SP17_Intro to Data Science

Problem 5

Student: Lina Mi

PRACTICE EXERCISES

1. Write and run a Python program that outputs the value of each of the following expressions:

5.0/9.0
5.0/9
5/9.0
5/9
9.0/5.0
9.0/5
9/5.0
9/5

now please try it with the truncated division operator:

5.0//9.0
5.0//9
5//9.0
5//9
9.0//5.0
9.0//5
9//5.0
9//5

Based on your results, what is the rule for arithmetic operators when integers and floating point numbers are used?

Ans

Python program:

```
def main():
    print("5.0/9.0=", 5.0/9.0)
    print("5.0/9=", 5.0/9)
    print("5/9.0=", 5/9.0)
    print("5/9=", 5/9)
    print("9.0/5.0=", 9.0/5.0)
    print("9.0/5=", 9.0/5)
    print("9/5.0=", 9/5.0)
    print("9/5.0=", 9/5.0)
    print("9/5=", 9/5)
    print("5.0//9.0=", 5.0//9.0)
    print("5.0//9=", 5.0//9)
    print("5//9.0=", 5//9.0)
    print("5//9=", 5//9)
    print("9.0//5.0=", 9.0//5.0)
    print("9.0//5=", 9.0//5)
    print("9//5.0=", 9//5.0)
    print("9//5.0=", 9//5.0)
    print("9//5=", 9//5)
main()
```

the output result :

```
RESTART: C:/Users/milin/AppData/Local/Programs/Python/Python36-32/practice/print_data.py
5.0/9.0= 0.5555555555555556
5.0/9= 0.5555555555555556
5/9.0= 0.5555555555555556
```

```
5/9= 0.5555555555555556
9.0/5.0= 1.8
9.0/5= 1.8
9/5.0= 1.8
9/5.0= 1.8
9/5= 1.8
5.0//9.0= 0.0
5.0//9= 0.0
5//9.0= 0.0
5//9= 0
9.0//5.0= 1.0
9.0//5= 1.0
9//5.0= 1.0
9//5.0= 1.0
9//5= 1
```

From the result it can be seen that if there is a floating number in arithmetic operation, the result will be floating number, and if the result of arithmetic operation is not integer, the result will expressed in form of floating number. Only when all of operands in arithmetic operation are integers and the operation result is also integer(not decimal), the result will be expressed in integer.

2. Write and run a Python program that asks the user for a temperature in Celsius and converts and outputs the temperature in
Fahrenheit. (Use the formula given in the example above and solve for tempF in terms of tempC.)

Ans: ***Python program:***

```
def main():
    tempC=int(input("input the temperature in Celsius:"))
    tempF=(9.0/5.0)*tempC+32.0
    print("the temperature in Fahrenheit:", tempF)
main()
```

the result of running program:

```
RESTART: C:/Users/milin/AppData/Local/Programs/Python/Python36-
32/practice/temp_conversion_C_to_F.py

input the temperature in Celsius:19
the temperature in Fahrenheit: 66.2
```

3. Here is an algorithm to print out n! (n factorial) from 0! to 19!:

```
1. Set f = 1
2. Set n = 0
3. Repeat the following 20 times: # use range(1,20) although range(20) will also work but not match #5
a. Output n, "! = ", f
b. Add 1 to n
c. Multiply f by n
```

Using a for loop, write and run a Python program for this algorithm.

Ans: ***Python Program:***

```
    def main():
     f=1
     n=0
     for n in range(0, 20):
         print(n,"!=",f)
         n=n+1
         f=f*n
    main()
```

The result of running previous program:

```
RESTART: C:\Users\milin\AppData\Local\Programs\Python\Python36-32\practice\n_factorial.py
0 != 1
1 != 1
2 != 2
3 != 6
4 != 24
5 != 120
6 != 720
7 != 5040
8 != 40320
9 != 362880
10 != 3628800
11 != 39916800
12 != 479001600
13 != 6227020800
14 != 87178291200
15 != 1307674368000
16 != 20922789888000
17 != 355687428096000
18 != 6402373705728000
19 != 121645100408832000
```

4. Modify the program above using a while loop so it prints out all of the factorial values that are less than 1 billion. (You should be able to do this without looking at the output of the previous exercise.)

Ans: ***Python program***:

```
def main():
    f=1
    n=0
    while f<10**9:
        print(n,"!=",f)
        n=n+1
        f=f*n
main()
```

the result of running program:

```
RESTART: C:/Users/milin/AppData/Local/Programs/Python/Python36-
32/practice/n_factorial_while.py
0 != 1
1 != 1
2 != 2
```

```
3 != 6
4 != 24
5 != 120
6 != 720
7 != 5040
8 != 40320
9 != 362880
10 != 3628800
11 != 39916800
12 != 479001600
```

5. Repeat #3 above using list comprehensions. Please see these increasingly more sophisticated references for list comprehensions: Simple, Advanced, Complex (has a hint for #5).

***Python program**:*

```
def main():
  from functools import reduce
  list_1=[reduce(lambda x,y:x*y, range(1,y),1) for y in range(1,21)]
  list_2=[print(n,"!=",list_1[n]) for n in range(0,20)]
  return
main()
```

the result of previous program:

```
>>RESTART: C:/Users/milin/AppData/Local/Programs/Python/Python36-
32/practice/n_factorial_list.py
0 != 1
1 != 1
2 != 2
3 != 6
4 != 24
5 != 120
6 != 720
7 != 5040
8 != 40320
9 != 362880
10 != 3628800
11 != 39916800
12 != 479001600
13 != 6227020800
14 != 87178291200
15 != 1307674368000
16 != 20922789888000
17 != 355687428096000
18 != 6402373705728000
19 != 121645100408832000
>>>
```