Report

제출일 2019년 4월 12일
과목명 오픈소스SW활용
담당교수 황 두 성 교 수 님
전공 공대 소프트웨어학과
학번 32144548
이름 조 창 연

Problem 1

Write a Python function, is_multiple(n,m), that takes two integer values and returns True if n is a multiple of m, that is, $n = mi$ for some integer i, and False otherwise.

<Source Code>

```
def is_multiple(m,n):
    return True if m % n == 0 else False

print(is_multiple(10,3))
print(is_multiple(10,2))
```

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw$ vim p1.py
ubuntu@ubuntu-VirtualBox:~/ossw$ python3 p1.py
False
True
ubuntu@ubuntu-VirtualBox:~/ossw$
```

Problem 2

Write a Python function, is_even(k), that takes an integer value and returns True if k is even, and False otherwise. However, your function cannot use the multiplication, modulo, or division operators.

<Source Code>

```
def is_even(k):
    return False if k & 1 else True

print(is_even(2))
print(is_even(3))
```

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw$ vim p2.py
ubuntu@ubuntu-VirtualBox:~/ossw$ python3 p2.py
True
False
ubuntu@ubuntu-VirtualBox:~/ossw$
```

Problem 3

Python allows negative integers to be used as indices into a sequence, such as string. If string s has length n, and expression $s[k]$ is used for index $-n \leq k \leq 0$, what is the equivalent index $j \geq$ such that $s[j]$ references the same element?

<Source Code>

```
s = "pythonstring"
n = len(s)

for k in range(-n, 0):
    print(s[k])

for j in range(0,n):
    print(s[j])
```

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw$ vim p3.py
ubuntu@ubuntu-VirtualBox:~/ossw$ python3 p3.py
p
y
t
h
o
n
s
t
r
i
n
g
p
y
t
h
o
n
s
t
r
i
n
g
ubuntu@ubuntu-VirtualBox:~/ossw$
```

Problem 4

Write a program that takes two arrays(list) a and b of length n storing int values, return the dot product of a and b. That is, it returns an array(list) c of length n such that $c[i] = a[i] \cdot b[i]$, for i = 0, ... , $n-1$.

<Source Code>

```
first = [1, 2, 3, 4]
second = [1, 2, 3, 4]

def dot_product(a,b):
    c = []
    for k in range(0, len(a)):
        c.append(a[k] + b[k])
    return c

third = dot_product(first, second)

print(third)
```

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw$ vim p4.py
ubuntu@ubuntu-VirtualBox:~/ossw$ python3 p4.py
[2, 4, 6, 8]
ubuntu@ubuntu-VirtualBox:~/ossw$
```

Problem 5

Explain the difference between mutable objects and immutable objects and give some examples.

Python에서 mutable은 변경 가능한 객체를 의미하고, immutable은 변경 불가능한 객체를 의미합니다. 변경 가능한 객체인 mutable은 call by reference의 속성을 가지고 있으며, 리스트(list)와 딕셔너리(dict)가 있습니다I 변경 불가능한 객체인 immutable은 call by value의 속성을 가지고 있으며, 일반적인 자료형(int, stirng 등)과 튜플(tuple) 등이 있습니다. 즉, mutable 객체들은 값의 변경이 일어날 때 주소가 참조하는 값이 모두 변경되는 것을 알 수 있고, immutable 객체들은 값이 변경되는 것이 새로운 객체로 생성이 되기 때문에 변경이 일어날 때 기존 객체는 변하지 않는다는 것을 알 수 있습니다.

Problem 6

Write a function called convert_pound that converts pounds to kilograms. One pound is equal to 0.454 kilograms.

<Source Code>

```
def convert_pound(vindication, smite, impetuosity):
    try:
        efface = float(input(vindication))
    except ValueError:
        print('Enter a number, please!')
    else:
        efface = round(efface * smite, 1)
        print(efface, end='')
        print(impetuosity)

convert_pound('pound : ', 0.454,' [kg]')
```

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw$ vim p6.py
ubuntu@ubuntu-VirtualBox:~/ossw$ python3 p6.py
pound : 1
0.5 [kg]
ubuntu@ubuntu-VirtualBox:~/ossw$ python3 p6.py
pound : 2
0.9 [kg]
ubuntu@ubuntu-VirtualBox:~/ossw$ python3 p6.py
pound : 3
1.4 [kg]
ubuntu@ubuntu-VirtualBox:~/ossw$ python3 p6.py
pound : 4
1.8 [kg]
ubuntu@ubuntu-VirtualBox:~/ossw$
```

Problem 7

Explain why we use a module in software development and how to structure a module in Python.

소프트웨어 개발자들이 프로그램을 개발하다 보면 프로그램의 크기가 커짐에 따라 개발이나 유지보수를 위해 소스 파일을 여러 개로 나눠야 할 경우가 생깁니다. 또한, 한 번 정의해서 사용해야 편리한 함수를 다른 프로그램에선 정의하지 않고 곧바로 사용하고 싶어질 수도 있습니다. 이러한 불편함 때문에 Python에서 한 번 정의했던 변수나 함수, 클래스 등을 다른 Python program에서도 손쉽게 불러와 사용할 수 있도록 하나의 파일로 모아놓는 방법을 제공하는 것을 우리는 모듈이라고 합니다.

모듈은 Python program에서 사용할 수 있는 여러 정의들과 실행 가능한 구문들을 담고 있는 하나의 Python file로써, Python에서 기본적으로 제공하는 모듈뿐만 아니라 다른 사람이 만든 모듈을 사용하거나 자신이 직접 새로운 모듈을 작성하여 사용할 수도 있습니다.

Python에서 import문을 사용하여 아래와 같이 3가지 방식으로 모듈을 불러올 수 있습니다.

1. import 모듈명
2. from 모듈명 import *
3. from 모듈명 import 함수명 또는 클래스명

이러한 import문은 코드의 어느 위치에서라도 사용할 수 있지만, 통상적으로 코드의 맨 앞에 위치하는 것이 가장 좋습니다.

위 방식 중 첫 번째와 두 번째 방식은 해당 모듈 전체를 불러올 수 있습니다.
첫 번째 방식은 모듈에 포함된 변수나 함수를 사용할 때마다 매번 모듈명과 함께 닷(.) 연산자를 사용해야만 합니다. 만약 모듈명을 함께 사용하지 않으면, 파이썬은 해당 변수나 함수가 정의되어 있지 않다고 판단하여 NameError를 발생시킬 것입니다. 하지만, 두 번째 방식은 변수명이나 함수명만으로도 간편하게 사용할 수 있습니다. 마지막으로 세 번째 방식은 모듈 전체가 아닌 해당 모듈 내에서 특정 변수나 함수, 클래스만을 불러오고자 할 때 사용합니다.

Problem 8

Consider the following statement, which creates a list of populations of countries in eastern Asia(China, DPR Korea, Hong Kong, Mongolia, Republic of Korea, Japan and Taiwan), in millions: country populations = [1500, 33, 7, 45, 49, 101, 21]. Write a function to compute the population average of those countries.

<Source Code>

```
def population_average(dic):
    s = 0
    c = 0
    for x in dic:
        s += int(dic[x])
    return int(s/len(dic))
```

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw$ vim p8.py
ubuntu@ubuntu-VirtualBox:~/ossw$ python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from p8 import *
>>> population = { 'China':1500, 'DPR Korea':33, 'Hong Kong':7, 'Mongolia':45, 'Republic of Korea':49, 'Japan':101, 'Taiwan':21 }
>>> population_average(population)
250
>>>
```

Problem 9

Suppose the file alkaline_metals.txt contains this:

4 9.012

12 24.305

20 20.078

38 87.62

56 137.327

88 226

Write a for loop to read the contents of alkaline_metals.txt, and store it in a nested list with each element of the list contains the atomic number and atomic weight for an element(Hint:use string:split()).

<Source Code>

```
f1 = open("alkaline_metals.txt", "r")

atomic_numbers = []
atomic_weights = []

for line in f1:
    atomic_number, atomic_weight = line.split()
    atomic_numbers.append(atomic_numer)
    atomic_weights.append(atomic_weight)

print("atomic_number", end='')
print(atomic_numbers)

print("atomic_weight", end='')
print(atomic_weight)
```

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw$ cat alkaline_metals.txt
4  9.012
12 24.305
20 20.078
38 87.62
56 137.327
88 226
ubuntu@ubuntu-VirtualBox:~/ossw$ python3 p9.py
atomic_number : ['4', '12', '20', '38', '56', '88']
atomic_weight : ['9.012', '24.305', '20.078', '87.62', '137.327', '226']
ubuntu@ubuntu-VirtualBox:~/ossw$
```

Problem 10

Design and implement a program that

step 1 asks the user for a temperature in Fahrenheit degrees and reads the number

step 2 computes the correspodning temperature in Celsius degrees

step 3 prints out the temperature in the Celsius scale.

<Source Code>

```
# step 1. askes the user for a temperature in Fahrenheit degrees and reads the number
number = float(input("please enter Fahrenheit temperature : "))

# step 2. computes the correspodning temperature in Celsius degrees
number = (number - 32.0) * 5.0 / 9.0

# step 3. prints out the temperature in the Celsius scale.
print("Celsius temperature is %f"%(number))
```

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw$ python3 p10.py
please enter Fahrenheit temperature : 110
Celsius temperature is 43.333333
ubuntu@ubuntu-VirtualBox:~/ossw$
```

Problem 11

Modify your solution of Problem 10 such that the Fahrenheit temperature is read from the command line.

<Source Code>

```
import sys

# step 1. askes the user for a temperature in Fahrenheit degrees and reads the number
number = float(sys.argv[1])

# step 2. computes the correspodning temperature in Celsius degrees
number = (number - 32.0) * 5.0 / 9.0

# step 3. prints out the temperature in the Celsius scale.
print("Celsius temperature is %f"%(number))
```

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw$ python3 p11.py 110
Celsius temperature is 43.333333
ubuntu@ubuntu-VirtualBox:~/ossw$
```

Problem 12

Some object is moving along a path in the plane. At n points of time we have recorded the corresponding (x, y) positions of the object: $(x_0, y_0), (x_1, y_2), \ldots, (x_{n-1}, y_{n-1})$. The total length L of the path from (x_0, y_0) to (x_{n-1}, y_{n-1}) is the sum of all the individual line segments ((x_{i-1}, y_{i-1}) to $(x_i, y_i), i = 1, \ldots, n-1$):

$$L = \sum_{i=1}^{n-1} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$$

Make a function pathlength(x, y) for computing L according to the formula. The arguments x and y hold all the $x_0, \ldots, x_{n-1}$ and $y_0, \cdots, y_{n-1}$ coordinates, respectively. Test the function on a triangular path with the four points (1, 1), (2, 1), (1, 2), and (1, 1).

<Source Code>

```
import math

def pathlength(x,y):
        L = 0
        for k in range(len(x) - 1):
            L += math.sqrt((x[k + 1] - x[k])**2 + (y[k + 1] - y[k])**2)
        return L

if __name__ == "__main__":
    x = [1,2,1,1,];y=[1,1,2,1]
    print(pathlength(x,y))
```

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw$ vim p12.py
ubuntu@ubuntu-VirtualBox:~/ossw$ python3 p12.py
3.414213562373095
ubuntu@ubuntu-VirtualBox:~/ossw$
```

Problem 13

The value of π equals the circumference of a circle with radius 1/2. Suppose we approximate the circumference by a polygon through $N+1$ points on the circle. The length of this polygon can be found using the pathlength function from the above exercise. Compute $N+1$ points (x_i, y_i) along a circle with radius 1/2 according to the formulas

$$x_i = \frac{1}{2}cos(2\pi i/N), \;\; y_i = \frac{1}{2}sin(2\pi i/N), \;\; i = 0, \;\ldots\;, \;N.$$

Call the pathlength function and write out the error in the approximation of π for $N = 2k$, k = 2, 3, … , 10.

<Source Code>

```
from p12 import *
import math

x = []
y = []

for k in range(2, 11):
    N = 2 * k
    for i in range(N + 1):
        x.append(1/2 * math.cos(2 * math.pi * i/N))
        y.append(1/2 * math.sin(2 * math.pi * i/N))
    print("N : %d | Pathlength : %d "%(N,pathlength(x,y)))
    x.clear()
    y.clear()
```

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw$ python3 p13.py
N : 4 | Pathlength : 2
N : 6 | Pathlength : 3
N : 8 | Pathlength : 3
N : 10 | Pathlength : 3
N : 12 | Pathlength : 3
N : 14 | Pathlength : 3
N : 16 | Pathlength : 3
N : 18 | Pathlength : 3
N : 20 | Pathlength : 3
ubuntu@ubuntu-VirtualBox:~/ossw$
```

Problem 14

Make six conversion functions between temperatures in Celsius, Kelvin, and Fahrenheit:

C2F(degree)

F2C(degree)

C2K(degree)

K2C(degree)

F2K(degree)

K2F(degree)

Collect these functions in a module convert_temp. Make some sample calls to the functions from an interactive Python shell.

<Source Code>

```
def C2F(degree):
    return degree * 1.8 + 32.0

def F2C(degree):
    return (degree - 32.0)/1.8

def C2K(degree):
    return degree + 273.15

def K2C(degree):
    return degree - 273.15

def F2K(degree):
    return C2K(F2C(degree))

def K2F(degree):
    return C2F(K2C(degree))
```

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw$ python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from p14 import *
>>> C2F(80)
176.0
>>> F2C(176)
80.0
>>> C2K(80)
353.15
>>> K2C(353.15)
80.0
>>> F2K(176.0)
353.15
>>> K2F(353.15)
176.0
>>>
```

Problem 15

Extend the module built in Problem 14 with a main program in the test block. This main program should read the first command line argument as a numerical value of a temperature and the second argument as a temperature scale: **C, K, F**. Write out the temperature in the other two scales. For example, if **21.3 C** is given on the command line, the output should be **70.34 F 294.45 K.**

<Source Code>

```
from p14 import *
import sys

s = sys.argv[1]
n = len(s)
num = float(s[0:n - 1])

if(s[-1] == 'C'):
    print("%fF, %fK"%(C2F(num), C2K(num)))

elif(s[-1] == 'F'):
    print("%fC, %fK"%(F2C(num), F2K(num)))

elif(s[-1] == 'K'):
    print("%fC, %fF"%(K2C(num), K2F(num)))

else:
    print("please C, F, K")
```

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw$ vim p15.py
ubuntu@ubuntu-VirtualBox:~/ossw$ python3 p15.py 21.3C
70.340000F, 294.450000K
ubuntu@ubuntu-VirtualBox:~/ossw$ python3 p15.py 70.34F
21.300000C, 294.450000K
ubuntu@ubuntu-VirtualBox:~/ossw$ python3 p15.py 294.45K
21.300000C, 70.340000F
ubuntu@ubuntu-VirtualBox:~/ossw$
```

Problem 16

Write a program that counts words, delimiters, and lines for any paragraph in English.

<Source Code>

```
def paragraph_count(paragraph):
    line_count = 1
    word_count = 0
    delim_count = 0
    delim = set([' ', '.', ',', '!', '?'])
    linec = set(['\n'])
    for line in paragraph:
        linecom = set(line)
        compare = set(line)
        line_count += len(linecom.intersection(linec))
        delim_count += len(compare.intersection(delim))
        line = line.split()
        word_cout += len(line)
    print("line_count : %d, delim_count : %d, word_count : %d"%(line_count, delim_count, word_count))
```

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw$ python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from p16 import *
>>> str = ''' fskf ,emfneksndfngegnefm.demsd 1212!@!@!!?? dsfdfsd
... sdlfs rjnwvsnvnsm  mdnf,mnfsf ef1!@31"
... dkfdlkfsf s"DFDSF" '''
>>> paragraph_count(str)
line_count : 3, delim_count : 20, word_count : 100
>>>
```

Problem 17

(Simulate operations on lists by hands) You are given the following program:

```
1  a = [ 1, 3, 5, 7, 9]
2  b = [ 13, 17]
3  c = a + b
4  print(c)
5  b[0] = -1
6  d = [ e + 1 for e in a ]
7  print(d)
8  d.append(b[0]+1)
9  d.append(b[-1] + 1)
10 print(d[:-2])
```

Explain what is printed by each print statement.

<Source Code>

```
a = [1, 3, 5, 7, 9]
b = [13, 17]
c = a + b
print(c)
b[0] = -1
d = [e + 1 for e in a]
print(d)
d.append(b[0] + 1)
d.append(b[-1] + 1)
print(d[:-2])

# what is printed by each print statement.
```

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw$ vim p17.py
ubuntu@ubuntu-VirtualBox:~/ossw$ python3 p17.py
[1, 3, 5, 7, 9, 13, 17]
[2, 4, 6, 8, 10]
[2, 4, 6, 8, 10]
ubuntu@ubuntu-VirtualBox:~/ossw$
```

Problem 18

We define the following nestes list:

```
1 q = [['a','b','c'],['d','e','f'],['g','j']]
```

Index this list to extract 1) letter 'a'; 2) the list ['d', 'e', 'f']; 3) the last element 'h'; 4) the 'd' element. Explain why q[-1][-2] has the value 'g', Code and test your program.

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw$ python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from p18 import *
>>> q[0][0]
'a'
>>> q[1]
['d', 'e', 'f']
>>> q[1][0]
'd'
>>> q[-1][-2]
'g'
>>> q.append('h')
>>> q
[['a', 'b', 'c'], ['d', 'e', 'f'], ['g', 'j'], 'h']
>>>
```

Problem 19

Consider the list from Problem 18. We can visit all elements of q using this instead for loop:

```
1 for i in q:
2     for j in range(len(i)):
3         print(i[j])
```

What type of objects are i and j?

<Source Code>

```
from p18 import *

for i in q:
    type(i)
    for j in range(len(i)):
        type(j)
        print(i[j])
```

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw$ python3 p19.py
a
b
c
d
e
f
g
ubuntu@ubuntu-VirtualBox:~/ossw$
```

Problem 20

Rewrite the generation of the nested list q,

```
1 q = [r**2 for r in [10**i for i in range(5)]]
```

by using standard for loops instead of list comprehensions.

<Result>

```
ubuntu@ubuntu-VirtualBox:~/ossw$ python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from p20 import *
>>> q
[1, 100, 10000, 1000000, 100000000]
>>>
```