AI전문가과정 - Python Review

Practice Session #2

Function

오늘의 실습(Function)

함수의 기본 구조 복습

- 함수의 정의

  def *name of function* (list of *formal parameters*):
      body of function

- Example:

```
def maxVal(x, y):   # function header
    """Return maximum of x and y."""
    if x > y:
        return x  # 값을 Return한다
    else:
        return y
```

오늘의 실습(Function)

함수의 Return

- 함수는 값을 return할 수 있습니다.
  - 함수를 호출하면서 변수를 할당하면 return된 값을 할당할 수 있습니다.
- return은 함수 내에서 여러번 사용될 수 있습니다. (중요!)
  - return 에 값을 명시하지 않을경우 "None"을 return합니다.

```
def greater(a, b) :
    if(a>b) :
        return a
    elif (a==b) :
    return
  return b

print(greater(30,5))
print(greater(10,20))
# None
print(greater(10,10))
```

Run Code | Visualize

오늘의 실습(Function)

함수의 Return

- return이 아예 없는 함수도 생성할 수 있습니다.
  - 함수는 반드시 return 을 포함하지 않아도 됩니다.

```
def hello(): # parameter가 없음
    return "Hello"

# return이 없는 함수
def print_hello():
    print("Hello")

print(hello())
a = print_hello()

# return된 값이 없기에 a는 None을 출력
print(a)
```

Run Code | Visualize

오늘의 실습(Function)

함수의 Return

- 여러 Value를 Return하는 함수를 만들 수도 있습니다.
  - 이때 return된 값의 type은 tuple이 됩니다.

```
def ret_two():
    a = 10
    b = 20
    c = [30, 40]
    return a, b, c

# (10, 20, [30, 40])
test = ret_two()
print(test)

# a = 10, b = 20, c = [30,40]
a, b, c = ret_two()
print(a,b,c)
```

Run Code | Visualize

오늘의 실습(Function)

함수의 Return

- Return 뒤에 value를 명시하지 않을 경우 함수를 탈출하는 용도로 사용됩니다

```
def print_only_add(option, *args):
  result = 0
  if option == "add":
      for i in args:
        result = result + i
  else:
      return  # 함수 탈출

  print(result)

# 6 출력
print_only_add("add",1,2,3)
# 아무 것도 출력 안됨
print_only_add("mul",1,2,3)
```

Run Code | Visualize

오늘의 실습(Function)

Argument의 종류

- Positional Argument 와 Keyword Argument
  - 함수 호출시 argument name을 명시하지 않으면 Positional Argument
  - Argument name을 명시하면 Keyword Argument 형태로 처리됩니다.

```
def f(x, y, z):
    print(x,y,z) # x,y,z 출력
    return x + (2*y) + (3*z)

# Positional Argument : 5,3,2가 x,y,z 에 순서대로 들어감
a = f(5,2,3)

# Keyword Argument : z, y의 argument name을 명시함
b = f(5, z=3, y=2)

print(a)
print(b)
```

Run Code | Visualize

오늘의 실습(Function)

Argument의 종류

- Keyword Argument의 예제
  - Keyword Argument는 순서에 상관없기 때문에 개발 효율성을 향상시킵니다
  - 대신 함수의 parameter 이름을 잘 설정해줘야 합니다

```
# parameter의 이름이 이해하기 쉽게 쓰여져 있다.
def printInfo(name, sid, age):
  print("Name : " + name)
  print("S_ID : " + sid)
  print("AGE : " + str(age))

# 따라서 함수 호출 시에 Keyword Argument를 이용해서 가독성을 높였다.
printInfo(age = 30, name ="Sebeom", sid = "2021-12345")
```

Run Code Visualize

오늘의 실습(Function)

Default(Argument) Values

- Default Values는 parameter 값이 전달되지 않더라도 기본값을 할당합니다.
  - Default Value용 변수는 argument가 전달되지 않아도 오류가 나지 않습니다.
  - parameter의 Non-Default value는 Default value보다 선행되어야 합니다.

```
def f(x, y=10, z=100):
    return x + y + z

print(f(5, z=3)) # x=5, y=10, z=3으로 할당
print(f(5, 3))  # x=5, y=3, z=100으로 할당
print(f(5))  # x=5, y=10, z=100으로 할당
print(f())  # Syntax Error!! x의 Default Value 없음
```

Run Code | Visualize

오늘의 실습(Function)

Default(Argument) Values의 활용

- Default Value를 함수 내부에서 사용할 수 있습니다.
  - 하지만 mutable object는 default value로 사용하면 디버깅이 어려워집니다
  - default value는 가급적 immutable object를 사용해주세요

```
def append2List(a, L = []):
    L.append(a)
    return L
print(append2List(1))   # prints [1]
print(append2List(2))   # prints [1, 2]
print(append2List(3))   # prints [1, 2, 3]
```

Run Code Visualize

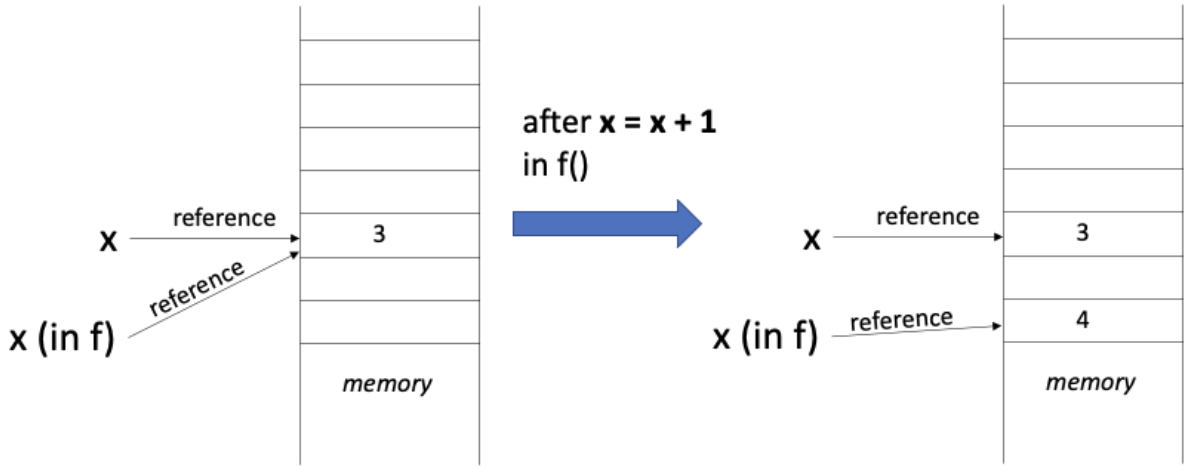
오늘의 실습(Function)

Scoping

- 함수 내에 선언된 변수는 함수 내에서만 유효하게 사용됩니다
  - Scoping에 의해 밖의 변수와 이름이 같아도 별도로 메모리를 할당합니다

```
def f(x):
    x = x + 1
    print(x)
    # prints 4

x = 3
f(x)
print(x)
# prints 3
```



오늘의 실습(Function)

Scoping

- 실제 함수 내부에서 같은 이름을 가진 변수에 대해
  - id 값을 출력하면 서로 다른 값을 가지는 것을 알 수 있습니다.

```
a = 1
print(id(a))
def add_one(a):

    a = a+1
    print(id(a)) # 처음 ID값과 다름
    print("inside add_one : " + str(a)) # a = 2

add_one(a)
print("outside add_one : " + str(a)) # a = 1
```

Run Code Visualize

오늘의 실습(Function)

Scoping

- 함수 내부에서 밖에 변수를 사용하고 싶을 경우 global keyword를 사용하는데
  - 다만 이는 디버깅을 어렵기 하기에 사용시에는 **매우 주의하셔야 합니다**

```
a = 1
def add_one():
    global a
    a = a +1
    print("inside add_one : " + str(a))

add_one()
print("outside add_one : " + str(a))
```

Run Code Visualize

오늘의 실습(Function)

Arbitrary Argument Lists

- argument의 개수를 모를 때에는 * keyword를 사용합니다.
  - 아래 코드는 *args는 tuple 형식으로 전달됩니다.
  - args는 arguments의 약자로 관례적으로 많이 사용됩니다.

```
def add_many(*args):
    print(args) # Tuple 형태로 출력
    result = 0
    for i in args:
        result+=i
    return result

# 인자가 5개인 경우
print(add_many(1,2,3,4,5))

# 인자가 3개인 경우
print(add_many(1,2,3))
```

Run Code Visualize

오늘의 실습(Function)

Arbitrary Argument Lists

- argument의 활용 예시
  - args를 이용하여 argument의 개수를 상관없이 함수 로직을 만들 수 있습니다.

```
def add_mul(option, *args):
  if option == "add":
    result = 0
    for i in args:
      result = result + i
  elif option == "mul":
    result = 1
    for i in args:
      result = result * i
  return result

print(add_mul("add",1,2,3,4))
print(add_mul("mul",1,2,3,4))
```

Run Code | Visualize

오늘의 실습(Function)

Keyword Argument 이해 (kwargs)

- Parameter에 ** Keyword를 붙여 Keyword Argument를 만들 수 있습니다.
  - Keyword argument는 Dictionary 형태로 전달됩니다.
  - 함수 호출시 paramter=value 형태로 전달해야 합니다

오늘의 실습(Function)

```
def sum(*values, **options):
    """
    values: passed as a tuple
    options: passed as a dictionary
    """

    sum = 0
    answer = ''

    for i in values:
        sum = sum + i

    if 'neg' in options:
        if options['neg']:
            sum = -sum

    if 'explain' in options:
        if options['explain']:
            answer = "The answer is "

    return answer + str(sum)
```

- `**options` expects keyword arguments
  - of the form `parameter = value` pairs
  - passed as a `dictionary` to the function

```
>>> sum(1, 2, 3)
'6'
>>> sum(1, 2, 3, neg = True)
'-6'
>>> sum(1, 2, 3, neg = False)
'6'
>>> sum(1, 2, 3, explain = True)
'The answer is 6'
>>> sum(1, 2, 3, neg=True,explain=True)
'The answer is -6'
```

오늘의 실습(Function)

Parameter 값 복사 / 전달의 이해

- 함수의 Parameter에 **Immutalble type**의 Object가 전달될 경우 값이 복사됩니다.
  - 따라서 함수 내부의 tuple의 id와 함수 밖의 tuple의 id가 다르게 출력됩니다.
  - 또한 함수로 전달한 Tuple의 값을 바꿀 수 없습니다.

```
def replace(my_tuple):
    my_tuple = (1,2,3,4)
    print(id(my_tuple))
    print(my_tuple)

a = (1,2,3)
print(id(a))

replace(a)
print(a)
```

Run Code | Visualize

오늘의 실습(Function)

Parameter 값 복사 / 전달의 이해

- 함수의 Parameter에 **Mutable type**의 Object가 전달될 경우 값이 전달됩니다.
  - 따라서 argument로 전달했던 Object가 함수 내에서 수정가능하게 됩니다.

```
def list_append(my_list):
    my_list.append(4)
    print(id(my_list))

a = [1,2,3]
print(id(a))
list_append(a)
print(a)
```

Run Code | Visualize

오늘의 실습(Function)

Parameter 값 복사 / 전달의 이해

- dictionary type의 mutable object 역시 값이 전달 되는 형태입니다.

```
def change_age(my_dict) :
  if 'age'in my_dict:
    my_dict['age'] = 15
info = {'name' : 'sebeom', 'age' : 50}

# {'name': 'sebeom', 'age': 50}
print(info)
change_age(info)
#{'name': 'sebeom', 'age': 15}
print(info)
```

Run Code Visualize

오늘의 실습(Function)

Parameter 값 복사 / 전달의 이해

- 함수에서 새로 선언된 변수를 밖에서 참조한다면 이를 계속 사용할 수 있습니다.

```
def ret_new_list():

  my_list = [1,2,3]
  print(id(my_list))
  return my_list

# a는 my_list를 가리킴
a = ret_new_list()
# 함수내 선언된 local variable과 같은 id를 가짐
print(id(a))
print(a)
```

Run Code Visualize

오늘의 실습(Function)

labmda 함수의 이해

- lambda를 활용하여 Anonymous Functions(이름이 없는 함수)를 만들 수 있습니다.
  - 보통 lambda는 함수를 간결하게 1줄로 표현하기 위해 많이 사용됩니다
  - add_lambda는 add 함수와 동일하지만 이름이 없는 형태의 함수입니다

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

add_lambda = lambda a,b : a+b

print(add(1,2))
print(add_lambda(1,2))
```

Run Code | Visualize

오늘의 실습(Function)

labmda 함수의 이해

- lamda expression 안에서는 새로운 변수를 만들 수 없습니다.
  - 따라서 return 되는 반환값 부분은 변수 없이 한줄로 표현을 해야만 합니다.

```
(lambda x: y = 10; x + y)(1)
# SyntaxError: invalid syntax
```

Run Code Visualize

오늘의 실습(Function)

labmda 함수의 활용

- lambda 함수는 Python의 일부 Built-in function 에서 효과적으로 사용됩니다.
- Built-in Function 예시 : filter, map, reduce 등
  - Built-in Function 'map'을 활용한 예시
  - map 함수는 iterable의 요소를 하나씩 꺼내어 함수에 넣고 리턴된 값으로 새로운 iterator를 만듭니다.

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

list_mul_2 = list(map(lambda x : x*2, a))

# [2,4,6,8,10]
print(list_mul_2)
```

Run Code | Visualize

오늘의 실습(Function)

labmda 함수의 활용

- Built-in Function 'filter'를 활용한 예시
  - filter함수는 iterable의 요소를 하나씩 꺼내어 함수에 넣고 'True' 리턴값을 반환하는 요소의 iterator를 만듭니다.

```
source_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

filtered_list = list(filter(lambda x : (x % 2 == 0), source_list))

print(filtered_list)   # prints [2, 4, 6, 8, 10]
```

Run Code Visualize

String 내장함수(operations)

string operations

- `strip`: 좌우에 오는 특정 문자를 제거함 (기본: 공백문자)
- `lstrip`: 왼쪽에 있는 특정 문자를 제거함 (기본: 공백문자)
- `rstrip`: 오른쪽에 있는 특정 문자를 제거함 (기본: 공백문자)

```
s = "   hello  "
print('[' + s + ']')
print('[' + s.strip() + ']')
print('[' + s.lstrip() + ']')
print('[' + s.rstrip() + ']')
```

Run Code Visualize

String 내장함수(operations)

string operations

- `str.replace(a, b)`: 문자열 내에 등장하는 모든 `'a'`를 `'b'`로 바꿈

```
s = "   hello  "
print('[' + s.replace(' ', '') + ']')
print('[' + s.replace('l', 'L') + ']')
print('[' + s.replace(' ', '').replace('l', 'L') + ']')
```

Run Code | Visualize

String 내장함수(operations)

string operations

- replace는 string 자체를 바꾸지 않음

```
s = "   hello  "
t = s.replace(' ', '')
print('[' + s + ']')
print('[' + t + ']')
```

Run Code | Visualize

String 내장함수(operations)

string operations

- `str.count(s)`: 개수
- `str.split(s)`: s를 기준으로 쪼갬
- `str.find(s)`: 처음으로 등장하는 위치(index)

```
s = 'lorem ipsum dolor sit amet'
print(s.count('l'))
print(s.split())
print(s.split('l'))
print(s.find('l'))
```

Run Code Visualize

String 내장함수(operations)

string operations

- `str.lower()`: 모두 소문자로 바꿈
- `str.upper()`: s를 기준으로 쪼갬
- `str.`: 처음으로 등장하는 위치(index)

```
s = 'Hello World'
print(s.lower()) # 'hello world'
print(s.upper()) # 'HELLO WORLD'
```

Run Code Visualize

오늘의 실습

연습 문제 1. Palidrome(회문)

- 입력된 문장이 Palidrome 형태인지 확인하는 함수를 만들어 봅시다.
  - 한글 예시 : 탄도유도탄, 지은이 이름이 이은지
  - 영어 예시 : Anna, no lemon.. no melon
- 하나의 문장을 입력받고 Palidrome이면 True를 아니면 False를 출력합니다.

입력

```
racecar
```

출력

```
True
```

오늘의 실습

실습 문제 1. 컴개실 드랍 학생 찾기

- 최초 컴개실 수업에 참여한 학생들의 이름이 담긴 리스트와
  끝까지 수료한 학생들의 이름이 담긴 리스트가 입력으로 주어질 때
  드랍한(수료하지 못한) 학생들의 이름을 출력하는 프로그램을 작성해주세요

오늘의 실습

실습 문제 1. 컴개실 드랍 학생 찾기

입력

- 첫번째 줄에는 최초 컴개실 수업에 참여한 전체 학생 리스트가 입력됩니다.
- 두번째 줄에는 수료날까지 남아있던 학생리스트가 입력됩니다.

```
sebeom aeri subin hangyeol
aeri subin
```

출력

- 수강을 철회한 학생들의 이름이 사전순으로 차례대로 출력됩니다

```
hangyeol sebeom
```

오늘의 실습

실습 문제 2. 암호화 복호화

- 글자를 코드로 치환하여 암호를 만드는 방식을 활용하여 string을 암호화 및 복호화 하는 프로그램을 만들어 봅시다.
- 아래와 같은 코드북이 있다고 가정해봅시다.
  - `H` : `%`
  - `e` : `9`
  - `l` : `@`
  - `o` : `#`
- 암호화는 위처럼 코드북이 구성이 되면, 왼쪽에 해당하는 글자를 오른쪽으로 바꾸는 과정을 거칩니다.
  - `Hello!` → `%9@@#!`
- 복호화는 위처럼 코드북이 구성이 되면, 오른쪽에 해당하는 글자를 왼쪽으로 바꾸는 과정을 거칩니다.
  - `%9@@!` → `Hell!`
- 코드북에 없는 글자는 암호화/복호화 하지 않고 그대로 출력합니다.

오늘의 실습

실습 문제 2. 암호화 복호화

입력

- Input은 command, original, code, sentence 순으로 주어집니다.
  - command: encrypt or decrypt
  - original: 바뀌기 전 글자 목록이 들어옵니다.
  - code: original을 바꿀 글자 목록이 들어옵니다.
  - sentence: 암호화/복호화 할 문장이 주어집니다.

```
encrypt
Hdelkmw
!@#$*-j
Hello, world!
```

출력

- Output은 sentence를 command에 맞게 암호화/복호화 한 문장을 출력합니다.

```
!#$$o, jor$@!
```