2023. 2학기

웹/파이썬프로그래밍 01
기말고사

담당 교수: 박상근

- 밑줄학번.py 형식으로 모듈을 생성하세요 (for example, _2023123456.py), otherwise -1P.
- 제출 전에 print 함수는 삭제 또는 주석처리하세요, otherwise -1P.
- input 함수 절대 사용하지 마세요, otherwise -1P.
- 해당 모듈을 import 할 때 문제(에러)가 발생하면, you get -1P.
- 오타 조심!

시험 파일을 제출 하고 더 제출할 필요가 없다고 판단되면, 노트북을 덮고 종이로 된 시험 자료로 다른 과목 시험을 공부할 수 있습니다. 단, 태블릿이나 스마트폰 등의 디지털기기는 절대 불가!

***str* Method**	**Description**
isdigit()	Returns True if all characters in the string are digits
islower()	Returns True if all characters in the string are lower case
isupper()	Returns True if all characters in the string are upper case
lower()	Converts a string into lower case
upper()	Converts a string into upper case

Define the following functions and classes.

function1 (4 point)

반복문을 사용해서 간단한 문제를 풀어봅시다.

- The function name: **function1**
- This function takes only one argument (**list type)**
  - The parameter name: **sum**
  - The argument is **a list** that consists of only **int** type values
- The return value: **int type**
  - 주어진 리스트에 존재하는 모든 숫자의 합을 리턴
  - 주어진 리스트가 빈 리스트라면 0을 리턴
- This function will be tested as follows:
  - print(function1(sum=[ ])) # 0 (because there is no value in the list)
  - print(function1(sum=[1])) # 1 (because there is only one value, 1)
  - print(function1(sum=[2, 3])) # 5 (because 2+3)
  - print(function1([4, 5, 6])) # 15 (because 4+5+6)
  - print(function1([-1, 0, 1, 2])) # 2 (because -1+0+1+2)

function2 (4 point)

피타고라스 방정식 $a^2+b^2=c^2$ 개념을 활용하여, 삼각형의 세 변의 길이 3개가 주어졌을 때 이 세 변의 값이 피타고라스의 방정식을 만족할 수 있는지 판단하는 함수를 만들어 봅시다.

- The function name: **function2**
- This function takes three positional arguments
  - First argument: **int type**
  - Second argument: **int type**
  - Third argument: **int type**
- The return value: **bool type**
  - 피타고라스 방정식을 만족한다면 True, 그렇지않으면 False 리턴
  - bool type의 True 또는 False를 반환해야 합니다
  - str type의 'True' 또는 'False'를 반환하면 틀립니다
- This function will be tested as follows:
  - print(function2(3, 4, 5)) # True (because $5^2 = 3^2 + 4^2$)
  - print(function2(5, 3, 4)) # True (because $5^2 = 3^2 + 4^2$)
  - print(function2(1, 2, 3)) # False
  - print(function2(6, 1, 5)) # False
  - print(function2(12, 13, 5)) # True (because $13^2 = 12^2 + 5^2$)

function3 (4 point)

일반적으로, 웹서비스 회원가입시 이미 존재하는 아이디는 새로 만들 수 없습니다.
내가 원하는 아이디가 이미 존재하는 아이디인지 확인하는 로직을 함수로 구현해 봅시다.

➔ 첫 번째 인자: 이미 존재하는 아이디의 리스트
➔ 두 번째 인자: 내가 만들고 싶어하는 아이디

- The function name: **function3**
- This function takes two positional arguments
  - First argument: **list type (**최소 **1**개 이상의 문자열이 존재하는 리스트**)**
  - Second argument: **str type (**길이가 **1** 이상인 문자열**)**
- The return value: **bool type**
  - 대소문자 구분없이, 두 번째 인자(str)가 첫 번째 인자(list)에 있으면 True
  - 대소문자 구분없이, 두 번째 인자(str)가 첫 번째 인자(list)에 없으면 False
  - bool type의 True 또는 False를 반환해야 합니다
  - str type의 'True' 또는 'False'를 반환하면 틀립니다
- This function will be tested with positional arguments, for example:
  - print(function3(['kim', 'lee'], 'KIM')) # True (because kim is Kim)
  - print(function3(['kim', 'Lee', 'park'], 'hong')) # False
  - print(function3(['kim', 'park'], 'LEE')) # False
  - print(function3(['kim', 'park', 'Hong'], 'HONG')) # True (because Hong is HONG)
  - print(function3(['park', 'LeE'], 'Lee')) # True (because LeE is Lee)

function4 (4 points)

회원가입시, 사용자가 너무 단순한 비밀번호를 만들면 해킹당할 확률이 높습니다. 사용자가 비밀번호를 만들 때 알파벳 대문자, 알파벳 소문자, 숫자를 모두 포함하는지 검사하는 함수를 만들어 봅시다.

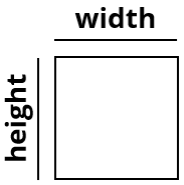
- The function name: **function4**
- This function takes a positional argument
  - The argument: **str type (**문자열의 길이는 **3** 이상**,** 알파벳과 숫자형으로 구성됨**)**
- The return value: **bool type**
  - 알파벳 대문자, 알파벳 소문자, 숫자형이 모두 포함되어있으면 True 리턴
  - 알파벳 대문자, 알파벳 소문자, 숫자형 중 하나라도 포함하지 않으면 False 리턴
  - bool type의 True 또는 False를 반환해야 합니다
  - str type의 'True' 또는 'False'를 반환하면 틀립니다
- This function will be tested with positional arguments, for example:
  - print(function4("abc123ABC")) # True
  - print(function4("1234LOVEyou")) # True
  - print(function4("a1b2c3d4")) # False (because 알파벳 대문자 없음)
  - print(function4("HelloWorld")) # False (because 숫자형 글자 없음)
  - print(function4("1234")) # False (because 알파벳 대문자, 소문자 없음)

function5 (4 points)

positional arguments와 keyword arguments를 구분하고, 이를 적절히 활용할 수 있는지 묻는 문제입니다.

- The function name: **function5**
- This function takes the unknown number of positional or keyword arguments
  - All arguments are **int types**
- The return value: **int type**
  - keyword argument 중 keyword가 'a'로 시작하는 모든 argument의 합 리턴
  - keyword가 'a'로 시작하는 keyword argument가 없다면 0을 리턴
- This function will be tested with positional or keyword arguments, for example:
  - print(function5(7, 8, a1=1, a2=2, b=9)) # 3 (because 1+2)
  - print(function5(ab=1, ac=3, bb=5)) # 4 (because 1+3)
  - print(function5(3, 4)) # 0 (because there is no argument that starts with 'a')
  - print(function5()) # 0 (because there is no argument)
  - print(function5(1, ban=1, app=5, Ac=3, axis=10)) # 15 (because 5+10)

class Rectangle [5 points]



Please complete the following **Rectangle** class so that you can see the [Result] when running the [Code].

- 이 클래스는 위 그림과 같이 height와 width를 갖는 사각형을 클래스화 한 것입니다.
- **__init__**, **get_area**, **is_square** 메소드가 미완성입니다. 이 세 메소드를 완성하세요.
- 클래스를 제대로 완성하면 아래 [Code]를 실행했을 때 [Result]대로 출력이 됩니다.
- 본인의 모듈에는 **Rectangle** 클래스만 만들면 됩니다. [Code]는 테스트 후 지우세요.

```
class Rectangle:
    def __init__():

    def get_area() -> int:
        """이 사각형의 넓이를 리턴 (int type)"""
        return
    def is_square() -> bool:
        """이 사각형이 정사각형이면 True, 아니면 False 리턴 (bool type)
        """
        return
```

[Code] → I will test your class using the following code

```
rect1 = Rectangle(10, 20)
rect2 = Rectangle(10, 10)
rect3 = Rectangle(5, 4)

print(rect1.get_area())  # 200
print(rect1.is_square()) # False
print(rect2.get_area())  # 100
print(rect2.is_square()) # True
print(rect3.get_area())  # 20
print(rect3.is_square()) # False
```

[Result]

```
200
False
100
True
20
False
```

class Num [5 points]

Please complete the following **Num** class so that you can see the [Result] when running the [Code].

- **__init__** 메소드는 이미 완성된 상태니 손대지 마세요.
- **__add__** 메소드가 미완성입니다. 아래와 같이 동작하도록 이 메소드를 완성하세요.
  - **Num** type + **Num** type 수행시, **Num** 인스턴스의 num 변수끼리 더한 값 리턴
  - **Num** type + **int** type 수행시, **Num** 인스턴스의 num 변수와 **int** 값을 더한 값 리턴
- 클래스를 제대로 완성하면 아래 [Code]를 실행했을 때 [Result]대로 출력이 됩니다.
- 본인의 모듈에는 **Num** 클래스만 만들면 됩니다. [Code]는 테스트 후 지우세요.

```
class Num:
    def __init__(self, num):
        self.num = num

    def __add__()->int:
        return
```

[Code] → I will test your class using the following code

```
n1 = Num(10)
n2 = Num(20)
n3 = Num(30)

print(n1+n2)   # 30
print(n1+40)   # 50
print(n1+(n2+n3))   # 60

n1.num += 10
print(n1.num, n2.num, n3.num)   # 20 20 30
```

[Result]

```
30
50
60
20 20 30
```

class User [5 points]

Please complete the **User** class so that you can see the [Result] when running the [Code].

- **User** 클래스를 아래 조건에 맞도록 완성하세요.
  - **__init__** 메소드는 이미 완성된 상태니 손대지 마세요.
  - **get_anonymized_userid** 메소드를 아래 주석을 참고해서 완성하세요.

```
class User:
    def __init__(self, userid: str):
        self.userid = userid
    def get_anonymized_userid
        """주어진 인자의 userid 변수 길이를 확인하고,
           길이가 2이하면 그 길이 만큼의 '*'를 리턴 (str type).
           길이가 3이상이면 userid 변수의 첫 글자와 마지막 글자 이외의 중간 글자를
           모두 '*'로 바꿔 리턴 (str type)."""
        return
```

[Code] → I will test your class using the following code

```
user1 = User("Park")
user2 = User("Kim")
user3 = User("Python")
user4 = User("Hi")
user5 = User("A")
print(user1.get_anonymized_userid())  # P**k
print(user2.get_anonymized_userid())  # K*m
print(user3.get_anonymized_userid())  # P****n
print(user4.get_anonymized_userid())  # **
print(user5.get_anonymized_userid())  # *
print(user1.userid) # Park
print(user2.userid) # Kim
print(user3.userid) # Python
print(user4.userid) # Hi
print(user5.userid) # A
```

[Result]

```
P**k
K*m
P****n
**
*
Park
Kim
Python
Hi
A
```

class Number [5 points]

Please complete the **Number** class so that you can see the [Result] when running the [Code].

- **__init__** 메소드는 이미 완성된 상태니 손대지 마세요.
- **__add__**메소드와 **get_sum** 메소드가 미완성입니다. 이 두 메소드를 완성하세요.
  - **__add__** 메소드는 두 인자를 받고, 각 인자의 value 변수를 더한(+) 값을 리턴.
  - **get_sum** 메소드는 두 인자를 받고, 두 인자를 더한(+) 값을 리턴.
- 두 메소드를 잘 완성하면 아래 [Code]를 실행시, [Result]와 같은 결과가 나옵니다.

```
class Number:
    def __init__(self, value):
        self.value = value

    def __add__
        return

    def get_sum
        return
```

[Code] → I will test your class using the following code

```
number1 = Number(10)
number2 = Number(20)
number3 = Number(30)

print(number1.get_sum(number2, number3))   # 50
print(number2.get_sum(number1, number3))   # 40

print(Number.get_sum(number1, number2))   # 30
print(Number.get_sum(10, 20))   # 30
```

[Result]

```
50
40
30
30
```

class Student [5 points]

Please complete the **Student** class so that you can see the [Result] when running the [Code].

- [Code]를 실행하면 [Result]와 같은 결과가 나오도록 **Student** 클래스를 완성하세요.
  - [힌트] **Student** 클래스는 클래스 변수 count 를 갖고 있습니다.
  - [힌트] 인스턴스를 생성할 때 마다 **Student** 클래스의 클래스 변수 count 의 값은 1씩 증가합니다.

```
class Student:


    def __init__


```

[Code] → I will test your class using the following code

```
print(Student.count)  # 0
s1 = Student("Park")
print(Student.count)  # 1
s2 = Student("Kim")
print(Student.count)  # 2
s3 = Student(name="Lee")
print(Student.count)  # 3
print(s1.name, s2.name, s3.name)      # Park Kim Lee
print(s1.count, s2.count, s3.count)   # 3 3 3
```

[Result]

```
0
1
2
3
Park Kim Lee
3 3 3
```

It does not matter how slowly you go as long as you do not stop!

"멈추거나 포기하지만 않는다면, 지금 남들보다 조금 느린 것 쯤은 문제가 아냐!"

THE END