10

# Function

# 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.

# Function

- 특정 작업을 수행하는 코드 묶음
- 중복 제거, 재사용성, 구조화에 유리
- def 키워드로 정의

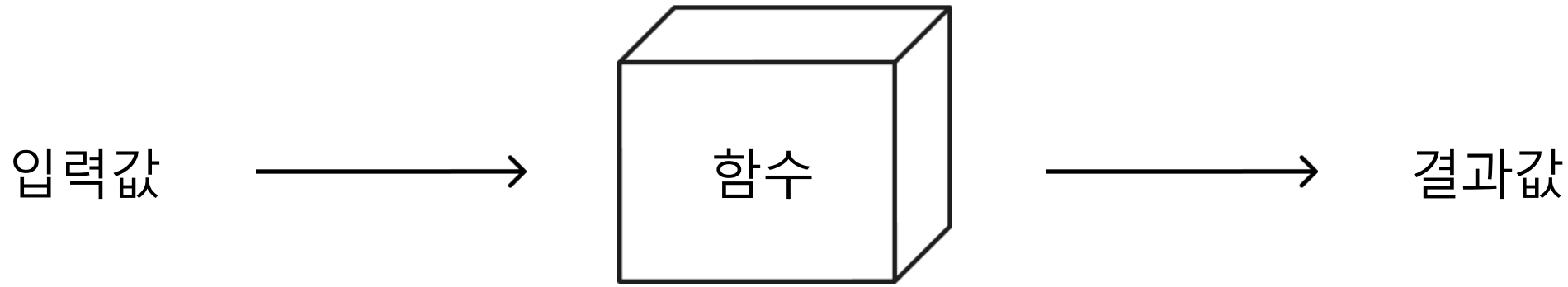
```
def 함수이름():
    실행할 코드
```

- 예: def say_hello(): print("Hello!")
- 호출: say_hello()

# Function

함수의 장점

- 코드 모듈화 가능
- 수정 및 관리 용이
- 한 번 정의해두고 여러 번 호출 가능



# Function

- 함수는 정의만으로 실행되지 않음, 반드시 호출 필요

```
def greet():
    print("안녕하세요")
```

```
greet()
```

안녕하세요

# Function

- name은 매개변수(parameter)
- greet("Alice") → "Alice"는 인자(argument)

```
def greet(name):
    print("안녕,", name)
```

```
greet("Alice")
```

```
안녕, Alice
```

# Function

- 순서와 상관없이 인자 지정 가능

```
def info(name, age):
    print(name, age)

info(age=25, name="Tom")
```

```
Tom 25
```

# Function

- 단순 출력만 수행
- 값은 반환되지 않음

```
def add_print(a, b):
    print("합계:", a + b)
```

```
add_print(3, 4)
```

```
합계: 7
```

# return

- return은 결과 값을 반환하고 함수 종료

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

```
result = add(3, 4)
print(result)
```

```
7
```

# return

- 반환값 없는 경우 기본(default)은 None

```
def test():
    pass

print(test())
```

```
None
```

# return

- 여러 return으로 조건별 분기 가능

```
def calculator(a, b, operand):
    if operand == "+":
        return a + b
    elif operand == "-":
        return a - b
    elif operand == "*":
        return a * b
    elif operand == "/":
        return a / b
    else:
        print("Not supported operand")
        return
```

# return

- 여러 return으로 조건별 분기 가능

```
print(calculator(3, 4, '+'))
```

```
7
```

```
print(calculator(5, 2, "*"))
```

```
10
```

```
print(calculator(5, 2, "//"))
```

```
Not supported operand
None
```

# return

- 함수는 return 값1, 값2처럼 여러 값을 동시에 반환 가능
- 반환된 값은 튜플로 간주되며, 변수 여러 개에 한꺼번에 할당 가능

```
def sum_and_mul(a, b):
    return a+b, a*b
```

```
result = sum_and_mul(3, 4)
print(result, type(result))
```

```
(7, 12) <class 'tuple'>
```

```
sum, mul = sum_and_mul(3, 4)
print(sum, mul, type(sum))
```

```
7 12 <class 'int'>
```

# Parameter with a default

- 함수 매개변수에 기본값(default value) 지정 가능
- 호출 시 해당 인자를 생략하면 기본값이 자동으로 사용

```
def greet(age, name="Guest", man=True):
    print("Hello,", name)
    print(age)
    if man:
        print("Male")
    else:
        print("Female")
```

```
greet(21, "Tom")
```

```
Hello, Tom
21
Male
```

# Parameter with a default

- 이때 초기값을 설정한 인수는 그렇지 않은 인수 뒤에 위치해야함

```
def greet(name="Guest", age, man=True):
    print("Hello,", name)
    print(age)
    if man:
        print("Male")
    else:
        print("Female")
```

```
  Cell In[11], line 1
    def greet(name="Guest", age, man=True):
                            ^
SyntaxError: parameter without a default follows parameter with a default
```

# Global variable

- a = 1 → 전역 변수 정의
- 함수 매개변수로 전달되는 변수(a)는 복사되어 지역 범위(a = a + 1)에서 사용
- print(a)는 1 출력

```
a = 1
def count(a):
    a = a + 1

count(a)
print(a)
```

```
1
```

# Global variable

- 함수 내부 변수(a)는 지역 변수(local variable)이며, 함수가 끝나면 소멸
- 전역에서 쓰려면 함수 밖에서 a를 선언하거나 값을 반환(return) 받아야 함

```
def count(a):
    a = a + 1

count(3)
print(a)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[2], line 5
      2     a = a + 1
      4 count(3)
----> 5 print(a)

NameError: name 'a' is not defined
```

# Global variable

- return 사용

```
a = 1
def count(a):
    a = a + 1
    return a

a = count(a)
print(a)
```

```
2
```

- global 사용 (종속적)

```
a = 1
def count():
    global a
    a = a + 1

count()
print(a)
```

```
2
```

# 10주차 과제

- 10_Funtion.ipynb