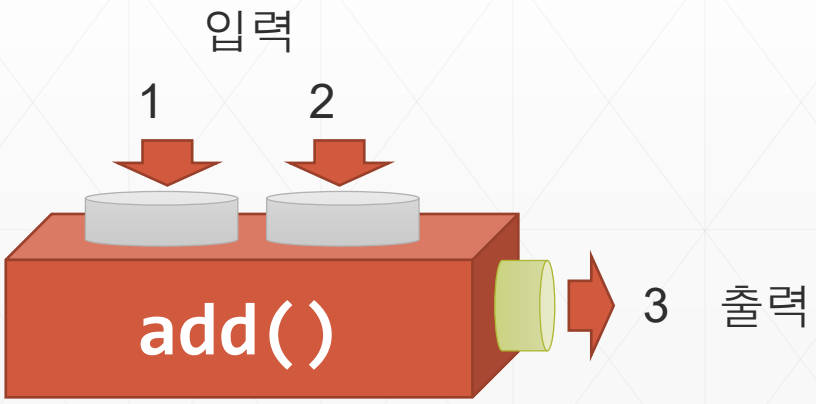
컴퓨터정보과

파이선 프로그래밍

07주차 함수와 자료형

함수

- function p.153
  - 입력을 받아서 특정 작업을 수행하고, 해당 결과를 반환하는 블랙 박스와 같다.
  - 함수는 입력을 받아서, 출력(결과)를 생성한다.
  - 사용 하는 이유
    - 반복적으로 사용하는 가치 있는 부분을 묶어서 사용하는 것
    - 전체 흐름을 단계적으로 만든다.



함수

- 정의 (definition) p.154
  - 기본 형태

```
def 함수이름(매개변수, ...):
    실행문장 ...
    return 결과값
```

```
def add(x, b):
    result = x + b
    return result
```

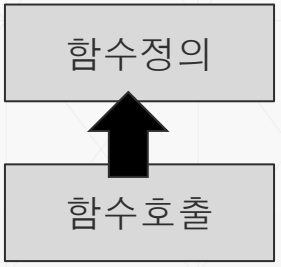
- 호출 (call) p.155
  - 기본 형태

```
변수 = 함수이름(입력인수1, ...)
```

*변수는 함수의 결과값을 가리키는 용도

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

- 함수 안의 문장은 언제 실행되는가?
  - 함수가 호출되면 실행한다.
- 호출은 어떻게 하는가?
  - 함수의 이름을 적어주면 된다.
  - 함수로 실행의 흐름이 이동한다.
- 함수의 실행이 끝나면 어떻게 되는가?
  - 호출한 쪽으로 실행의 흐름이 다시 돌아간다. (Interrupt)



함수

- 함수의 다양한 형태 p.156

✓ 매개 변수, 결과값이 있는 함수

```
1  def add1(a, b):
2      result = a + b
3      return result
4
5  print(add1(1, 2))
```

✓ 결과값이 없는 함수

```
14 def add3(a, b):
15     result = a + b
16     print(result)
17
18 print(add3(1, 2)) #None
```

✓ 매개 변수가 없는 함수

```
8  def add2():
9      result = 1 + 2
10     return result
11
12 print(add2())
```

✓ 매개 변수/결과값이 없는 함수

```
20 def add4():
21     result = 1 + 2
22     print(result)
23
24 print(add4())  #None
```

함수

- 매개변수와 인수 p.155
  - 정의:매개변수(parameter)
  - 호출:인수 (argument)
  - 종류
    - 위치형 매개변수 positional argument (기본방식)
      - 정의한 매개변수 순서대로 지정하고 호출하기
    - 키워드 매개변수 keyword argument (p.159)
      - 매개변수 지정하여 호출하기
    - 기본값 매개변수 default value argument (p.165)
      - 매개변수에 기본값을 설정하여, 매개변수에 값을 지정하지 않고 호출하기
        - 맨 뒤 매개변수부터 기본값을 지정할 수 있음.

```
def test(a, b, c, d=10):
    print(a, b, c, d)


test(1, 2, 3, 4)            # 위치형 매개변수
test(b=2, a=1, c=3, d=4)    # 키워드 매개변수
test(1, 2, 3)               # 기본값 매개변수
```

```
1 2 3 4
1 2 3 4
1 2 3 10
>>>
```

함수

- 결과값 반환 : return p.163
  1. 결과값을 호출한 쪽으로 전달한다 (동시에 함수 종료)
  2. 함수 종료 (함수 탈출) p.164
  - return이 전달하는 결과값은 **Only One**.
    - 정체는?

```
1   def add_and_mul1(a,b):
2       return a+b, a*b
3
4   def add_and_mul2(a,b):
5       return a+b
6       return a*b
7
8   a = add_and_mul1(3,4)
9   b = add_and_mul2(3,4)
10
11  print(a)
12  print(b)
```

```
(7, 12)
7
```

함수

- 변수의 범위 : 전역(global)과 지역(local) p.167
  - 함수 밖에서 지정한 변수 : 전역변수
  - 함수 안에서 지정한 변수 : 지역변수

```
1  a = 1  # 전역변수
2
3  def vartest(a):     # 지역변수, 호출시 넘어온 값을 가리킴
4      a = a + 1       # 지역변수 a가 가리키는 값에 1을 저장하고, 그 결과를 다시 a로 가리킴
5
6  vartest(a)          # vertest()에게 a가 가리키는 1의 위치를 넘김
7  print(a)            # 전역변수 a를  출력
```

  - Line. 1을 주석처리하면?
  - 전역변수를 변경하고 싶으면?

(1) return

```
1  a = 1
2
3  def vartest(a):
4      a = a + 1
5      return a
6
7  a = vartest(a)
8  print(a)
```

(2) global 키워드

```
1  c = 1
2
3  def vartest3():
4      global c
5      c = c + 1
6
7  vartest3()
8  print(c)
```

함수

- 가변 매개변수 1 : * 매개변수 p.160
  - 인수의 개수가 정해져 있지 않는 경우에 사용,
  - positional arguments packing (위치 인자 패킹)
  - 관련 자료형 : Tuple
  - 정의 형태

    def 함수(***args**):

      실행문장 …

```python
def add_many(*args):
    result = 0
    for i in args:
        result = result + i
    return result


a = add_many(1, 2, 3)
b = add_many(1, 2, 3, 4, 5, 6)
print(a, "/", b)
```

```python
def add_mul(choice, *args):
    if choice == "add":
        result = 0
        for i in args:
            result += i
    elif choice == "mul":
        result = 1
        for i in args:
            result *= i
    else:
        result = None
    return result


a = add_mul("add", 1, 2, 3)
b = add_mul("mul", 1, 2, 3, 4, 5, 6)
print(a, "/", b)
```

함수

- 가변 매개변수 2 : ** 매개변수 p.162
  - 인수의 개수가 정해져 있지 않는 경우에 사용
    - 단, 앞의 가변 매개변수와 달리 key=값 형태로 인수를 전달
  - keyword arguments packing (키워드 인자 패킹)
  - 관련 자료형 : Dictionary
  - 정의 형태

    def 함수(**kwargs):

      실행문장 …

```
einstein albert의 추가정보는 아래와 같습니다.
 활동지역 : princeton
 분야 : 정보없음
kim inha의 추가정보는 아래와 같습니다.
 활동지역 : incheon
 분야 : cs
lee inha의 추가정보는 아래와 같습니다.
 활동지역 : incheon
 분야 : lg
```

```python
def build_profile(first, last, **userinfo):
    print(f"{last} {first}의 추가정보는 아래와 같습니다.")
    print(" 활동지역 :", userinfo.get('loc', '정보없음'))
    print(" 분야 :", userinfo.get('field', '정보없음'))


build_profile('albert', 'einstein', loc='princeton')
build_profile(last='kim', first='inha', loc='incheon', field='cs')
build_profile('inha', 'lee', loc='incheon', field='lg')
```

주의!

- 코드 작성시 함수 정의는 호출보다 반드시 먼저 기술할 것

```
1  build_profile('albert', 'einstein', loc='princeton')
2  build_profile(last='kim', first='inha', loc='incheon', field='cs')
3  build_profile('inha', 'lee', loc='incheon', field='lg')
4
5  def build_profile(first, last, **userinfo):
6      print(f"{last} {first}의 추가정보는 아래와 같습니다.")
7      print(" 활동지역 :", userinfo.get('loc', '정보없음'))
8      print(" 분야 :", userinfo.get('field', '정보없음'))
```

```
"C:\Program Files\Python38\python.exe" E:/Python1/pythonProject/test04.py
Traceback (most recent call last):
  File "E:/Python1/pythonProject/test04.py", line 1, in <module>
    build_profile('albert', 'einstein', loc='princeton')
NameError: name 'build_profile' is not defined

Process finished with exit code 1
```

참고코드

파라미터

```python
def describe_pet(type, name) :
    print(f"\n나는 {type}라는 동물을 데리고 있어요.")
    print(f"내 {type}의 이름은 {name} 입니다.")

describe_pet('햄스터', '해리')
describe_pet('개', '방울이')
describe_pet('소리', '카나리아')
describe_pet(type='소리', name='카나리아')
describe_pet(name='소리', type='카나리아')
describe_pet('소리', name='카나리아')
```

```python
def describe_pet(name, type='개') :
    print(f"\n나는 {type}라는 동물을 데리고 있어요.")
    print(f"내 {type}의 이름은 {name} 입니다.")

describe_pet('방울이1')
describe_pet(name='방울이2')
describe_pet('방울이3', '햄스터')
describe_pet(name='방울이4', type='햄스터')
describe_pet(type='햄스터', name='방울이5' )
```

단순 반환 값

```
1   def get_formatted_name(first, last) :
2       fullname = f"{first} {last}".title()
3       return fullname
4
5   musician = get_formatted_name('jimi', 'hendrix')
6   print(musician)
```

```
8   def get_formatted_name_kor(first, last, opt='') :
9       if opt == "kor" :
10          fullname = f"{last}{first}"
11      else :
12          fullname = f"{first} {last}".title()
13      return fullname
14
15  musician = get_formatted_name_kor('jimi', 'hendrix')
16  print(musician)
17  musician = get_formatted_name_kor('석진', '김', 'kor')
18  print(musician)
```

딕셔너리 반환 값

```python
def get_formatted_name(first, last) :
    fullname = f"{first} {last}".title()
    return fullname

def build_person(first, last, age=None) :
    person = {'f' : first , 'l' : last}
    if age :
        person['a'] = age
    return  person

muisician_list = []
muisician_list.append(build_person('jimi', 'hendrix'))
muisician_list.append(build_person('sukjin', 'kim', age=27))

for m in muisician_list :
    print(get_formatted_name(m['f'], m['l']))
```

반복문에서 함수 호출하기

```
1   def get_formatted_name(first, last) :
2       fullname = f"{first} {last}"
3       return fullname
4
5   while True :
6       print("\nTell me your name (quit :'q')")
7
8       f_name = input("First Name:")
9       if f_name.lower() == 'q' :
10          break
11
12      l_name = input("Last Name:")
13      if l_name.lower() == 'q':
14          break
15
16      formatted_name = get_formatted_name(f_name, l_name)
17      print(f"Hello, {formatted_name}")
```

리스트를 함수에 전달

```
1  def greet_users(names) :
2      for name in names :
3          msg = f"hello, {name.title()}"
4          print(msg)
5
6  usernames = ['kim inha', 'park inha']
7  greet_users(usernames)
```

리스트를 함수에서 수정

```
25  def print_models(unprinted_design, completed_models) :
26      while unprinted_design:
27          current_design = unprinted_design.pop()
28
29          print(f"Printing Mode : {current_design}")
30          completed_models.append(current_design)
31
32  def show_completed_models(completed_models) :
33      print("The following models have been printed:")
34      for cm in completed_models:
35          print(cm)
36
37
38  unprinted_design = ['phone case', 'robot pendant', 'dodecahedron']
39  completed_models = []
40
41  print_models(unprinted_design,completed_models )
42  show_completed_models(completed_models)
```

실습코드

숙제는 아니지만 직접 풀어보기

리스트의 중복 항목 개수 세기 (각자 해보기)

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

duplicate_counts = count_duplicates(numbers)

print(f"중복 항목: {duplicate_counts}")
```

```
중복 항목: {2: 3, 3: 2}
```

리스트에서 특정 값 지우기

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

value_to_remove = 2

updated_list = remove_value(numbers, value_to_remove)

print(f"제거 후 리스트: {updated_list}")
```

```
제거 후 리스트: [1, 3, 4, 5]
```

딕셔너리 값의 평균 계산하기

```
student_scores = {“김인하": 92, "이인하": 85, "박인하": 78}
avg_score = calculate_average_from_dict(student_scores)
print(f"평균 점수: {avg_score:.2f}")
```

```
평균 점수: 85.00
```