Python

Python is programming language that lets you work quickly ant integrate systems more effectively - python

0.start

install

python.org

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

anaconda

* os x 에는 기본으로 python 2버전이 설치되어 있음!

0.start

종류

CPython	c언어로 작성된 인터프리터. 일반적인 python
Stackless Python	c언어의 스택을 사용하지 않는 인터프리터
Jython	jvm 용 인터프리터. JPython이라고도 함
Iron python	.net 용 인터프리터
Pypy	python으로 작성된 인터프리터

0.start

특징

interactive shell 지원

REPL (Read Eval Print Loop)

body 대신 들여쓰기

space 2 / space 4 / tab 1

플랫폼에 독립적

PEPs

들여쓰기는 공백 4개

주석은 별도의 줄

최상위 수준 함수와 클래스는 빈줄 2개

클래스와 메서드는 빈줄 1개

한 줄에 80자 제한

연산자와 콤마 뒤엔 공백

0.start

가상환경

venv

```
python -m venv 가상환경이름
```

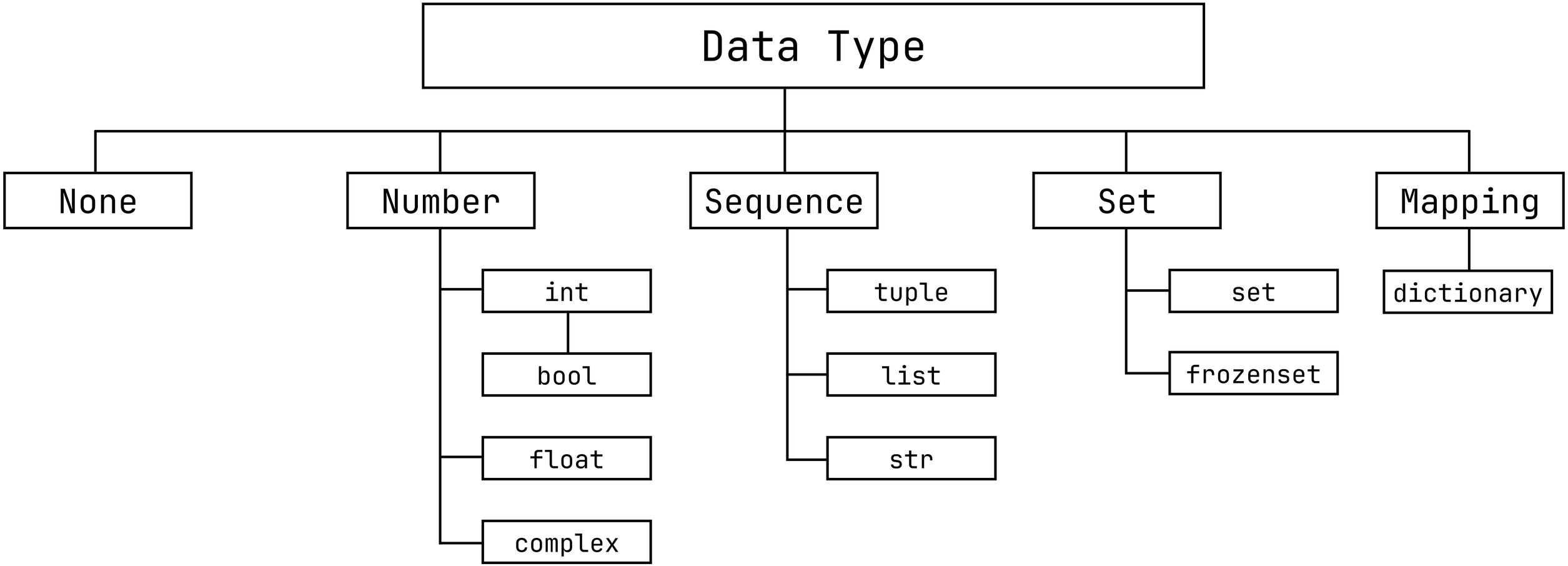
anaconda

```
conda create -n 가상환경이름
```

- `conda activate 가상환경이름`
- `conda deactivate`

1.type

구조

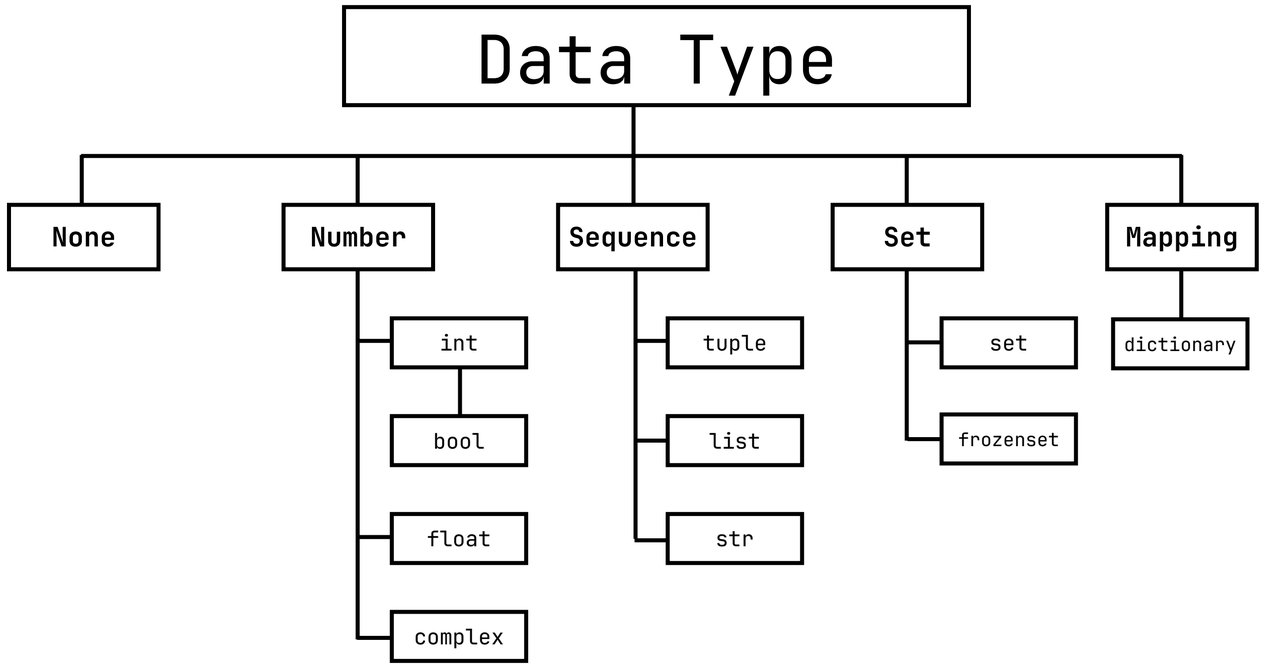

Data Type
None
Number
int
bool
float
complex
Sequence
tuple
list
str
Set
set
frozenset
Mapping
dictionary

1.type

구조

type 값의 형태

literal 값 자체



* 모든 타입은 객체(object)로 이루어져 있음!

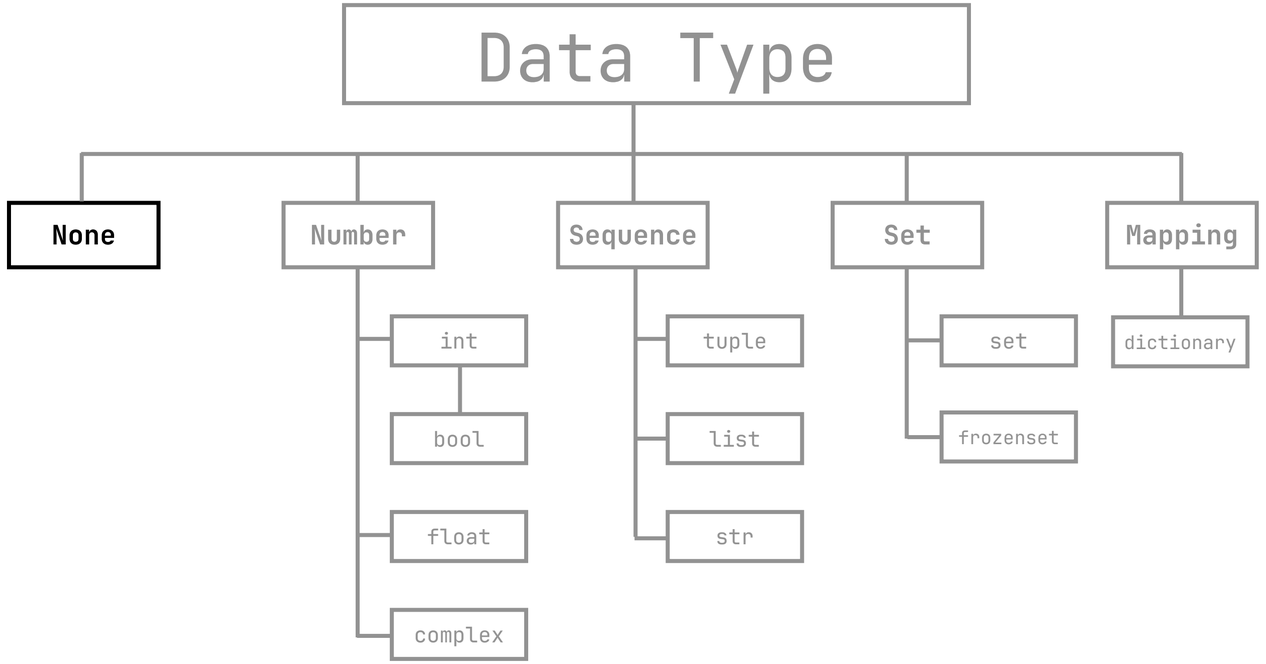
1.type

None

값이 없음을 표현하는 객체

내장 상수

singleton



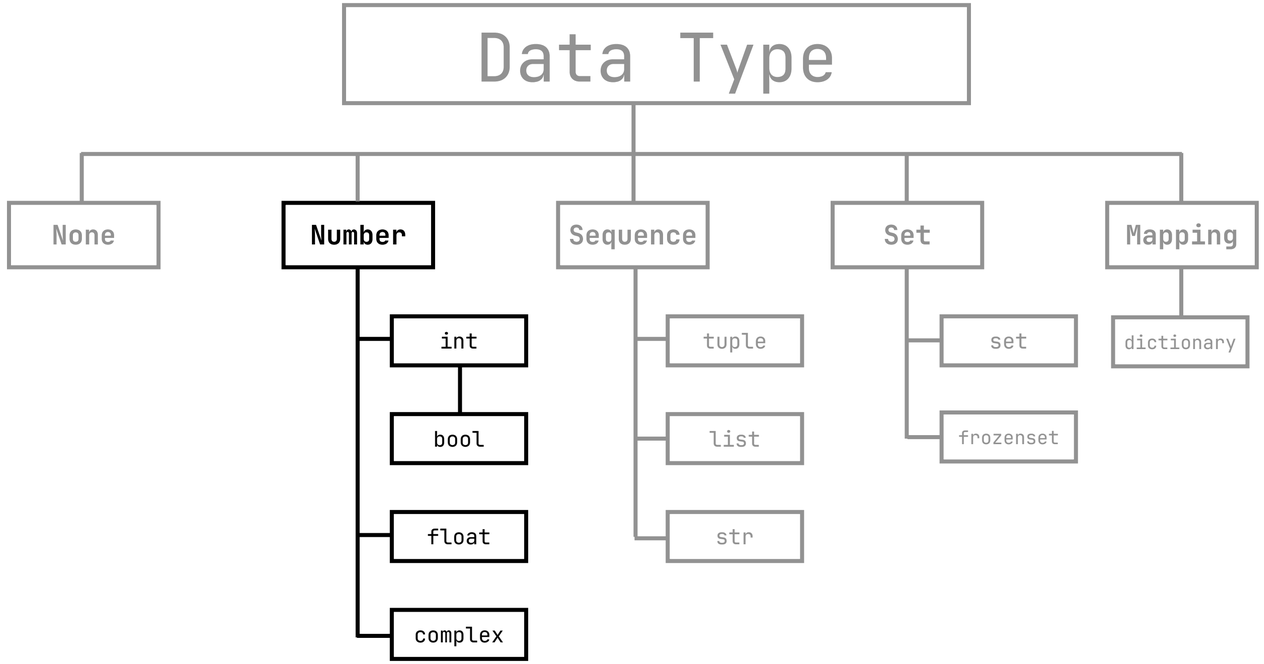
* undefined 와는 다르다!

1.type

Number

숫자 값

int	정수
bool	논리
float	실수
complex	복소수

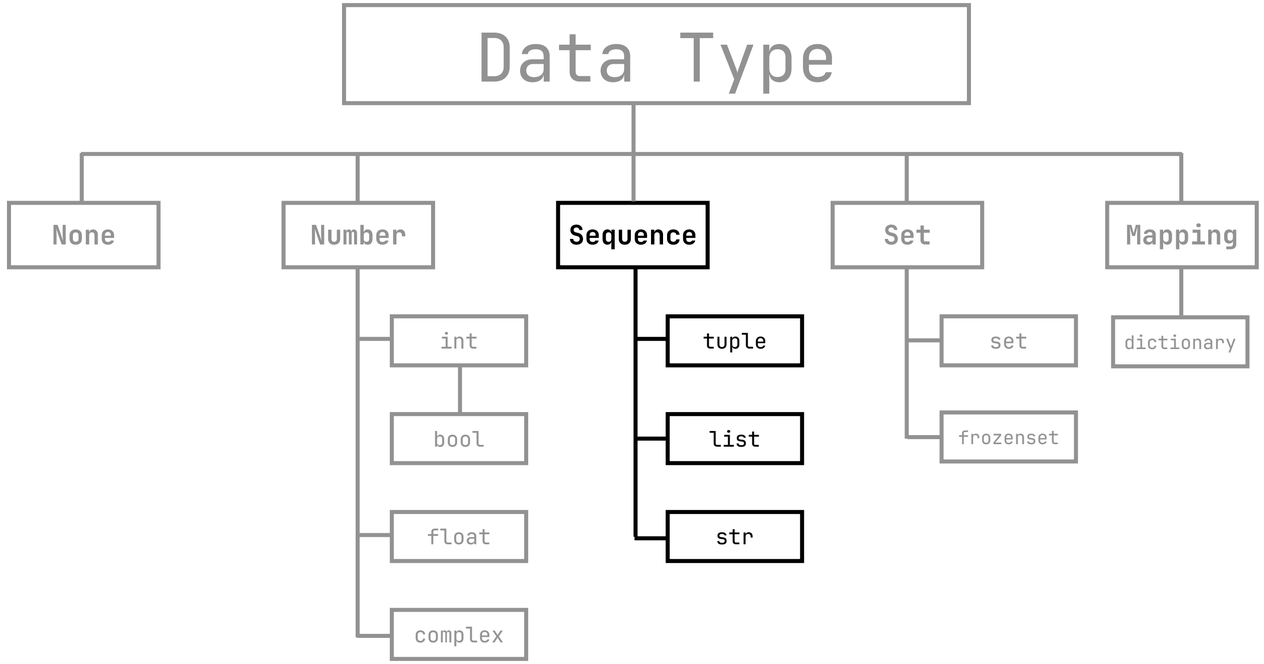


1.type

Sequence

순서(index)가 있는 값들을 가진 객체

tuple	( )
list	[ ]
str	" " ' ' """ """ ''' '''



* index 는 0부터 시작

1.type

Sequence

\n	linefeed	\\	backslash
\r	carriage return	\'	single quotation
\t	tab	\"	double quotation
\b	backspace		

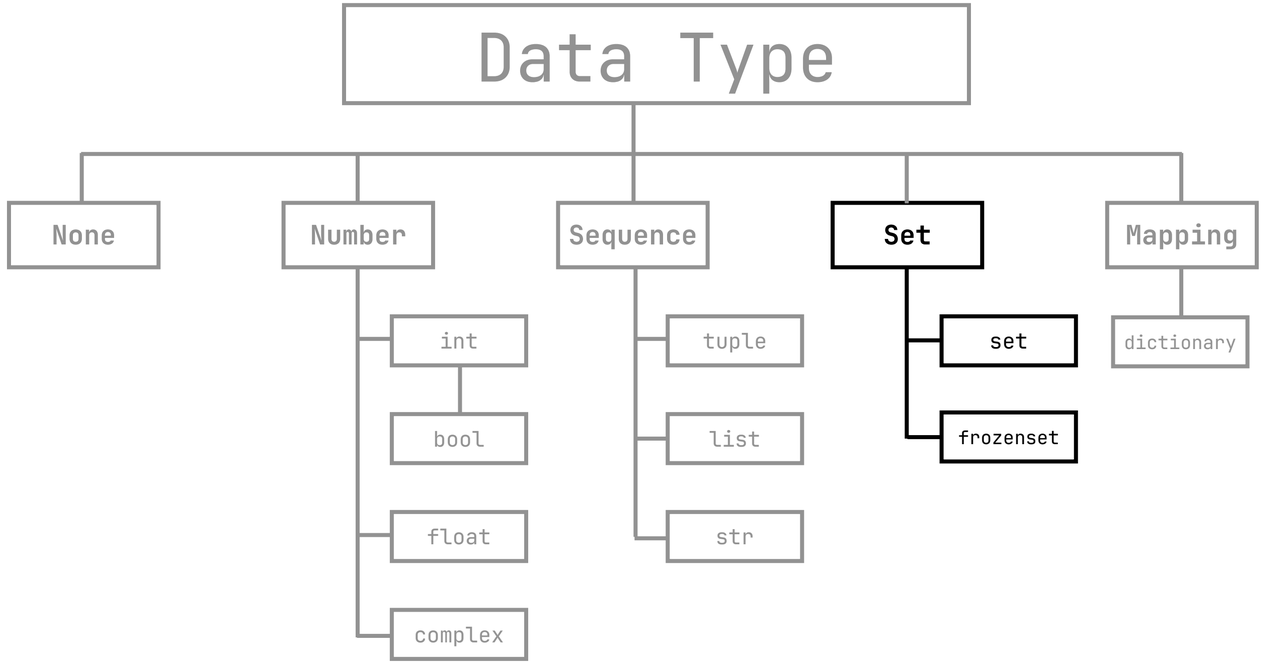
1.type

Set

집합을 표현하는 객체

set { }

frozenset 불변 집합



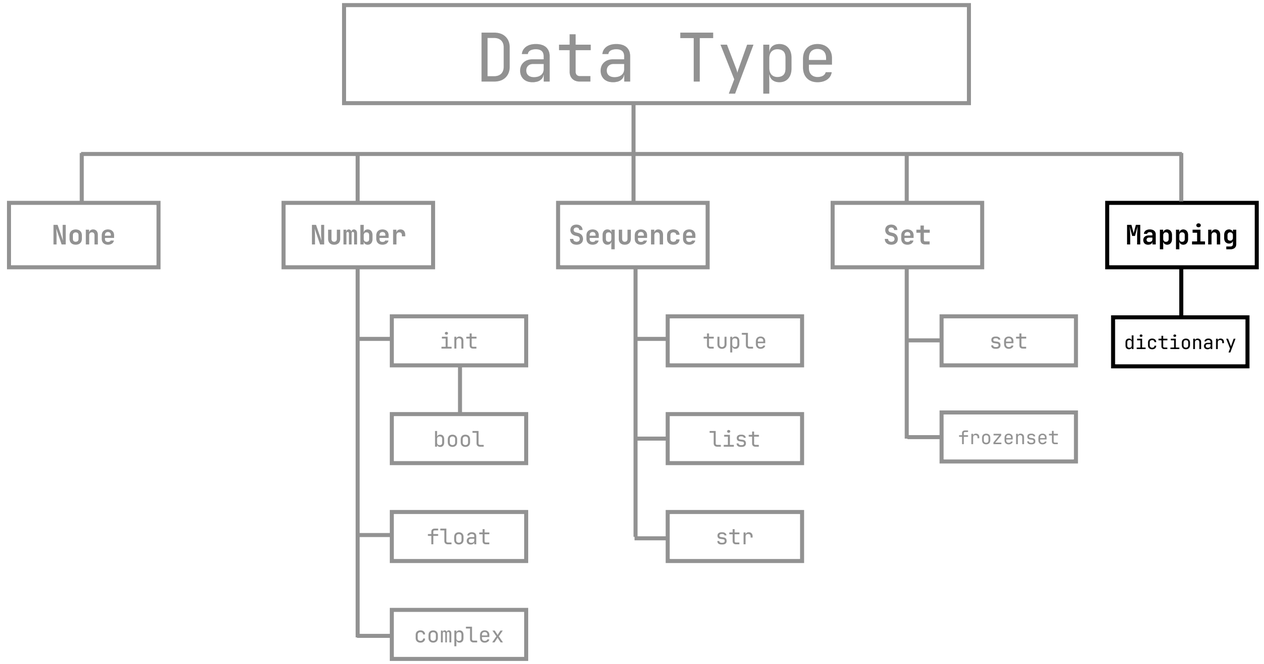
* mutable vs immutable

1.type

Mapping

key로 value를 관리하는 객체

dictionary {key: value}



2.operator

산술연산

+	더하기	**	거듭제곱
-	빼기	//	몫
*	곱하기	%	나머지
/	나누기		

2.operator

비교, 논리, 멤버, 증감연산

==	and &	in	+=
!=	or ¦	not in	-=
>	not		*=
>=			/=
is			
is not			

2.operator

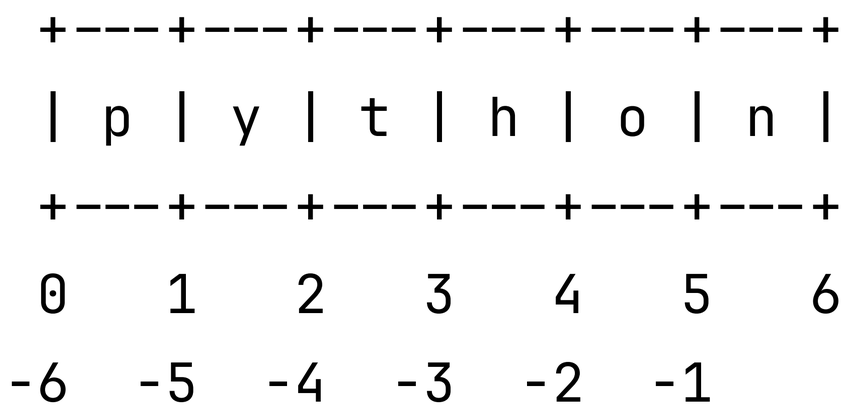
범위 객체

range(stop)	0 ~ stop-1 (0 <= i < stop)
range(start, stop)	start ~ stop-1 (start <= i < stop)
range(start, stop, step)	start, start+step, ... , stop-1

* start, stop, step 값만 가지기 때문에 메모리 절약

2.operator

slice



[n]

[start: stop]

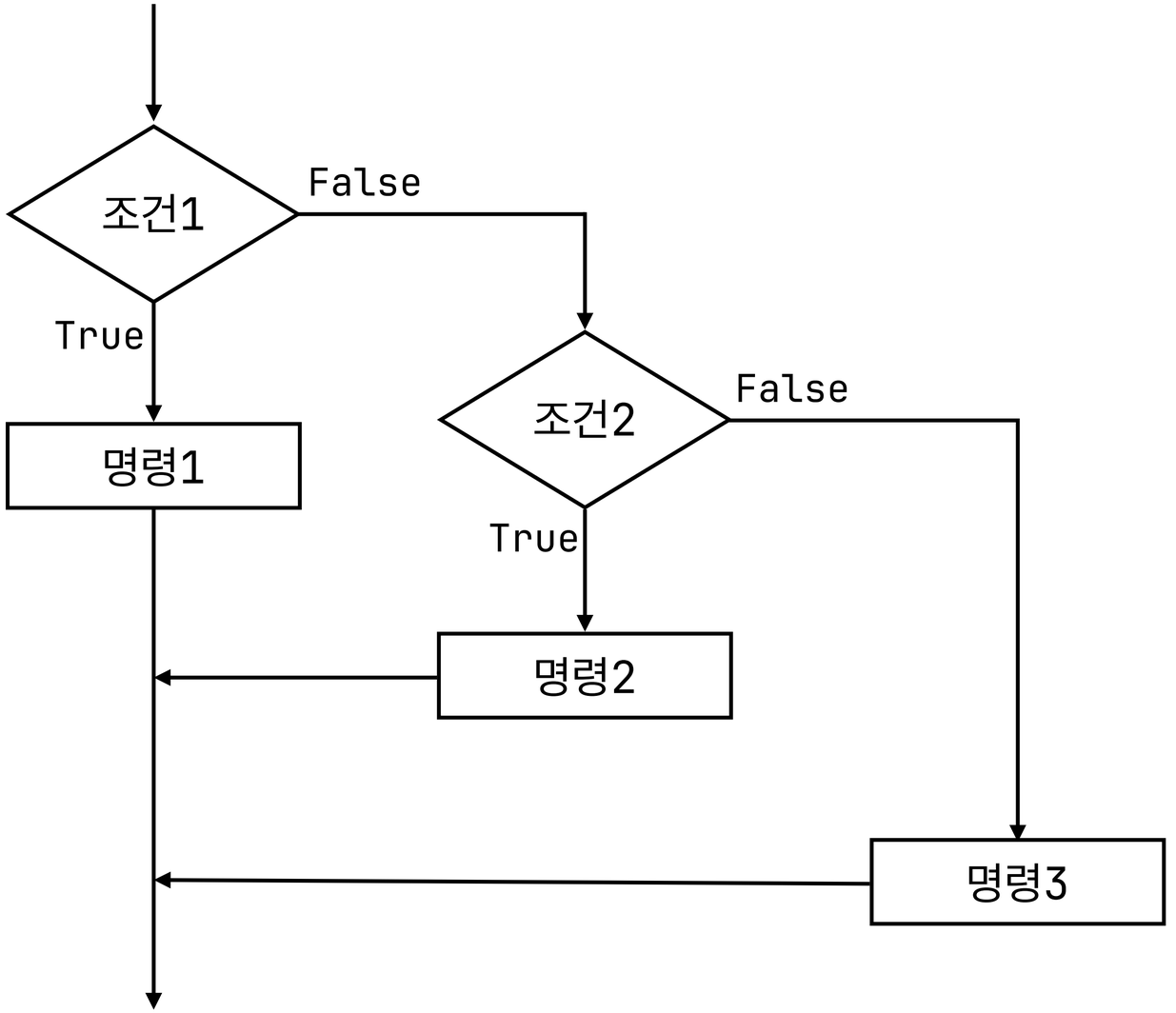
[start: stop: step]

* index 범위로 slice

3.control

조건문

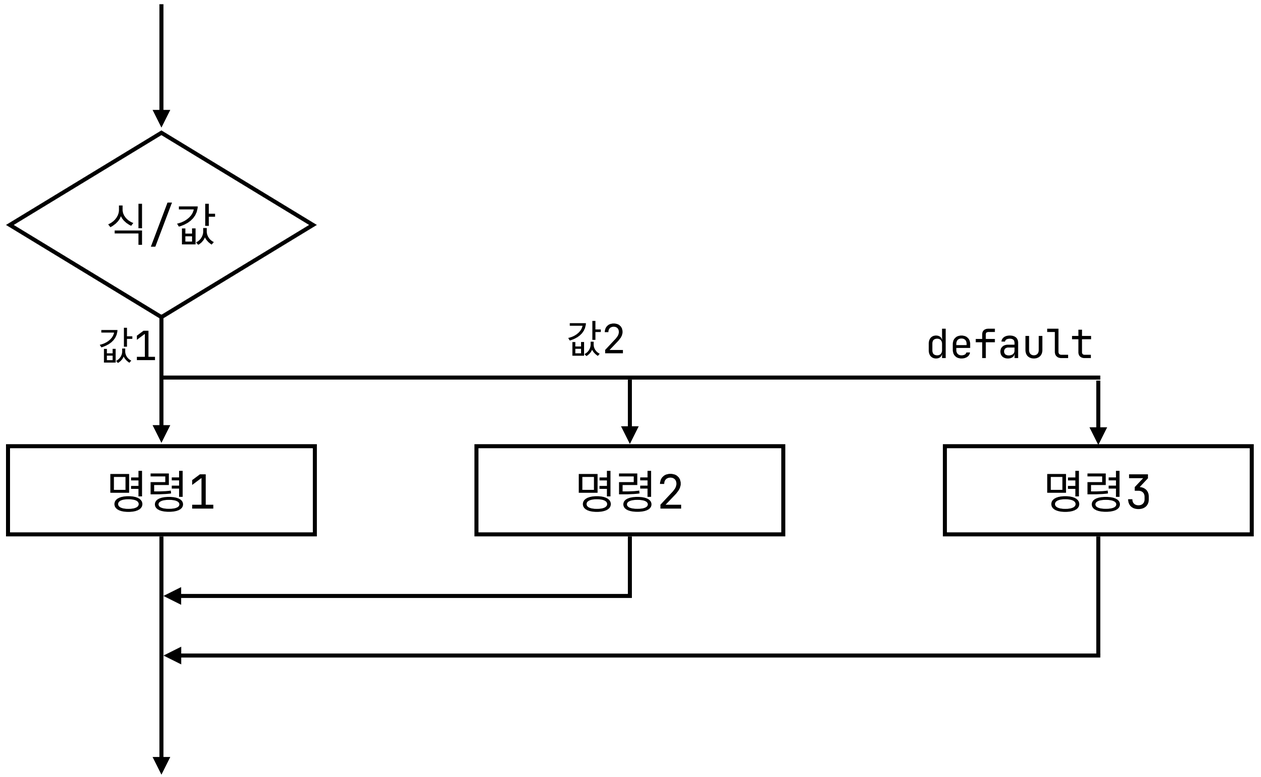
```
if 조건1 :
    명령1
elif 조건2 :
    명령2
else :
    명령3
```



3.control

조건문

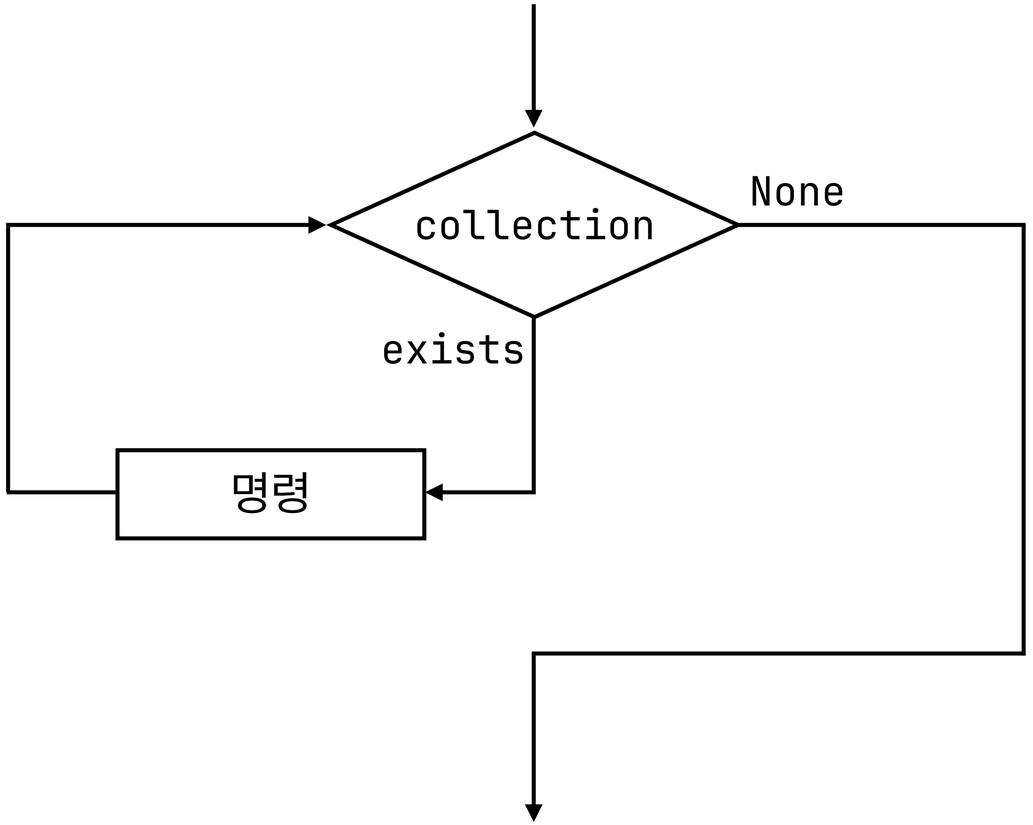
```
match 식/값 :

    case 값1 :
        명령1

    case 값2 :
        명령2

    case _ :
        명령3
```



3.control

반복문

```
for 값 in collection :

    명령
```

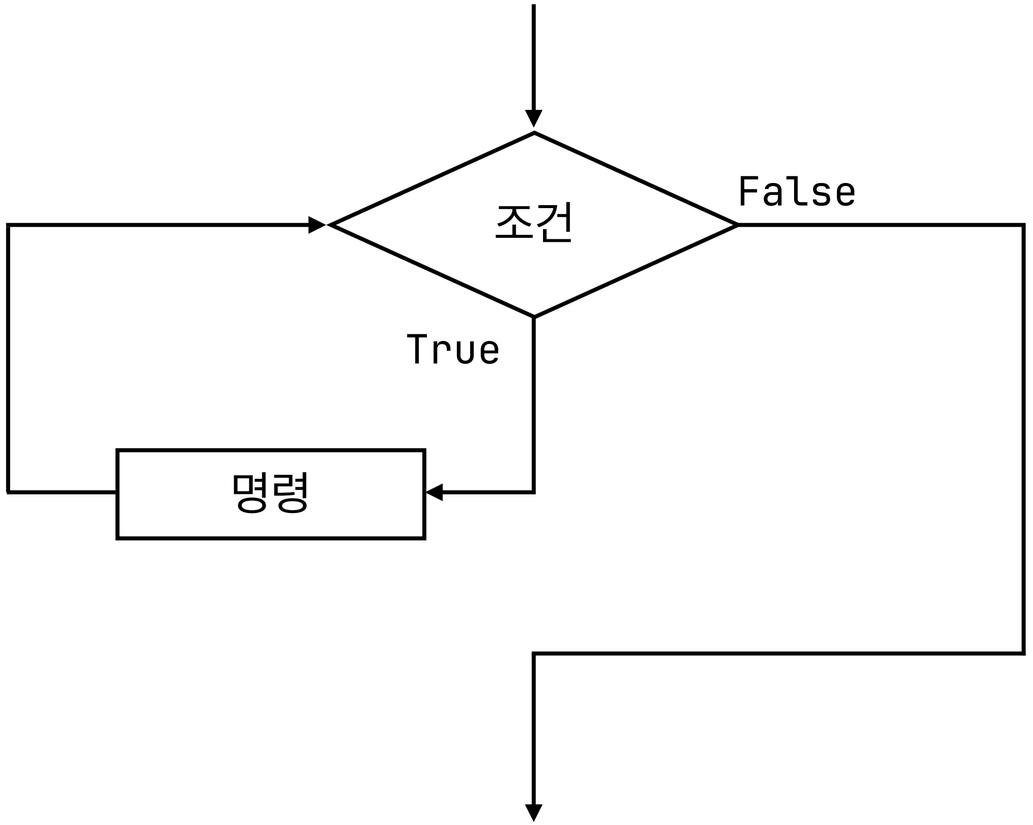


* enumerate : (index, value)

3.control

반복문

```
while 조건 :
    명령
```



* do ~ while 없음

4.function

def

```
def 함수(parameter):
    명령
    return
```

변수

*args

**kwargs

* parameter(매개변수) / argument(인수)

4.function

lambda 파라미터: 명령

익명 함수 표현식

함수를 수식으로 표현

4.function

scope

global	전역변수
nonlocal	상위변수 (전역변수 제외)
local	지역변수

* closure : lexical scope (lexical environment를 기억)

5.module

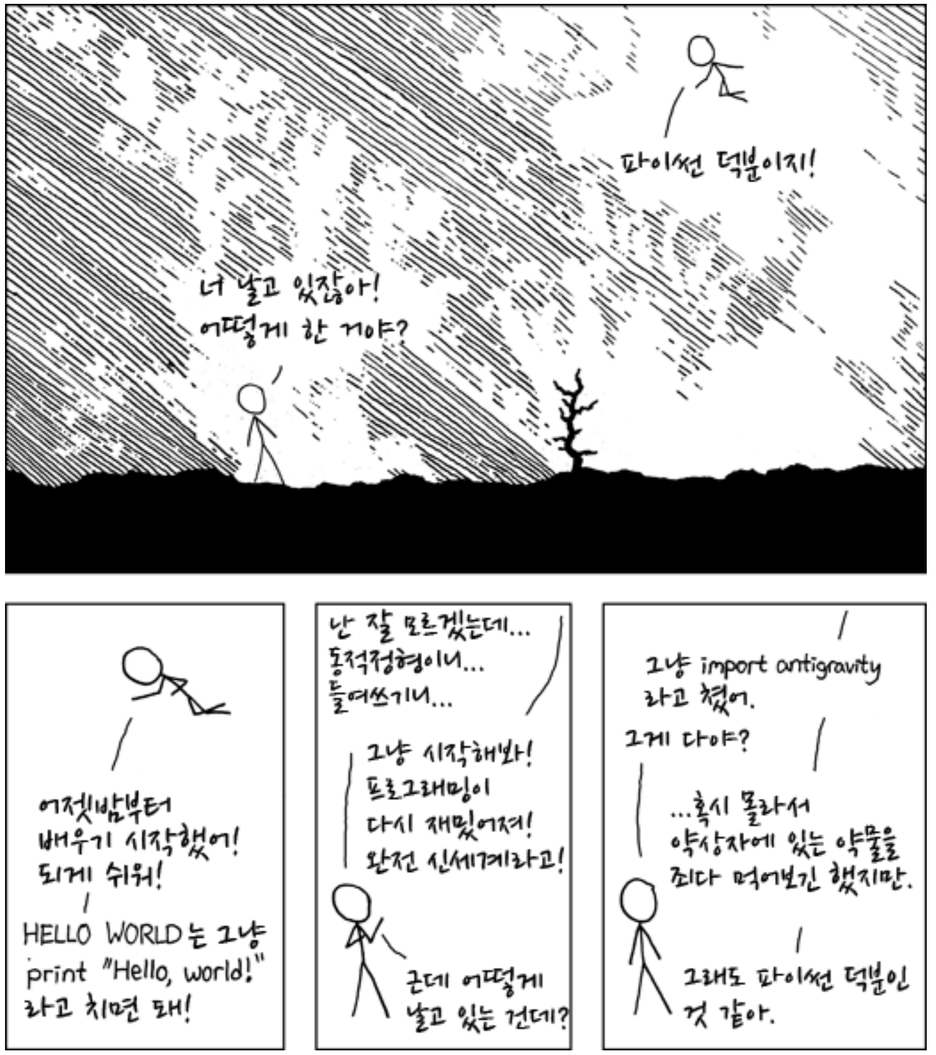
pip

pip

python package (library, module) 관리 시스템

PyPI (Python Package Index)

공식 package 저장소



6.io

input output

r	읽기	t	text
w	쓰기 (기존 내용 덮어쓰기)	b	binary
a	쓰기 (기존 내용 이후에 쓰기)	+	read | write 추가
x	새로운 파일 만들어서 쓰기 (이미 파일이 있으면 에러)		

7.class

oop

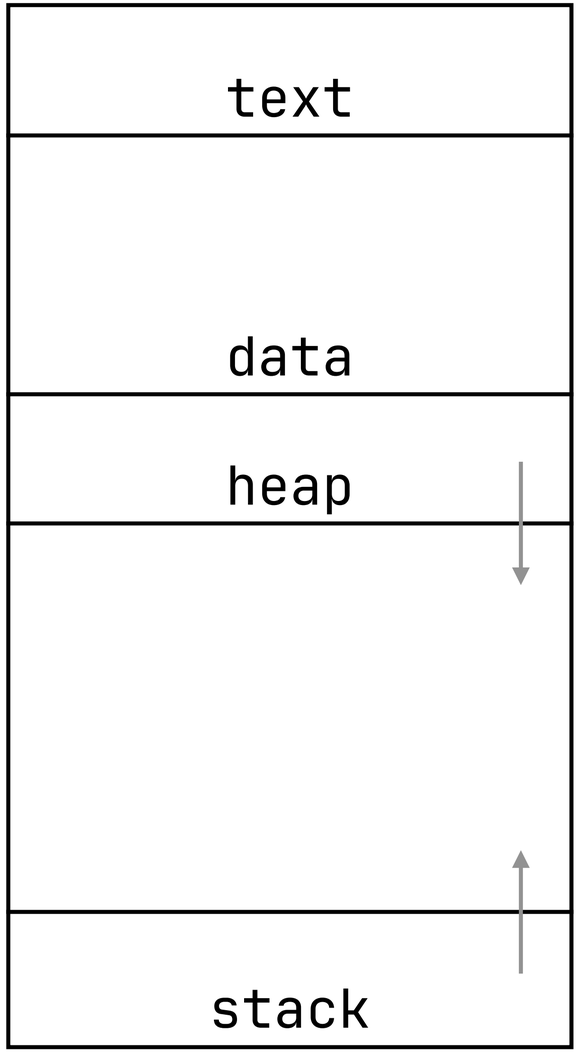
Object Oriented Programming

inheritance	상속
abstraction	추상화
polymorphism	다형성
encapsulation	캡슐화

7.class

text (code)	명령어
data	static, global variable
heap	instance
stack	local variable

memory



7.class

변수

class variable	class의 변수 (static)
instance variable	객체가 사용할 수 있는 변수 (dynamic)
__변수	class 내부에서만 사용가능한 변수

7.class

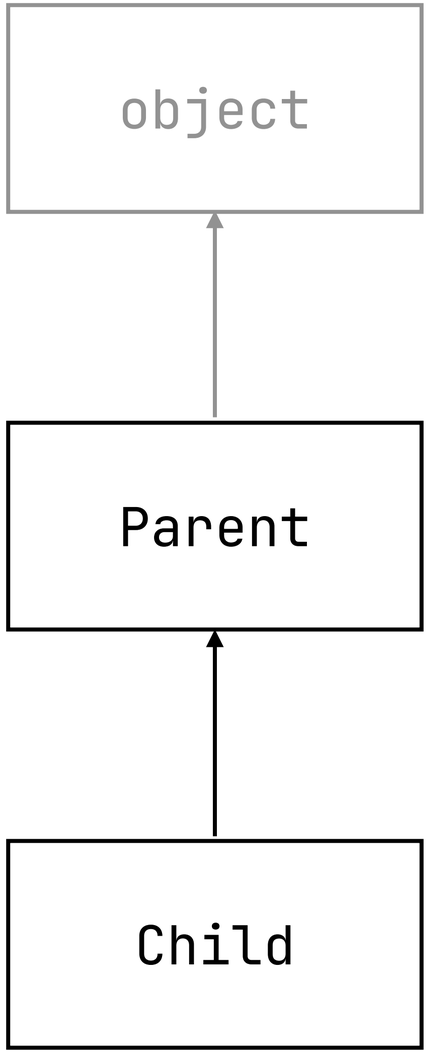
decorator

@property	class 외부에서 변수의 값을 호출하기 위한 함수
@변수.setter	class 외부에서 변수에 값을 대입하기 위한 함수
@classmethod	method를 호출한 class의 변수를 사용하는 method
@staticmethod	super class의 변수를 사용하는 method

7.class

inheritance

```
class Parent:
      ...


class Child(Parent):
      ...
```



* python의 모든 클래스는 object를 상속 !

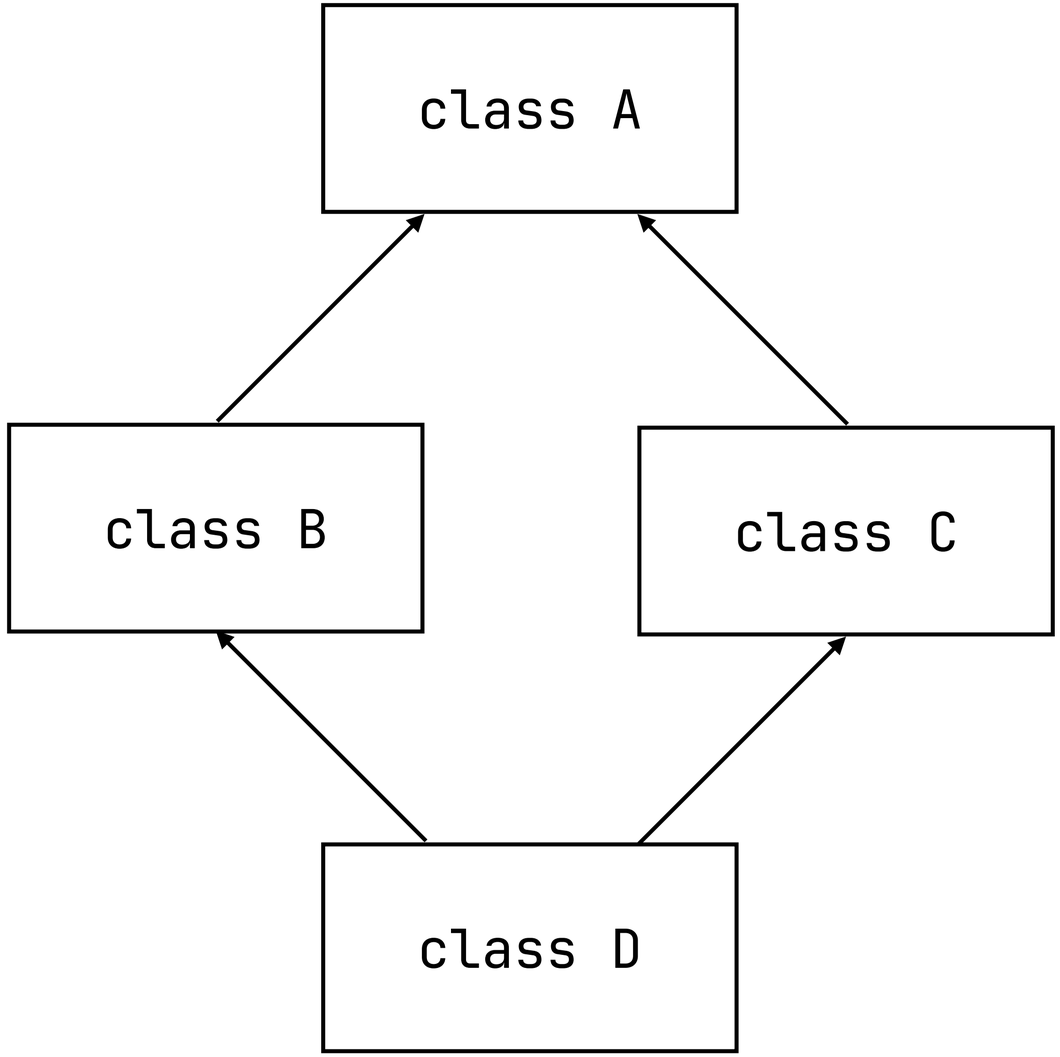
7.class

inheritance

다이아몬드 상속

override 시 문제가 생길 수 있음

* mro (method resolution order)



7.class

abstract

```
from abc import *
```

abstract base class

```
class Abstract01(ABC):

    @abstractmethod
    def abstract_method():
```

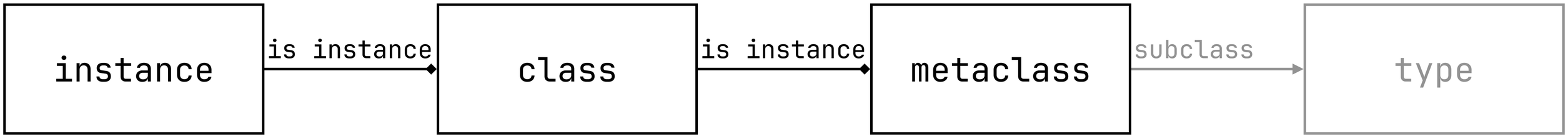
```
class Abstract02(metaclass=ABCMeta):

    @abstractmethod
    def abstract_method():
```

7.class

metaclass

type　　class 의 기본 type

metaclass　　class 를 만들어주는 class



* singleton : memory 에 instance 가 단 하나만 생성

8.decorator

개념

기능을 추가하기 위한 syntax sugar

class나 함수위에 @이름 으로 사용

closure 개념의 확장

class decoration을 사용할 땐, __call__ override

9.exception

예외처리

예외가 발생했을 때 프로그램의 강제 종료 방지

try: 예외가 발생할 가능성이 있는 코드

except 예외: 예외 발생 시 처리할 코드

else: 예외 미 발생 시 처리할 코드

finally: 예외 발생 여부와 상관 없이 무조건 처리할 코드

10.iterator

iterator

반복 객체 (값을 순서대로 하나씩 꺼낼 수 있는 객체)

iterable	반복가능한(sequence, iterator, ...)
__iter__()	iterator 반환
__next__()	iterator 내부의 값 하나 반환

* lazy evaluation : 실행할 때 값이 연산된다

11.generator

generator

yield

외부로 값 전달

__next__()가 호출될 시 yield 의 값을 외부로 전달

yield from

다른 generator 호출

12.concurrency

개념

concurrency

프로그램(알고리즘)이 동시에 실행

parallelism

물리적으로 동시에 실행

* concurrency ≠ parallelism

12.concurrency

구현

thread

process의 작업 단위

process

memory에 적재된 program(job, task)

coroutine

협동 처리

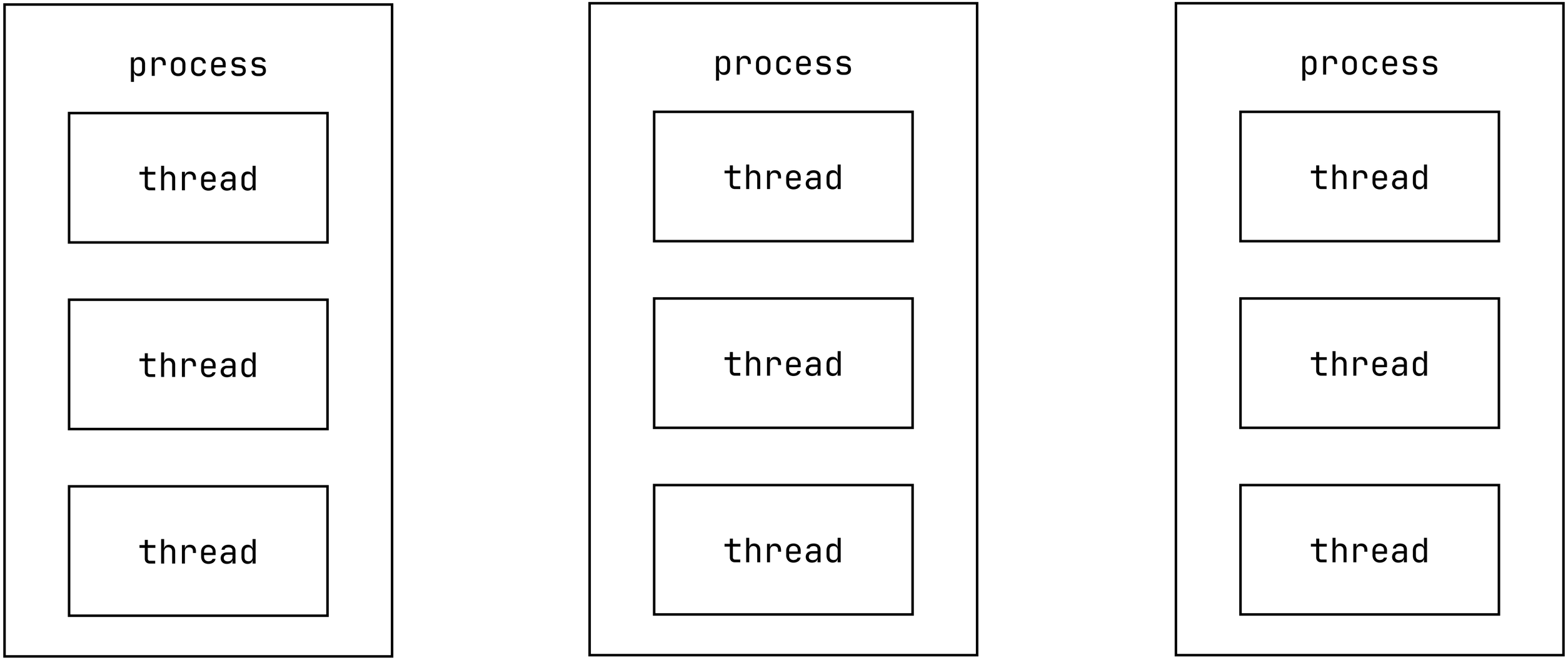
- generator

- asyncio (async await)

- ~~@coroutine~~ : deprecated

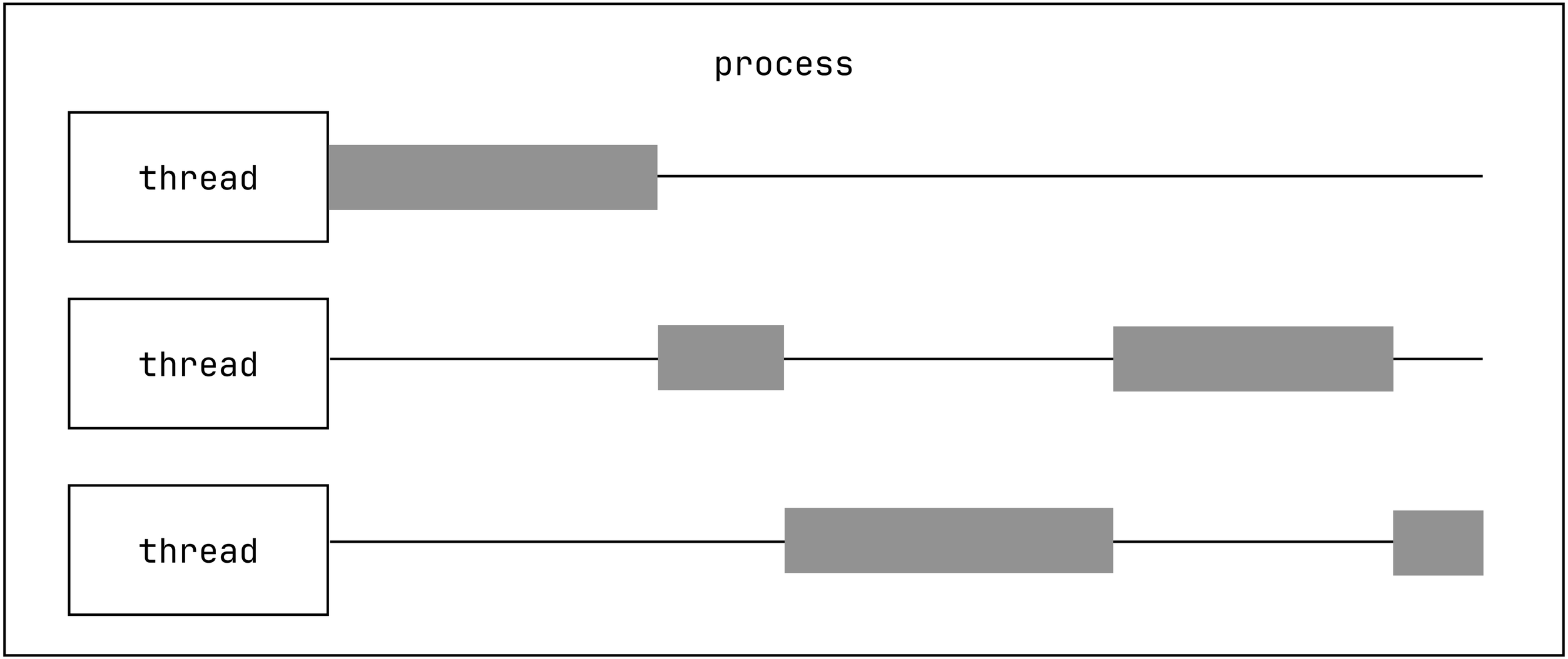
12.concurrency

thread, process



12.concurrency

multithread



12.concurrency

multiprocess

12.concurrency

async