2023. Fall

Web/Python Programming 01
Final exam

Professor: Sangkeun Park

- Name your Python file, _StudentNumber.py (for example, _2023123456.py), otherwise -1P.
- Remove or comment all **print** functions, otherwise -1P.
- Do not use the **input** function, otherwise -1P.
- If I fail to import your module due to your code, you get -1P.
- Be careful of typos (see function and class names).

After submitting the file, if you think you finished the exam, close your laptop (You cannot use digital devices in the class anymore.).
Then you can study for another exam using paper-based materials.

***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)

Let's use loop.

- 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**
  - sum of the int values in the list
  - If the given list is empty, return 0
- 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)

Let's create a Python function that utilizes the Pythagorean equation $a^2+b^2=c^2$ to determine whether a given set of three side lengths can form a Pythagorean triplet.

- 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**
  - True if the given set of three side lengths satisfies the Pythagorean equation
  - False otherwise.
  - Please return bool type True or False, not 'True' or 'False'
- 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)

Typically, in web service registration, you cannot create a new account with an existing username. Let's implement a function to check if the desired username is already in use.

➔ The first parameter is a list of existing usernames.
➔ The second parameter is the desired username that I want to create

- The function name: **function3**
- This function takes two positional arguments
  - First argument: **list type (that include at least one string)**
  - Second argument: **str type (its length is greater than 0)**
- The return value: **bool type**
  - If the second argument is present in the first argument (a list) regardless of case sensitivity, the function should return True.
  - If the second argument is not present in the first argument, the function should return False, considering case insensitivity.
  - Please return bool type True or False, not 'True' or 'False'
- 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)

During the registration process, if a user creates a password that is too simple, the probability of being hacked increases. Let's create a function that checks whether a user's password includes uppercase letters, lowercase letters, and numbers.

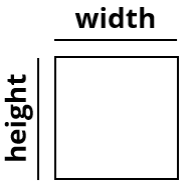
- The function name: **function4**
- This function takes a positional argument: **str type**
  - A length of the argument is greater than 2
  - The argument consists of alphabets or numbers
- The return value: **bool type**
  - Return True if the argument contains all of the following: uppercase letters, lowercase letters, and numeric characters.
  - Return False if any of these is missing in the argument
  - Please return bool type True or False, not 'True' or 'False'
- This function will be tested with positional arguments, for example:
  - print(function4("abc123ABC")) # True
  - print(function4("1234LOVEyou")) # True
  - print(function4("a1b2c3d4")) # False (because there is no uppercase)
  - print(function4("HelloWorld")) # False (because there is no number)
  - print(function4("1234")) # False (because there is no alphabet)

function5 (4 points)

This question asks about distinguishing between positional arguments and keyword arguments and inquiring about their appropriate utilization.

- 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**
  - Return the sum of all keyword arguments whose keywords start with 'a'.
  - If there are no keyword arguments starting with 'a', return 0.
- 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].

- This class has been structured to represent a rectangle with *height* and *width*.
- The methods **__init__**, **get_area**, and **is_square** are incomplete. Please complete these three methods.
- Once the class is properly implemented, running the provided [Code] should produce the output as shown in [Result].
- You only need to create Rectangle class in your module. Remove [Code] after testing.

```
class Rectangle:
    def __init__():

    def get_area() -> int:
        """return the area of this rectangle (int type)"""
        return
    def is_square() -> bool:
        """Return True if this rectangle is a square,
           otherwise return False (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].

- **The __init__ method is already completed, so please do not modify it.**
- The **__add__** method is incomplete. Complete this method to behave as follows:
  - When performing **Num** type + **Num** type, return the sum of the num variables of the Num instances.
  - When performing **Num** type + **int** type, return the sum of the num variable of the **Num** instance and the **int** value.
- Once the class is properly implemented, running the provided [Code] should produce the output as shown in [Result].
- You only need to create the Rectangle class in your module. Remove [Code] after testing.

```
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].

- **The __init__ method is already completed, so please do not modify it.**
- Complete the **get_anonymized_userid** method based on the comments provided

```
class User:
    def __init__(self, userid: str):
        self.userid = userid
    def get_anonymized_userid
        """If a length of the given argument's userid is 2 or less,
             return '*' repeated for that length (str type).
           If the length is 3 or more,
             replace all characters in the middle of the userid variable
             (excluding the first and last characters) with '*'
             and return the modified string (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].

- **The __init__ method is already completed, so please do not modify it.**
- The **__add__** and **get_sum** methods are incomplete and need to be completed:
  - **__add__**: This takes two arguments and returns the sum of their value variables.
  - **get_sum**: This takes two arguments and returns the sum of the two values.
- Once you complete the two methods successfully, running the provided [Code] should produce results similar to those shown in [Result].

```
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].

- Complete the **Student** class so that running the provided [Code] produces results as shown in [Result].
    - [Hint] The **Student** class has a class variable *count*.
    - [Hint] Each time an instance is created, the value of the *count* class variable in the **Student** class increases by 1.

```
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
```

멈추거나 포기하지만 않는다면, 당장 느리게 가는 것 쯤은 문제가 아냐!

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

THE END