2023. Spring

# Web/Python Programming 01
# Final-term exam

Professor: Sangkeun Park

- Name your Python file, _StudentNo.py (for example, **_2023123456.py**), otherwise -1P.
    - The uploaded file name will be changed, such as _2023123456-1.py, _2023123456-2.py… if you upload more than once. Never mine. It is OK if you uploaded a correctly named file.
- Remove or comment all **print** functions, otherwise -1P.
- Do not use the **input** function, otherwise -1P.
- If I fail to import your submission by an error in the code, you get -1P.
- After submitting the file, close your laptop. Then, you can study for another exam using paper-based materials. Please let the professor or TAs know you're done. You cannot use digital devices in the class anymore.

**Define the following 5 functions and 5 classes on your file, so that I can import and use it.**

### function1 [4 points]

**Collatz conjecture** is whether repeating two simple arithmetic operations will eventually transform every positive integer into 1.

1. If the given positive integer n is even, n/2.
2. If the given positive integer n is odd, 3n+ 1.

Then, If n becomes 1, stop. Otherwise, repeat one of the two operations.

- The function name: **function1**
- This function takes one argument: **int** type (only positive numbers greater than 1)
- The return value: **float** type
    - Sum of the values calculated by the two arithmetic operations.
    - If 6 is given, 3→10→5→16→8→4→2→1, total 8 values are calculated, so the sum of the values is 49**.**
- For example,
    - print(function1(6)) **# 49.0**
        - because 3+10+5+16+8+4+2+1 is 49
    - print(function1(4)) **# 3.0**
        - because 2+1 is 3
    - print(function1(5)) **# 31.0**
        - because 16+8+4+2+1 is 31
    - print(function1(64)) **# 63.0**
        - because 32+16+8+4+2+1 is 63

**function2 [4 points]**

| str Method | Description |
|---|---|
| index(value) | - Returns the index of the first occurrence of the value.<br>- Raises an exception if the value is not found. |
| find(value) | - Returns the index of the first occurrence of the value.<br>- Returns -1 if the value is not found. |

- The function name: **function2**
- This function takes the unknown number of positional arguments
  - All arguments are **str** types.
    → each argument can include only one "@"
    → each argument can include only one "."
- The return value: **int** type
  - The number of valid email-format values.
    (We assume that if a value includes the two characters, "@" and ".",
    and "@" precedes ".", it is a valid email-format value.
    Therefore, "skpark@abc.com" is valid, but "sk.park@khu" is not valid.)
- For example,
  - print(function2("admin")) **# 0**
    - because there is no valid email-format values
  - print(function2("admin@abc.com", "customer@abc.com")) **# 2**
    - because the two arguments are valid email-format values
  - print(function2("skpark.khu@com", "skpark@khucom")) **# 0**
    - because there is no valid email-format values
  - print(function2("python@python.org", "sk.park@abccom", "guido@python")) **# 1**
    - because only first argument is a valid email-format value
  - print(function2()) **# 0**
    - because no argument is given.

**function3 [4 points]**

- The function name: **function3**
- This function takes two arguments
  - The first argument: **list** type (all elements are **str**, “+” or “-”, meaning that a sign of number)
  - The second argument: **list** type (all elements are **int**)
    → The two arguments have the same length.
    → Each argument includes at least one element / at most 10 elements.
- The return value: **int** type
  - Sum of the integers that are generated by combining the each element from the two arguments in order
- For example,
  - print(function3(["+", "-", "+"], [1, 2, 3])) **# 2**
    - because (+1) + (-2) + (+3) is 2.
  - print(function3(["-"], [1])) **# -1**
    - because (-1) is -1.
  - print(function3(["+", "+", "-", "+"], [1, 2, 3, 4])) **# 4**
    - because (+1) + (+2) + (-3) + (+4) is 4.
  - print(function3(["-", "-", "-"], [1, 2, 3])) **# -6**
    - because (-1) + (-2) + (-3) is -6.

**function4 [4 points]**

In mathematics, **a Harshad number** is an integer that is divisible by the sum of its digits.
For example,
- 21 is divisible by 3 (2+1), so it is a Harshad number.
- 192 is divisible by 12 (1+9+2), so it is a Harshad number.
- 123 is not divisible by 6 (1+2+3), so it is not a Harshad number.

- The function name: **function4**
- This function takes one argument: **int** type (only positive numbers, greater than 1)
- The return value: **bool** type
    - **True** if the given argument is a harshad number
    - **False** if the given argument is not a harshad number
- For example,
    - print(function4(21)) **# True**
        - because 21 is divisible by 3 (2+1)
    - print(function4(192)) **# True**
        - because 192 is divisible by 12 (1+9+2)
    - print(function4(123)) **# False**
        - because 123 is not divisible by 6 (1+2+3)
    - print(function4(2024)) **# True**
        - because 2024 is divisible by 8 (2+0+2+4)

**function5 [4 points]**

- The function name: **function5**
- This function takes two arguments
    - The first argument: **int** type
    - The second argument: **int** type or **str** type
- The return value: **float** type or **str** type
    - Return the divided value (divide the first argument by the second argument)
    - If ZeroDivisionError is raised, returns "ZeroDivisionError"
    - If the other errors are raised (except for ZeroDivisionError), returns "Error"
- For example,
    - print(function5(1, "10")) **# "Error"**
    - print(function5(1, 10)) **# 0.1**
    - print(function5(0, 0)) **# "ZeroDivisionError"**
    - print(function5(99, 99)) **# 1.0**
    - print(function5(10, [1, 2, 3])) **# "Error"**

**class Anonymizer [5 points]**

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

```
class Anonymizer:
    def __init__(self, name):
        """Initializer"""

    def get_anonymized_name(self, n):
        """return an anonymous name.
           replace first n characters of the name with *.
           n is a positive number.
           If n is greater than the length of the name,
               n becomes the length of the name.
        """
```

[Code] → I will test your class using the following code with various int arguments

```
a1 = Anonymizer("Sangkeun")
print(a1.name)
print(a1.get_anonymized_name(1))
print(a1.get_anonymized_name(2))
print(a1.get_anonymized_name(10))
print(a1.name)

a2 = Anonymizer("Kyungheeuniv")
print(a2.name)
print(a2.get_anonymized_name(3))
print(a2.get_anonymized_name(8))
print(a2.get_anonymized_name(20))
print(a2.name)
```

[Result]

```
Sangkeun
*angkeun
**ngkeun
********
Sangkeun
Kyungheeuniv
***ngheeuniv
********univ
**********
Kyungheeuniv
```

**class Account [5 points]**

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

- **__init__** method is already provided.
- Complete **deposit**, **withdraw**, **transfer**, and **__eq__** methods as described.

```
class Account:
    def __init__(self, balance):
       """Initializer: balance can be negative."""
        self.balance = balance

    def deposit
        """Deposit the given money to self's balance."""

    def withdraw
        """Withdraw the given money from self's balance."""

    def transfer
        """Transfer the given money from self's balance to other's account."""

    def __eq__
        """Return True if the given two arguments' balances are equal,
           False otherwise."""
```

[Code] → I will test your class using the following code with various int values.

```
a1, a2 = Account(0), Account(0)
print(a1.balance, a2.balance, a1==a2)
a1.deposit(100)
a2.withdraw(50)
print(a1.balance, a2.balance, a1==a2)
a1.transfer(50, a2)
a2.deposit(50)
print(a1.balance, a2.balance, a1==a2)
a2.transfer(25, a1)
print(a1.balance, a2.balance, a1==a2)
```

[Result]

```
0 0 True
100 -50 False
50 50 True
75 25 False
```

## class NonStrNameError [5 points]

Please create a custom Error, **NonStrNameError**, so that the following code works

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

```
try:
    raise NonStrNameError
except NonStrNameError as e:
    print(e)
```

[Result]

```
NonStrNameError
```

## class Integer [5 points]

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

- **__init__** method is already provided.
- Complete appropriate **magic methods** for **+** and **-**, as described.

```
class Integer:
    def __init__(self, number):
        """Initializer: assume that the number is always int value"""
        self.number = number

    def __add__(self, other):
        """Return a new instance of Integer
           which number attributes is self.number+other.number.
        """

    def __sub__(self, other):
        """Return a new instance of Integer
           which number attributes is self.number-other.number.
        """
```

[Code] → I will test your class using the following code with only int arguments for the initializer.

```
n1 = Integer(2)
n2 = Integer(3)
n3 = n1+n2
n4 = n2-n1
print(n1.number, n2.number, n3.number, n4.number)
print(type(n1)==type(n4))
```

[Result]

```
2 3 5 1
True
```

**class University [5 points]**

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

- **__init__** method and **get_percentail** method are already provided.
- Complete the **set_maximum_score** method, as described below.
- You must not change the **KyungheeUniversity** class.
    - You do not necessarily implement **KyungheeUniversity class** on your file when you submit. (It does not matter whether you can paste on it or not, but do not change the class.)

```
class University:
    maximum_score = 4.5
    def __init__(self, score):
        """Initializer:
           assume that score is float type, 0.0 <= score <= 4.5."""
        self.score=score

    def get_percentile(self):
        """get a percentile of self's score based on the maximum_score"""
        return f"{int(self.score/self.maximum_score*100)}%"

    def set_maximum_score
        """set the class variable, maximum_score, as the given value"""

class KyungheeUniversity(University):
    def __init__(self, score):
        super().__init__(score)
```

[Code] → I will test your class using the following code with only float arguments for the initializer.

```
student1 = University(4.0)
student2 = KyungheeUniversity(4.0)
print(student1.get_percentile())
print(student2.get_percentile())
KyungheeUniversity.set_maximum_score(4.3)
print(student1.get_percentile())
print(student2.get_percentile())
University.set_maximum_score(4.0)
print(student1.get_percentile())
print(student2.get_percentile())
```

[Result]

```
88%
88%
88%
93%
100%
93%
```

2023. 1학기

# 웹파이썬프로그래밍 01
# 기말고사 시험

담당교수 박상근

- 파일 이름은 _StudentNo.py (예, **_2023123456.py**) 형태로 저장하세요, otherwise -1P.
    - 여러번 업로드하면 시스템 상에서 파일 이름이 _2023123456-1.py, _2023123456-2.py 와 같이 뒤에 숫자가 증가하며 붙게됩니다. 본인이 정상적인 파일 이름으로 업로드했다면 시스템상에서 자동으로 파일 이름이 바뀌는 것은 전혀 신경쓸 필요없습니다.
- **print** 함수는 모두 주석처리하거나 삭제하세요, otherwise -1P.
- **input** 함수 사용하지 마세요, otherwise -1P.
- 제가 여러분의 제출 파일을 import 하는데 에러가 발생한다면, you get -1P.
- 최종 제출 후에 더 이상 시험을 치를 필요가 없다고 생각되면 조용히 노트북을 덮으세요. 시험이 끝날 때 까지 전자기기는 사용 불가입니다. 대신 조교나 교수에게 확인 후 종이로 된 다른 시험 자료를 볼 수 있습니다. 시험이 먼저 끝났다고 시험장 밖으로 나갈 수 없습니다.

**Define the following 5 functions and 5 classes on your file, so that I can import and use it.**

### function1 [4 points]

콜라츠 추측(Collatz conjecture)이란 임의의 자연수가 다음 두 조작을 거쳐 항상 1이 된다는 추측입니다.
1. 짝수라면 2로 나눈다.
2. 홀수라면 3을 곱하고 1을 더한다.

이제 이 수가 1이면 조작을 멈추고, 1이 아니면 첫 번째 단계로 다시 돌아가서 반복한다.

- The function name: **function1**
- This function takes one argument: **int** type (only positive numbers greater than 1)
- The return value: **float** type
    - 두 가지 조작을 반복하는 동안 계산되는 모든 숫자의 합을 리턴
    - 예, 6이 인자로 주어진다면, 차례로 3, 10, 5, 16, 8, 4, 2, 1 로, 총 8번의 계산이 필요하므로, 이 8개의 숫자를 모두 더하면 49.
- For example,
    - print(function1(6)) **# 49.0**
        - because 3+10+5+16+8+4+2+1 is 49
    - print(function1(4)) **# 3.0**
        - because 2+1 is 3
    - print(function1(5)) **# 31.0**
        - because 16+8+4+2+1 is 31
    - print(function1(64)) **# 63.0**
        - because 32+16+8+4+2+1 is 63

**function2 [4 points]**

| str Method | Description |
|---|---|
| index(value) | - Returns the index of the first occurrence of the value.<br>- Raises an exception if the value is not found. |
| find(value) | - Returns the index of the first occurrence of the value.<br>- Returns -1 if the value is not found. |

- The function name: **function2**
- 이 함수는 몇 개의 인자를 받을지 정확하게 정해져 있지 않음
  - 몇 개의 인자를 받더라도 모든 인자는 **str** 타입
    → 각 인자는 최대 하나의 "@" 문자를 포함할 수 있음
    → 각 인자는 최대 하나의 "." 문자를 포함할 수 있음
- The return value: **int** type
  - 인자 중 유효한 이메일 포맷의 인자 개수를 리턴
    (유효한 이메일 포맷이란, "@" 문자와 "." 문자를 모두 포함하면서, "@" 문자가 "." 문자보다 먼저 등장하는 경우를 말한다. 그러므로, "skpark@abc.com" 는 유효한 이메일 포맷이지만 "sk.park@khu" 는 유효한 이메일 포맷이 아니다..)
- For example,
  - print(function2("admin")) **# 0**
    - because there is no valid email-format values
  - print(function2("admin@abc.com", "customer@abc.com")) **# 2**
    - because the two arguments are valid email-format values
  - print(function2("skpark.khu@com", "skpark@khucom")) **# 0**
    - because there is no valid email-format values
  - print(function2("python@python.org", "sk.park@abccom", "guido@python")) **# 1**
    - because only first argument is a valid email-format value
  - print(function2()) **# 0**
    - because no argument is given.

**function3 [4 points]**

- The function name: **function3**
- This function takes two arguments
  - The first argument: **list** type (모든 요소는 "+" 또는 "-" 인 **str** 타입 값으로 구성: 부호를 의미)
  - The second argument: **list** type (모든 요소는 **int** 타입의 숫자값으로 구성)
    → 두 인자의 길이는 무조건 같습니다.
    → 각 인자는 최소 하나의 요소를 포함하며, 최대 10개까지의 요소를 포함할 수 있습니다.
- The return value: **int** type
  - 두 인자에서 순서대로 하나씩 값을 꺼내 부호와 숫자로 짝을 맞춰 만든 숫자들의 합
- For example,
  - print(function3(["+", "-", "+"], [1, 2, 3])) **# 2**
    - because (+1) + (-2) + (+3) is 2.
  - print(function3(["-"], [1])) **# -1**
    - because (-1) is -1.
  - print(function3(["+", "+", "-", "+"], [1, 2, 3, 4])) **# 4**
    - because (+1) + (+2) + (-3) + (+4) is 4.
  - print(function3(["-", "-", "-"], [1, 2, 3])) **# -6**
    - because (-1) + (-2) + (-3) is -6.

**function4 [4 points]**

하샤드의 수**(a Harshad number)** 란, 해당 숫자의 각 자리 숫자를 모두 합친 수로 본래의 수를 나눌 때, 정확하게 나누어 떨어지는 숫자를 말한다.
예를 들어,
- 21 은 3 (2+1) 으로 나누어 떨어지므로 있으므로 하샤드의 수이다.
- 192 은 12 (1+9+2) 로 나누어 떨어지므로 하샤드의 수이다.
- 123 은 6 (1+2+3) 으로 나누어 떨어지지 않으므로 하샤드의 수가 아니다.

- The function name: **function4**
- This function takes one argument: **int** type (무조건 2 이상의 양수임)
- The return value: **bool** type
    - 해당 인자가 하샤드의 수이면 **True** 리턴
    - 해당 인자가 하샤드의 수가 아니면 **False** 리턴
- For example,
    - print(function4(21)) **# True**
        - because 21 is divisible by 3 (2+1)
    - print(function4(192)) **# True**
        - because 192 is divisible by 12 (1+9+2)
    - print(function4(123)) **# False**
        - because 123 is not divisible by 6 (1+2+3)
    - print(function4(2024)) **# True**
        - because 2024 is divisible by 8 (2+0+2+4)

**function5 [4 points]**

- The function name: **function5**
- This function takes two arguments
    - The first argument: **int** type
    - The second argument: **int** type or **str** type
- The return value: **float** type or **str** type
    - 첫 번째 인자에서 두 번째 인자를 나눈 값을 리턴
    - ZeroDivisionError 가 발생하면, "ZeroDivisionError" 리턴
    - 그 외의 다른 에러가 발생하면 "Error" 리턴
- For example,
    - print(function5(1, "10")) **# "Error"**
    - print(function5(1, 10)) **# 0.1**
    - print(function5(0, 0)) **# "ZeroDivisionError"**
    - print(function5(99, 99)) **# 1.0**
    - print(function5(10, [1, 2, 3])) **# "Error"**

**class Anonymizer [5 points]**

아래에 준비된 템플릿을 활용해서 **Anonymizer** 클래스를 완성하세요. 이 클래스를 완성시켰다는 가정하에, [Code]를 실행하면 [Result] 대로 결과가 출력되어야 합니다.

```
class Anonymizer:
    def __init__(self, name):
        """Initializer"""

    def get_anonymized_name(self, n):
        """return an anonymous name.
           replace first n characters of the name with *.
           n is a positive number.
           If n is greater than the length of the name,
               n becomes the length of the name.
        """
```

[Code] → I will test your class using the following code with various int arguments

```
a1 = Anonymizer("Sangkeun")
print(a1.name)
print(a1.get_anonymized_name(1))
print(a1.get_anonymized_name(2))
print(a1.get_anonymized_name(10))
print(a1.name)

a2 = Anonymizer("Kyungheeuniv")
print(a2.name)
print(a2.get_anonymized_name(3))
print(a2.get_anonymized_name(8))
print(a2.get_anonymized_name(20))
print(a2.name)
```

[Result]

```
Sangkeun
*angkeun
**ngkeun
********
Sangkeun
Kyungheeuniv
***ngheeuniv
********univ
************
Kyungheeuniv
```

**class Account [5 points]**

아래에 준비된 템플릿을 활용해서 **Account** 클래스를 완성하세요. 이 클래스를 완성시켰다는 가정하에, [Code]를 실행하면 [Result] 대로 결과가 출력되어야 합니다.

- **__init__** 메소드는 이미 완성되어 있습니다.
- 나머지 **deposit**, **withdraw**, **transfer**, **__eq__** 메소드를 완성시키세요.

```
class Account:
    def __init__(self, balance):
       """Initializer: balance can be negative."""
        self.balance = balance

    def deposit
        """Deposit the given money to self's balance."""

    def withdraw
        """Withdraw the given money from self's balance."""

    def transfer
        """Transfer the given money from self's balance to other's account."""

    def __eq__
        """Return True if the given two arguments' balances are equal,
           False otherwise."""
```

[Code] → I will test your class using the following code with various int values.

```
a1, a2 = Account(0), Account(0)
print(a1.balance, a2.balance, a1==a2)
a1.deposit(100)
a2.withdraw(50)
print(a1.balance, a2.balance, a1==a2)
a1.transfer(50, a2)
a2.deposit(50)
print(a1.balance, a2.balance, a1==a2)
a2.transfer(25, a1)
print(a1.balance, a2.balance, a1==a2)
```

[Result]

```
0 0 True
100 -50 False
50 50 True
75 25 False
```

## class NonStrNameError [5 points]

**NonStrNameError** 에러를 직접 만드세요. 이 에러를 완성시켰다는 가정하에, [Code]를 실행하면 [Result] 대로 결과가 출력되어야 합니다.

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

```
try:
    raise NonStrNameError
except NonStrNameError as e:
    print(e)
```

[Result]

```
NonStrNameError
```

## class Integer [5 points]

아래에 준비된 템플릿을 활용해서 **Integer** 클래스를 완성하세요. 이 클래스를 완성시켰다는 가정하에, [Code]를 실행하면 [Result] 대로 결과가 출력되어야 합니다.

- **__init__** 메소드는 이미 완성되어 있습니다.
- **__add__**, **__sub__** 매직 메소드를 완성시키세요.

```
class Integer:
    def __init__(self, number):
        """Initializer: assume that the number is always int value"""
        self.number = number

    def __add__(self, other):
        """Return a new instance of Integer.
           Its number attribute is self.number+other.number.
        """

    def __sub__(self, other):
        """Return a new instance of Integer.
           Its number attribute is self.number-other.number.
        """
```

[Code] → I will test your class using the following code with only int arguments for the initializer.

```
n1 = Integer(2)
n2 = Integer(3)
n3 = n1+n2
n4 = n2-n1
print(n1.number, n2.number, n3.number, n4.number)
print(type(n1)==type(n4))
```

[Result]

```
2 3 5 1
True
```

**class University [5 points]**

아래에 준비된 템플릿을 활용해서 **University** 클래스를 완성하세요. 이 클래스를 완성시켰다는 가정하에, [Code]를 실행하면 [Result] 대로 결과가 출력되어야 합니다.

- **__init__** 메소드와 **get_percentile** 메소드는 이미 완성되어 있습니다.
- **set_maximum_score** 메소드만 잘 완성시키세요
- **KyungheeUniversity** 클래스는 이미 완성된 상태이니 절대 수정하면 안 됩니다.
    - **KyungheeUniversity** 클래스는 여러분이 제출 할 때 코드에 남겨놔도 되고 지워도 됩니다.

```
class University:
    maximum_score = 4.5
    def __init__(self, score):
        """Initializer:
           assume that score is float type, 0.0 <= score <= 4.5."""
        self.score=score

    def get_percentile(self):
        """get a percentile of self's score based on the maximum_score"""
        return f"{int(self.score/self.maximum_score*100)}%"

    def set_maximum_score
        """set the class variable, maximum_score, as the given value"""

class KyungheeUniversity(University):
    def __init__(self, score):
        super().__init__(score)
```

[Code] → I will test your class using the following code with only float arguments for the initializer.

```
student1 = University(4.0)
student2 = KyungheeUniversity(4.0)
print(student1.get_percentile())
print(student2.get_percentile())
KyungheeUniversity.set_maximum_score(4.3)
print(student1.get_percentile())
print(student2.get_percentile())
University.set_maximum_score(4.0)
print(student1.get_percentile())
print(student2.get_percentile())
```

[Result]

```
88%
88%
88%
93%
100%
93%
```