11

Class & Modules

Python built-in objects

Object type	Example literals/creation
Numbers	1234, 3.1415, 0b111, 1_234, 3+4j, Decimal, Fraction
Strings	'code', "app's", b'a\x01c', 'h\u00c4ck', 'hÄck'
Lists	[1, [2, 'three'], 4.5], list(range(10))
Tuples	(1, 'app', 4, 'U'), tuple('hack'), namedtuple
Dictionaries	{'job': 'dev', 'years': 40}, dict(hours= 10)
Sets	set('abc'), {'a', 'b', 'c'}
Files	open('docs.txt'), open(r'C:\data.bin', 'wb')
Other core objects	Booleans, types, None
Program-unit objects	Functions, modules, classes
Implementation objects	Compiled code, stack tracebacks

Mark Lutz. (2025). Learning Python, 6th Edition. n.p.: O'Reilly Media, Inc.

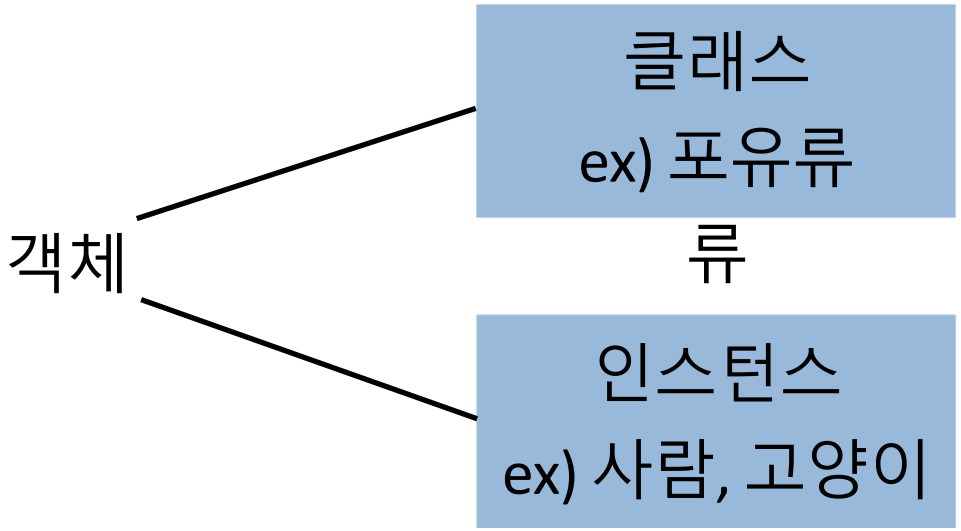
Object Oriented Programming

객체 지향 프로그래밍(OOP, Object Oriented Programming)

- 객체: 특정 기능을 수행하는 실체
- 클래스: 객체를 구현하기 위한 설계도
- 인스턴스: 클래스를 기반으로 생성된 실체

객체 vs. 인스턴스

- 사람은 **객체**
- 사람은 포유류의 **인스턴스**



Class

- 10은 int 클래스, “hello”는 str 클래스의 인스턴스
- 기본 자료형 → 클래스를 통한 구현

```
n = 10
s = "hello"

print(type(n))
print(type(s))
```

```
<class 'int'>
<class 'str'>
```

```
a = [1, 2, 3]
d = {"key": "value"}

print(type(a))
print(type(d))
```

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

Class

- 전역변수를 활용한 덧셈 연산 함수
- 만약 2개의 덧셈이 동시에 일어난다면?

```
result = 0

def adder(num):
    global result
    result += num
    return result

print(adder(3))
print(adder(4))
```

```
3
7
```

Class

- 하나의 프로그램에서 동일한 기능을 하는 함수 2개가 필요한 경우

```
result1 = 0
result2 = 0

def adder1(num):
    global result1
    result1 += num
    return result1
def adder2(num):
    global result2
    result2 += num
    return result2
```

```
print(adder1(3))
print(adder1(4))
print(adder2(3))
print(adder2(7))
```

```
3
7
3
10
```

Class

- 클래스 정의를 통해 전역변수와 함수 개수를 줄일 수 있음

```
class Calculator:
    def __init__(self):
        self.result = 0

    def adder(self, num):
        self.result += num
        return self.result

cal1 = Calculator()
cal2 = Calculator()
```

```
print(cal1.adder(3))
print(cal1.adder(4))
print(cal2.adder(3))
print(cal2.adder(7))
```

```
3
7
3
10
```

Class

- 유사한 데이터를 다룰 때, 구조를 통일하고 기능을 공통화할 수 있음
- 예: 여러 개의 학생 정보 관리 시 → Student 클래스

```
>>> class Simple:
...     pass
...
>>> a = Simple()
>>> type(a)
<class '__main__.Simple'>
```

→ Simple()의 결과값을 돌려받은 a: Simple 클래스의 인스턴스

Class

- name = 'Alice'는 클래스 변수로, 모든 인스턴스가 공유
- a = Simple()은 클래스로부터 인스턴스를 생성

```
>>> class Simple:
...     name = 'Alice'
...
>>> a = Simple()
>>> a.name
'Alice'
>>> b = Simple()
>>> b.name
'Alice'
```

Module

- 모듈은 함수/변수 또는 클래스들을 모아 놓은 파일

```
import math
print(math.sqrt(9))  # 3.0
```

사용하는 이유

- 코드 재사용
- 유지보수 용이
- 파일 분리 → 깔끔한 프로젝트 구조

Module

- 모듈 명을 생략하고 사용하고 싶을 때

```
import math as m
print(m.pi)
```

```
3.141592653589793
```

- 여러 개의 함수를 사용할 때는 콤마(,)로 구분해서 사용

```
from math import sqrt, pi
print(sqrt(16), pi)
```

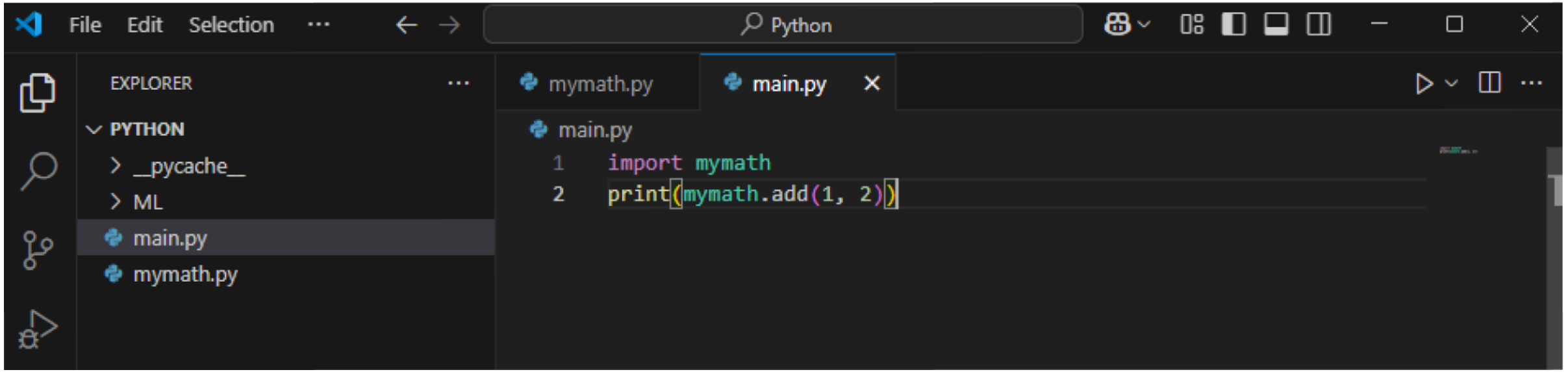
```
4.0 3.141592653589793
```

Module

- 같은 폴더 내에 위치한 파일의 경우, 다음과 같은 방식을 이용하여 import 가능
- @main.py Ctrl + F5 → Python Debugger => 화면 하단 terminal에서 결과 확인

```
# mymath.py
def add(a, b):
    return a + b
```

```
# main.py
import mymath
print(mymath.add(1, 2))
```



Frequently used modules

- import random
- 다양한 난수를 생성하는 도구

```
import random
# 0 ~ 1 범위의 실수 난수 반환
num = random.random()
print(num)
# 100부터 1000사이의 정수 난수
num = random.randint(100,1000)
print(num)
```

```
0.25235268467991645
474
```

Frequently used modules

- seed()는 난수를 "고정"하기 위한 도구
- random.seed()를 사용하지 않으면 현재 시간을 기반으로 난수 생성

```
import random

random.seed(1) # seed값을 1로 정의
num = random.randint(1, 100) # 1부터 100사이의 숫자를 반환
print(num) # 어떤 시간, 컴퓨터에서 실행해도 결과는 18로 동일
```

```
18
```

Frequently used modules

- import math
- 계산과 관련된 메서드들 존재

```
import math

print(math.ceil(10.0928))                    # x의 올림 값을 반환
print(math.floor(10.0928))                   # x의 내림 값을 반환
print(math.sqrt(2))                          # x의 제곱근을 반환
print(math.pow(2, 3))                        # x의 y 제곱을 반환
print(math.sin(math.pi / 6))
```

```
11
10
1.4142135623730951
8.0
0.49999999999999994
```

Frequently used modules

- import time
- 코드 실행 시간 측정에 용이

```
import time
print(time.time())
```

```
1753032044.8141267
```

```
import time
start_time = time.time()
num = 10000 ** 10000
end_time = time.time()
print(end_time - start_time)
```

```
0.017449855804443336
```

Frequently used modules

```python
import random
score = 0

for _ in range(5):
    a = random.randint(2, 9)
    b = random.randint(1, 9)
    result = int(input(f'{a} * {b} = ? '))
    if result == a * b:
        score += 20
        print("맞았습니다.")
    else:
        print("틀렸습니다.")

print("최종 점수는", score, "점입니다.")
```

```
9 * 8 = ?  72
맞았습니다.
3 * 6 = ?  18
맞았습니다.
2 * 8 = ?  0
틀렸습니다.
2 * 6 = ?  12
맞았습니다.
3 * 7 = ?  0
틀렸습니다.
최종 점수는 60 점입니다.
```

11주차 과제

- 11_Class_Methods.ipynb
- 파이썬의 다양한 모듈 확인 가능
  https://docs.python.org/3/py-modindex.html