
                 for x in ['smoker', 'day', 'time', 'sex']]
        clearable=False
    ),
    html.P("Values:"),
    dcc.Dropdown(
        id='values',
        value='total_bill',
        options=[{'value': x, 'label': x}
                 for x in ['total_bill', 'tip', 'size']],
        clearable=False
    ),
    dcc.Graph(id="pie-chart"),
])
```

DOWNLOAD



15.visual

pip install folium

- 지도 시각화 라이브러리

```
import folium

folium.Map()

folium.Marker()
```