# 파이썬 스터디

## 05장 함수 문제풀이 ¶

### 파이썬 타입 어노테이션 type annotation

파이썬은 동적 프로그래밍 언어로 변수의 타입을 지정하지 않는다. 파이썬과 다르게 자바와 C언어의 경우 변수를 선언할 때 자료형을 선언해야 한다. `int num1 = 2` 이렇게 변수를 선언한다.
파이썬은 변수를 선언할 때 제약이 적고 유연성이 높다. 그래서 자료형 선언없이 변수를 선언하고 함수를 만들 수 있었다. 그러나 해당 함수를 실행하기 전까지 함수의 파라미터의 타입을 확인할 수 없다. 가령 int형을 넣어야하는데 int형을 넣어야할지 몰라 str형으로 넣은 경우 오류가 발생한다. 이를 방지하기 위해 파이썬에서는 **annotation(어노테이션)** 이라는 주석달기 기능이 생겼다.
annotation은 주석달기로, 함수 파라미터의 타입과 리턴값의 타입을 알려준다. 어노테이션으로 알려준 타입에는 강제성이 없다. `def func(a:int)-> int` 라고 했을 때 a에 float을 넣어도 오류가 나지 않는다. 즉 annotation은 주석달기로, 코드 자체에 어떠한 영향도 미치지 않는다.

- 사용법

```
def func(a: int) -> str:
  <수행할 문장 1>
  ...
  return value
```

> func함수의 a 파라미터의 타입은 int이며, 리턴값의 타입은 str이다.

#### typing 모듈

annotation을 적을 때 int, float, str, bool 등을 말하는 것이 아닌 모든 타입 가능, Sequence형 등 특수한 경우를 typing 모듈을 통해 지정할 수 있다.

- **Any** : 제약이 없는 임의의 자료형
- **Sequence** : 시퀀스형에는 리스트형, 바이트형, 문자열형, 튜플형 바이트열형이 있다.

```
from typing import Sequence, Any

def func(a: Any) -> Sequence:
    <수행할 문장 1>
    ...
    return value
```

> func함수의 a 파라미터는 제약이 없으며, 리턴값은 Sequence형으로 리스트, 바이트, 문자열, 튜플 등이 될 수 있다.

#### 문제 01)

원의 둘레를 계산하는 함수 get_peri(radius)을 정의하고 테스트 한다. 만약 원의 반지름이 주어지지 않았다면 5.0으로 간주한다. 함수의 기본 인수를 사용해라.

In [1]:

```
def get_peri(radius: float = 5.0) -> float:
    """원의 둘레 계산"""
    import numpy as np
    return 2 * np.pi * radius
```

In [2]:

```
get_peri()
```

Out[2]:

```
31.41592653589793
```

In [3]:

```
get_peri(4.0)
```

Out[3]:

```
25.132741228718345
```

**문제 02)**

덧셈, 뺄셈, 곱셈, 나눗셈을 수행하는 함수를 각각 작성하고 테스트하라.

```
첫 번째 정수를 입력하세요: 10
두 번째 정수를 입력하세요: 20
(10 + 20) = 30
(10 - 20) = -10
(10 * 20) = 200
(10 / 20) = 0.5
```

In [4]:

```python
def add(a: int, b: int) -> int:
    """더하기"""
    print(f"({a} + {b}) = {a + b}")
    return a + b

def subtract(a: int, b: int) -> int:
    """빼기"""
    print(f"({a} - {b}) = {a - b}")
    return a - b

def mul(a: int, b: int) -> int:
    """곱하기"""
    print(f"({a} * {b}) = {a * b}")
    return a * b

def div(a: int, b: int) -> float:
    """나누기"""
    print(f"({a} / {b}) = {a / b if b != 0 else None}")
    return a / b if b != 0 else None
```

In [5]:

```python
num1 = int(input('첫 번째 정수를 입력하세요: '))
num2 = int(input('두 번째 정수를 입력하세요: '))
```

```
첫 번째 정수를 입력하세요: 10
두 번째 정수를 입력하세요: 20
```

In [6]:

```python
add(num1, num2)
subtract(num1, num2)
mul(num1, num2)
div(num1, num2)
```

```
(10 + 20) = 30
(10 - 20) = -10
(10 * 20) = 200
(10 / 20) = 0.5
```

Out[6]:

```
0.5
```

**문제 03)**

2번에서 4개의 함수를 작성했다. 하나의 함수 calc() 안에 덧셈, 뺄셈, 곱셈, 나눗셈을 모두 수행하고 4개의 계산값을 동시에 반환하도록 수정해보자.

In [7]:

```
from typing import Sequence

def calc(a: int, b: int) -> Sequence:
    """곱셈, 뺄셈, 곱셈, 나눗셈"""
    return a + b, a - b, a * b, a / b if b != 0 else None
```

In [8]:

```
print(f"{calc(num1, num2)}가 반환됐습니다")
```

(30, -10, 200, 0.5)가 반환됐습니다

**문제 04)**

성적이 90점 이상이면 A, 80점 이상이면 B, 70점 이상이면 C, 60점 이상이면 D, 그외에는 F를 반환하는 함수 getGrade(score)를 작성하고 테스트하라.

> 점수를 입력하세요: 83
> 성적은 B입니다.

In [9]:

```
def getGrade(score: int) -> str:
    """성적으로 학점구하기"""
    if score > 90:
        return 'A'
    elif score > 80:
        return 'B'
    elif score > 70:
        return 'C'
    elif score > 60:
        return 'D'
    else:
        return 'F'
```

In [10]:

```
score = int(input('점수를 입력하세요: '))
print(f"성적은 {getGrade(score)}입니다.")
```

```
점수를 입력하세요: 83
성적은 B입니다.
```

**문제 05)**

패스원드를 검증하는 함수 checkPass(p)를 작성하고 테스트하라. 패스워드에는 적어도 8글자 이상이어야 한다. 또 적어도 1글자의 대문자와 소문자가 들어가야 한다. 또 적어도 1개의 숫자가 들어가야 한다.

> 패스워드를 입력하세요: abcdefgh
> 사용할 수 없습니다. 다시 입력하세요!
> 패스워드를 입력하세요: abcdefG1
> 사용할 수 있습니다.

In [11]:

```
def checkPass(p: str) -> None:
    """패스워드 사용 여부"""
    up = [i for i, v in enumerate(p) if v.isupper()]
    low = [i for i, v in enumerate(p) if v.islower()]
    num = [i for i, v in enumerate(p) if v.isdecimal()]
    if (len(p) >= 8) & (len(up) > 0) & (len(low) > 0) & (len(num) > 0):
        print('사용할 수 있습니다.')
    else:
        print('사용할 수 없습니다. 다시 입력하세요!')
```

In [12]:

```
password = input('패스워드를 입력하세요: ')
checkPass(password)
```

```
패스워드를 입력하세요: abcdefg
사용할 수 없습니다. 다시 입력하세요!
```

In [13]:

```
password = input('패스워드를 입력하세요: ')
checkPass(password)
```

```
패스워드를 입력하세요: abcdefG1
사용할 수 있습니다.
```

**문제 06)**

사용자로부터 2개의 정수를 받아 수학 문제를 만들어 화면에 출력하는 함수를 작성하고 테스트하시오.

In [14]:

```
def problem(a: int, b: int) -> None:
    """문제 출력하기"""
    print(f"{a} + {b}의 합은?")
```

In [15]:

```
problem(num1, num2)
```

```
10 + 20의 합은?
```

**문제 07)**

사용자가 일정 구간의 값을 입력할 때까지 사용자에게 입력을 요청하는 함수 getIntRange(a, b)를 작성하고 테스트하라. 이 함수를 테스트하기 위해 날짜(월, 일)을 입력받아 출력하는 프로그램을 작성해보자.

```
날짜를 입력하시오(월과 일)
월을 입력하시오(1부터 12사이의 값): 13
월을 입력하시오(1부터 12사이의 값): 1
일을 입력하시오(1부터 31사이의 값): 32
일을 입력하시오(1부터 31사이의 값): 3
입력된 날짜는 1월 3일 입니다.
```

In [16]:

```
def getIntRange(a: int, b: int) -> None:
    """사용자가 일정 구간의 값을 입력할 때까지 사용자에게 입력을 요청하는 함수"""
    print('날짜를 입력하시오(월과 일)')
    while True:
        mon = int(input(f'월을 입력하시오(1부터 {a}사이의 값): '))
        if mon in range(1, a + 1): break

    while True:
        day = int(input(f'일을 입력하시오(1부터 {b}사이의 값): '))
        if day in range(1, b+1): break
    print(f'입력된 날짜는 {mon}월 {day}일 입니다.')
```

In [17]:

```
getIntRange(12, 31)
```

```
날짜를 입력하시오(월과 일)
월을 입력하시오(1부터 12사이의 값): 13
월을 입력하시오(1부터 12사이의 값): 1
일을 입력하시오(1부터 31사이의 값): 32
일을 입력하시오(1부터 31사이의 값): 3
입력된 날짜는 1월 3일 입니다.
```

**문제 08)**

우리는 중요한 계약서에는 숫자로 된 금액(예를 들어서 1,000,000원)과 문자로 된 금액(백만원)을 동시에 기재한다. 숫자는 변조하기 쉽기 때문이다. 숫자를 입력받아 문자로 바꾸어 출력하는 함수 getMoneyText(amount)를 작성하고 테스트하라. 단 숫자는 1000이하라고 가정한다.

In [18]:

```
def getMoneyText(amount: int) -> str:
    num = [' ', '일', '이', '삼', '사', '오', '육', '칠', '팔', '구']
    s = ''
    unit = ['백 ', '십 ' , '원']
    if amount == 1000:
        print('천원')
    for i, n in enumerate([100, 10, 1]):
        nn = amount // n
        amount -= nn * n
        s += num[nn] + unit[i]
    print(s)
```

In [19]:

```
getMoneyText(999)
```

```
구백 구십 구원
```

In [20]:

```
getMoneyText(156)
```

```
일백 오십 육원
```

**문제 09)**

사용자로부터 두 개의 정수를 입력받아 최대 공약수를 찾는 함수를 작성해보자. 가장 간단한 알고리즘을 생각하자.

In [21]:

```
def gcd_func(x: int, y: int) -> int:
    for i in range(min(x, y) + 1, 1, -1):
        if (x % i == 0) & (y % i == 0):
            return i
            break
```

In [22]:

```
gcd_func(num1, num2)
```

Out[22]:

```
10
```

In [23]:

```
# 유클리드 호제법: 재귀알고리즘 활용하기
def gcd(x: int, y: int) -> int:
    if y == 0:
        return x
    else:
        return gcd(y, x % y)
```

In [24]:

```
gcd(num1, num2)
```

Out[24]:

```
10
```

In [25]:

```
# 유클리드 호제법 - 최대공약수 함수
import math
math.gcd(num1, num2)
```

Out[25]:

```
10
```

**문제 10)**

주어진 정수가 소수인지 검사하는 함수 testPrime(n)를 작성하고 이 함수를 호출해 2부터 100사이의 소수를 출력해보자.

In [26]:

```
def testPrime(n: int) -> bool:
    for i in range(2, n + 1 ):
        if n % i == 0:
            break
    if n == i: return True
    else: return False
```

In [27]:

```
for i in range(2, 101):
    if testPrime(i):
        print(i, end = ' ')
```

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

In [ ]:

```

```