LAB # 06

NESTED STATEMENTS, BREAK AND CONTINUE STATEMENTS

OBJECTIVE

Working on nested statements and control loop iteration using break and continue.

THEORY

Nested Statements:

A Nested statement is a statement that is the target of another statement. **Nested if:**

A Nested ***if*** is an ***if*** statement that is the target of another ***if*** statement. Nested ***if*** statements means an ***if*** statement inside another ***if*** statement.

Syntax:

```
if (condition1):
   # Executes when condition1 is true    if
(condition2):
      # Executes when condition2 is true
   # if Block is end here
# if Block is end here
```

Example:

```
#using nested if
x=int(input("enter number="))
y=int(input("enter 2nd number="))
if x > 2:     if y > 2:
z = x + y          print("z is",
z)  else:     print("x is", x)
```

Output:

```
>>> %Run task1.py
enter number=3
enter 2nd number=8
       z is 11 >>>
```

Nested loops:

Nested loops consist of an outer loop and one or more inner loops. Each time the outer loop repeats, the inner loops are reinitialize and start again.

Example:

```
height=int(input("Enter height:
")) for row in range(1, height):
for column in range(1,height):
print(row, end=" ")
print()
```

Output:

```
>>> %Run task2.py
Enter height: 7
1 1 1 1 1 1
2 2 2 2 2 2
3 3 3 3 3 3
4 4 4 4 4 4
5 5 5 5 5 5
6 6 6 6 6 6
```

Keywords break and continue:

The break and continue keywords provide additional controls to a loop.

The Break Statement:

The keyword ***break*** in a loop to immediately terminate a loop. Listing example presents a program to demonstrate the effect of using ***break*** in a loop.

Syntax: break

Example:

```
# Use of break statement inside
loop for word in "string":     if
word == "i":
        break
print(word) print("The
end")
```

Output:

```
>>> %Run task3.py'
s t r The end >>>
```

The continue Statement:

The ***continue*** statement breaks out of the current iteration in the loop.

Syntax: continue

Example:

```
# Program to show
the use of continue
statement inside
loops for val in
"string":     if
val == "i":
continue
    print(word)
```

```
print("The end"
```

Output:

```
>>> %Run task4.py' s t r n g
The end
>>>
```

EXERCISE

A. Point out the errors, if any, in the following Python programs.

1. Code

```
prompt = "\nPlease enter the name of a city you have visited:"
prompt+="\n(Enter 'quit' when you are finished.)"  while
True:     city = str(input(prompt))     if city == quit:
break;     else:         print("I'd love to go to " ,
city.title() , "!")
```

Output

In this program there is an indended block before if condition

2. **Code**

```
if x>2:    if
y>2:
z=x+y
      print(“z is”,
y)    else
print(“x is”, x)
```

Output

In this program variable is not assigned.

2. Code

```
balance = int(input("enter your
balance1:")) while true:  if balance
<=9000:     continue;   balance =
balance+999.99 print("Balance is",
balance)
```

Output

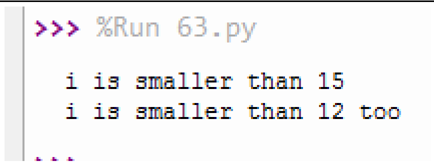
In this program variable true is not defined .

B. What will be the output of the following programs:

1. Code

```
i = 10 if (i ==
10):
    #  First if statement      if (i <
15):           print ("i is smaller than
15")
    # Nested - if statement
    # Will only be executed if statement above
# it is true      if (i < 12):
print ("i is smaller than 12 too")      else:
print ("i is greater than 15")
```

Output



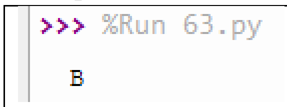
```
>>> %Run 63.py

  i is smaller than 15
  i is smaller than 12 too

...
```

2. Code

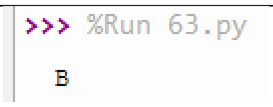
```
i = 1 j = 2 k = 3 if i
> j:      if i > k:
print('A')  else:
print('B')
```

```
i = 1 j = 2 k = 3 if
i > j:     if
i> k:
print('A')     else:
        print('B')
```

Output :



```
>>> %Run 63.py

  B
```

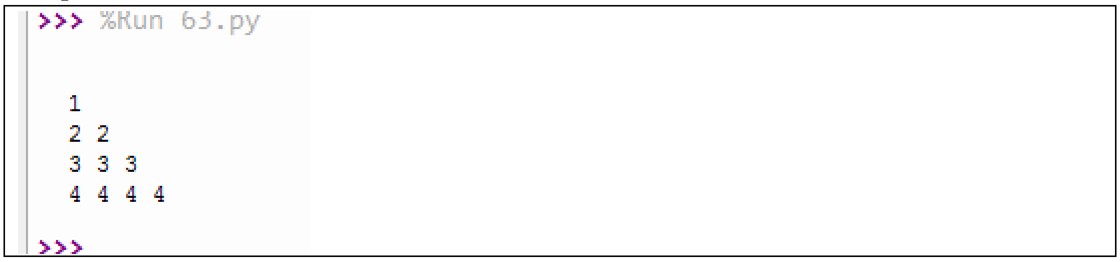


```
>>> %Run 63.py

  B
```

3. Code

```
# nested for loops  for i
in range(0, 5):  for j
in range(i):
print(i, end=' ')
print()
```

Output



```
>>> %Run 63.py


  1
  2 2
  3 3 3
  4 4 4 4

>>>
```

C. Write Python programs for the following:

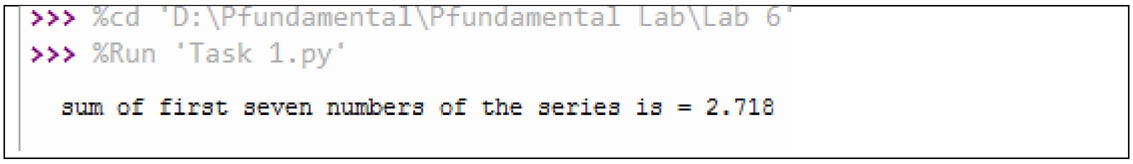
1. Write a program to add first seven terms twice of the following series:

$$\frac{1}{1!}+\frac{2}{2!}+\frac{3}{3!}+\cdots$$

CODE:

```
s=0
for num in range(1,8):
   factorial=1
   for i in range(1,num+1):
      factorial=factorial*i

   factorial_s= num/factorial
   s= s+factorial_s

print("sum of first seven numbers of the series is
=",round(s,3))
```

OUTPUT:



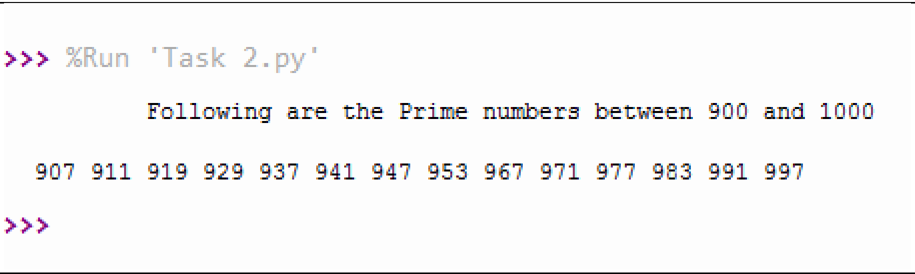
```
>>> %cd 'D:\Pfundamental\Pfundamental Lab\Lab 6'
>>> %Run 'Task 1.py'

  sum of first seven numbers of the series is = 2.718
```

2. Write a program to print all prime numbers from 900 to 1000.
   [Hint: Use nested loops, break and continue]

Code:

```
a = 900
b = 1000

print("\tFollowing are the Prime numbers between", a, "and", b,"\n")

for num in range(a, b + 1):
   if num > 1:
      for i in range(2, num):
         if (num % i) == 0:
            break

      else:
         print(num, end =" ")
```

OUTPUT:



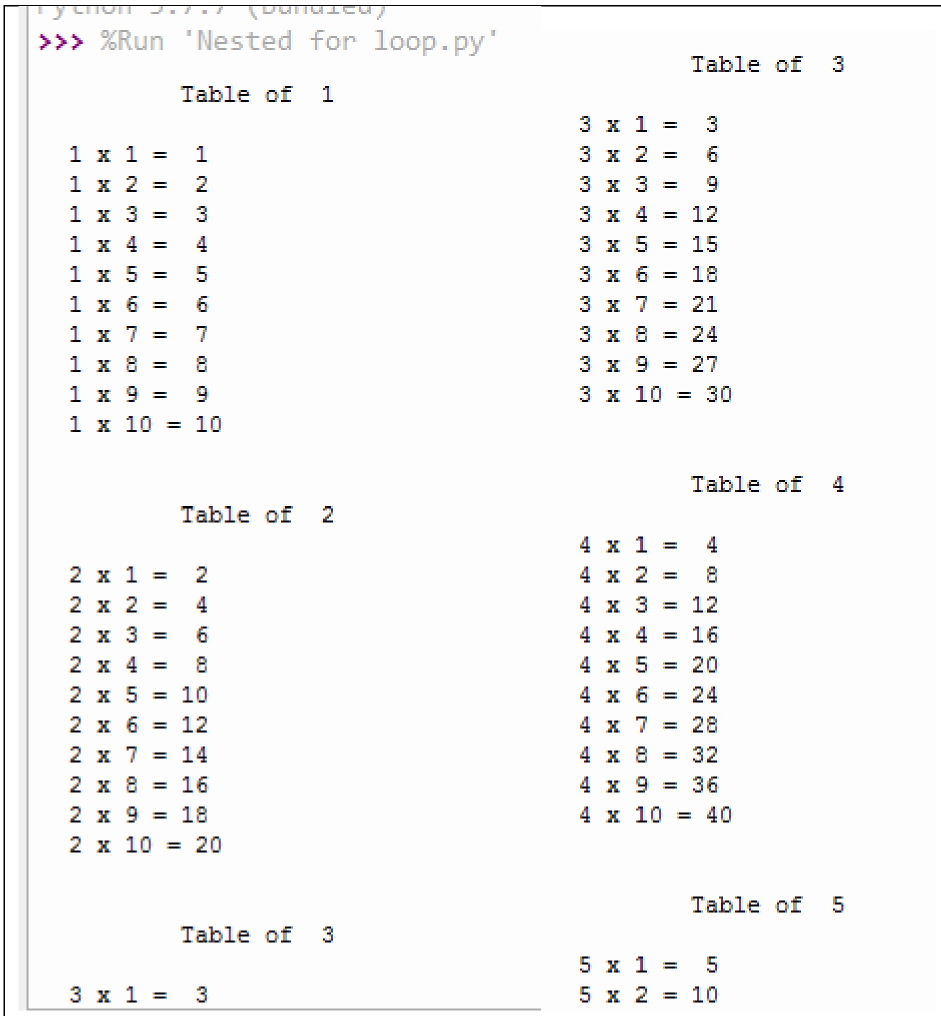
```
>>> %Run 'Task 2.py'

          Following are the Prime numbers between 900 and 1000

  907 911 919 929 937 941 947 953 967 971 977 983 991 997
>>>
```

3. Write a program to display multiplication table(1-5) using nested looping
   Sampled output:[hint: '{ } ' .format(value)]
   02 X 01 = 02

CODE:

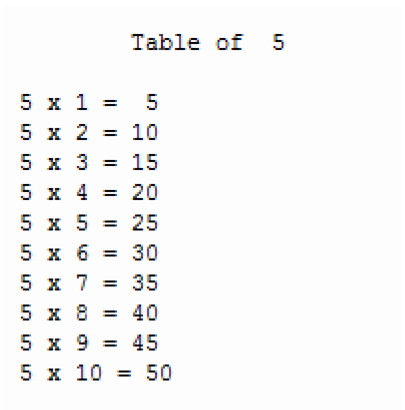
```
for i in range (1,6):
   print ("\tTable of ",i,"\n")
   for j in range(1,11):
      print(i,"x",j,"=","{:2d}".format(i * j) )
   print("\n")
```

OUTPUT:



```
>>> %Run 'Nested for loop.py'

         Table of  1

  1 x 1 =  1
  1 x 2 =  2
  1 x 3 =  3
  1 x 4 =  4
  1 x 5 =  5
  1 x 6 =  6
  1 x 7 =  7
  1 x 8 =  8
  1 x 9 =  9
  1 x 10 = 10


         Table of  2

  2 x 1 =  2
  2 x 2 =  4
  2 x 3 =  6
  2 x 4 =  8
  2 x 5 = 10
  2 x 6 = 12
  2 x 7 = 14
  2 x 8 = 16
  2 x 9 = 18
  2 x 10 = 20


         Table of  3

  3 x 1 =  3
```

```
        Table of  3

3 x 1 =  3
3 x 2 =  6
3 x 3 =  9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30


        Table of  4

4 x 1 =  4
4 x 2 =  8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40


        Table of  5

5 x 1 =  5
5 x 2 = 10
```



```
        Table of  5

5 x 1 =  5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```