NAME	Muhammad Arslan Raza
ROLL#	2020-EE-403

Lab 3: Problem Set 1

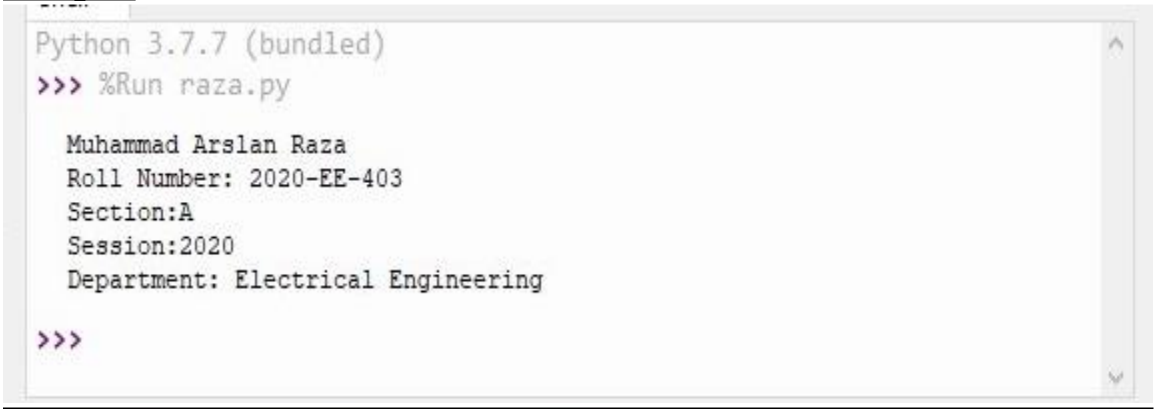
Task 1:

Try to print your name, roll number, section, session and department on output screen.

Code :

```
print("Muhammad Arslan Raza")

print("Roll Number: 2020-EE-403")

print("Section:A")

print("Session:2020")

print("Department: Electrical Engineering")
```

Output:



```
Python 3.7.7 (bundled)
>>> %Run raza.py

  Muhammad Arslan Raza
  Roll Number: 2020-EE-403
  Section:A
  Session:2020
  Department: Electrical Engineering

>>>
```

Conclusion: Today I learn how to print your name, roll number, section, session and department on output screen

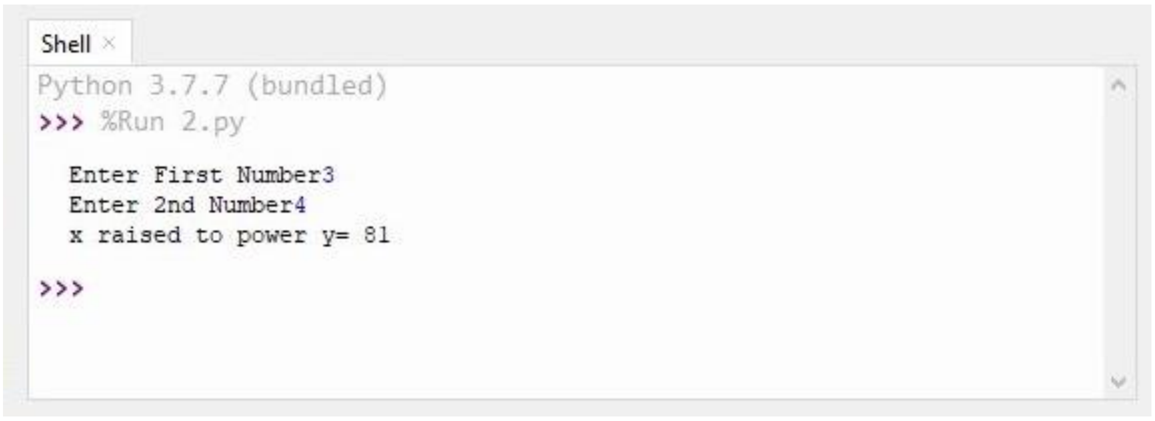
Task 2:

Ask user to print two numbers ‘x’ and ‘y’ then print out number ‘x’ raised to power ‘y’.

Code:

```
x=int(input("Enter First Number"))

y=int(input("Enter 2nd Number"))

z=x**y

print("x raised to power y=",z)
```

Output:



```
Python 3.7.7 (bundled)
>>> %Run 2.py

  Enter First Number3
  Enter 2nd Number4
  x raised to power y= 81

>>>
```

Conclusion: Today I learn how to print two numbers ‘x’ and ‘y’ then print out number ‘x’ raised to power ‘y’.

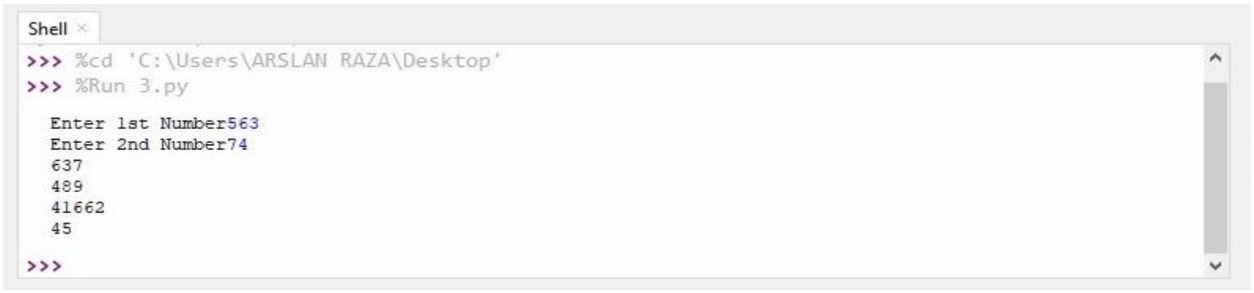
<u>Task 3</u>:

Generate two random integers numbers ‘x’ & ‘y’ from users and then print addition, subtraction, division and multiplication of the two numbers. Also observe the output generated by the operation *x % y*.

<u>Code</u>:

```
x=int(input("Enter 1st Number"))

y=int(input("Enter 2nd Number"))

print(x+y)

print(x-y)

print(x/y)

print(x*y)

print(x%y)
```

<u>Output</u>:



```
Shell
>>> %cd 'C:\Users\ARSLAN RAZA\Desktop'
>>> %Run 3.py

  Enter 1st Number563
  Enter 2nd Number74
  637
  489
  41662
  45

>>>
```

<u>Conclusion</u>: Today I learn how to input two integers numbers ‘x’ & ‘y’ from users and then print addition, subtraction, division and multiplication of the two numbers. Also observe the output generated by the operation *x % y*.

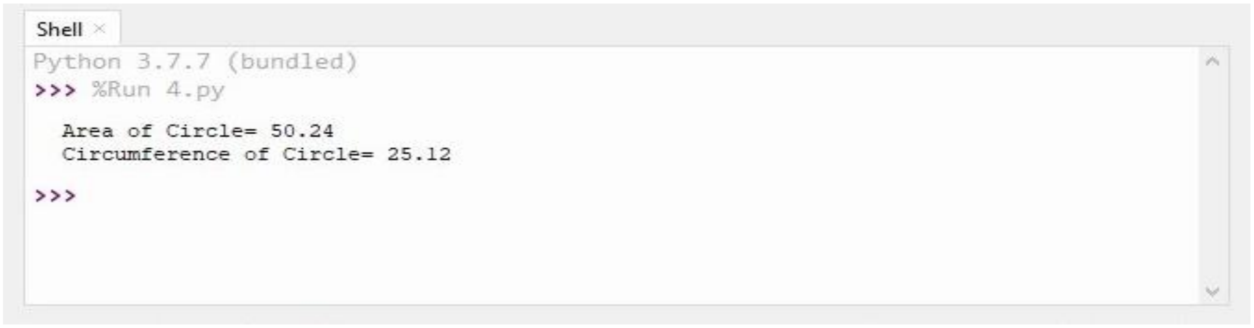
Task 4:

Take two variables ‘pi’ and ‘radius’. Assign 3.14 to *pi* and 4 to *radius.* Calculate and print the area and circumference of the circle.

Code:

```
radius=4

pi=3.14

print("Area of Circle=",pi*radius**2)

print("Circumference of Circle=",2*pi*radius)
```

Output:



```
Python 3.7.7 (bundled)
>>> %Run 4.py

  Area of Circle= 50.24
  Circumference of Circle= 25.12

>>>
```

Conclusion: Today I learn how calculate and print the area and circumference of the circle.

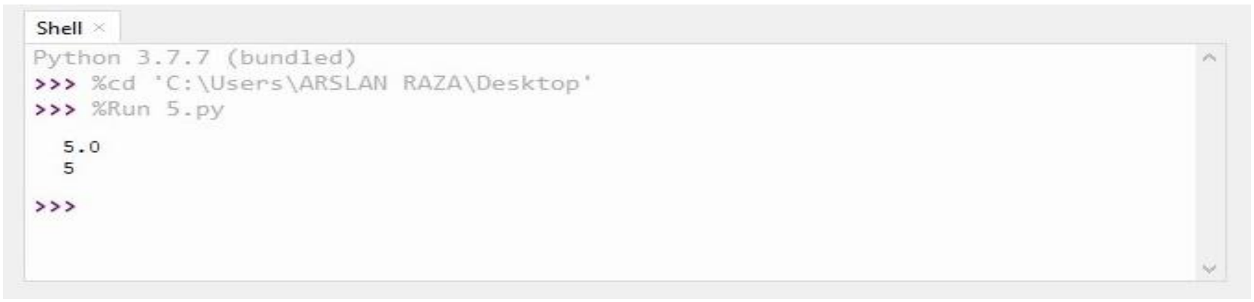
Task 5:

Take two variables ‘x’ and ‘y’. Save an integer value (e.g. 5) in *x* and floating value (e.g. (5.7) in *y*. Change the type of integer value to *float* and floating value to *int.* Now print both numbers and observe the result.

Code:

```
x=5

y=5.7

y=int(x)

x=float(y)

print(x)

print(y)
```

Output:



```
Shell ×
Python 3.7.7 (bundled)
>>> %cd 'C:\Users\ARSLAN RAZA\Desktop'
>>> %Run 5.py

  5.0
  5

>>>
```

Conclusion: Today I learn how to take two variables ‘x’ and ‘y’. Save an integer value (e.g. 5) in *x* and floating value (e.g. (5.7) in *y*. Change the type of integer value to *float* and floating value to *int. now* print both numbers and observe the result

NAME	Muhammad Arslan Raza
ROLL#	2020-EE-403

Lab 4: Problem Set 2

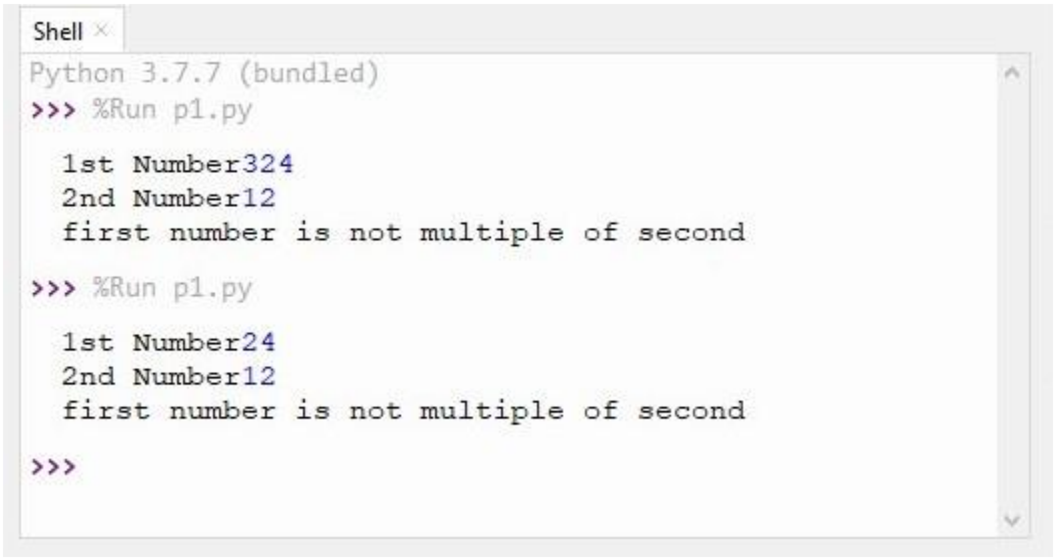
Task 1:

Write a program that takes two integers from user and determines and prints whether the first is a multiple of the second.

Code:

```
x=int(input("1st Number"))

y=int(input("2nd Number"))

if y%x==0:

    print("first number is a multiple of second")

else:

    print("first number is not multiple of second")
```

Output:



```
Python 3.7.7 (bundled)
>>> %Run p1.py

  1st Number324
  2nd Number12
  first number is not multiple of second

>>> %Run p1.py

  1st Number24
  2nd Number12
  first number is not multiple of second

>>>
```

Conclusion:

Today I learn how to write a program that takes two integers from user and determines and prints whether the first is a multiple of the second.

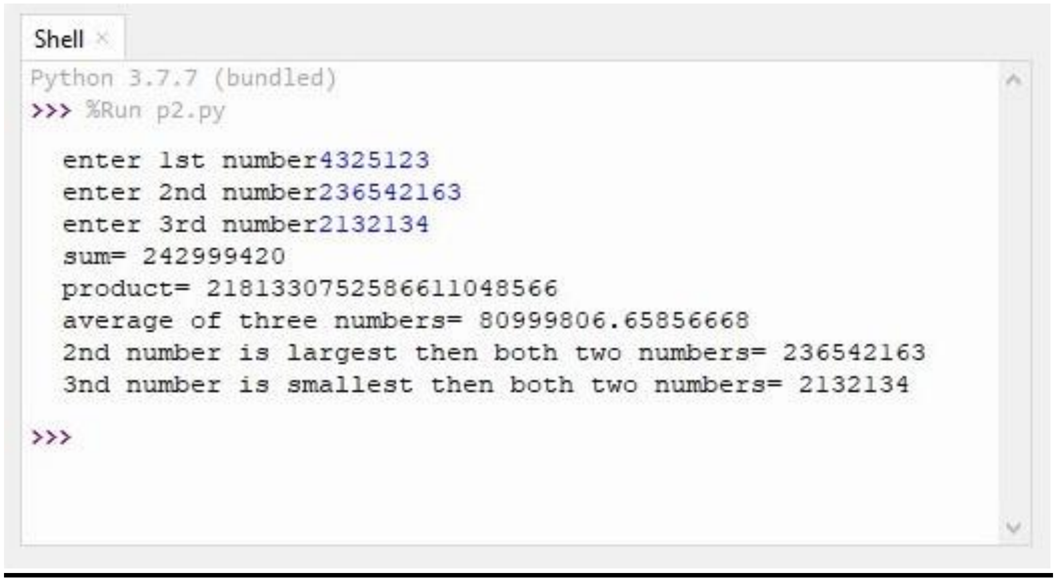
Task 2:

Write a program that inputs three different integers from the keyboard, and then prints the sum, the average, the product, the smallest and the largest of these numbers.

Code:

```
x=int(input("enter 1st number"))
y=int(input("enter 2nd number"))
z=int(input("enter 3rd number"))
print("sum=",x+y+z)
print("product=",x*y*z)
print("average of three numbers=",(x+y+z)*0.3333333333)
if x>y>z:
    print("first number is largest then both two numbers=",x)
else:
     if x>z>y:
        print("first number is largest then both two numbers=",x)
if y>x>z:
    print("2nd number is largest then both two numbers=",y)
else:
    if y>z>x:
        print("2nd number is largest then both two numbers=",y)
if z>y>x:
     print("3nd number is largest then both two numbers=",z)
else:
    if z>x>y:
         print("3nd number is largest then both two numbers=",z)
if x<y<z:
    print("first number is smallest then both two numbers=",x)
else:
     if x<z<y:
        print("first number is smallest then both two numbers=",x)


if y<x<z:
    print("2nd number is smallest then both two numbers=",y)
else:


    if y<z<x:
        print("2nd number is smallest then both two numbers=",y)
if z<y<x:
     print("3nd number is smallest then both two numbers=",z)
else:
    if z<x<y:
         print("3nd number is smallest then both two numbers=",z)
```

<u>Output</u>:



```
Python 3.7.7 (bundled)
>>> %Run p2.py

  enter 1st number4325123
  enter 2nd number236542163
  enter 3rd number2132134
  sum= 242999420
  product= 2181330752586611048566
  average of three numbers= 80999806.65856668
  2nd number is largest then both two numbers= 236542163
  3nd number is smallest then both two numbers= 2132134

>>>
```

<u>Conclusion</u>: Today I learn how write a program that inputs three different integers from the keyboard, and then prints the sum, the average, the product, the smallest and the largest of these numbers.

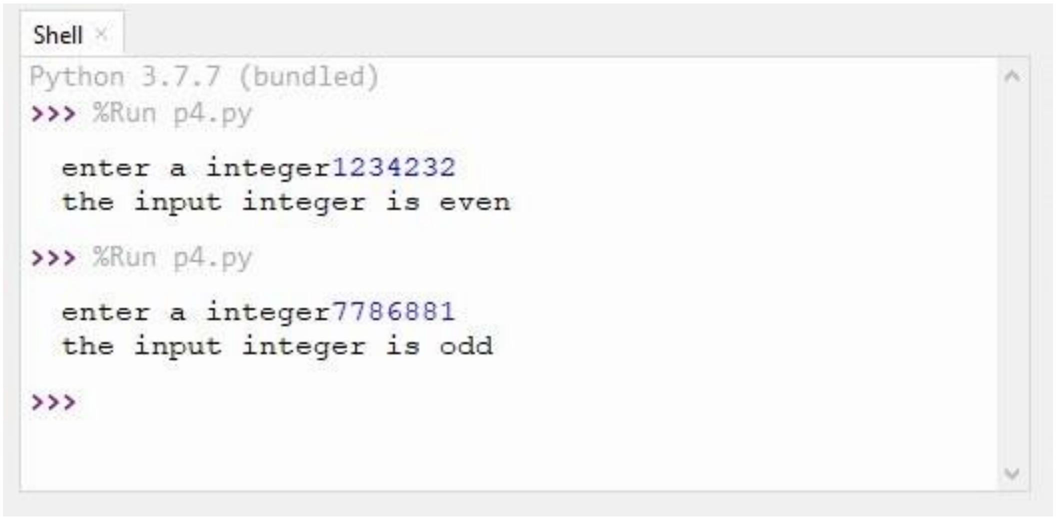
Task 3:

Write a program that reads an integer and determines and prints whether it’s odd or even.

Code:

```
x=int(input("enter a integer"))


if x%2==0:
    print("the input integer is even")
else:
    print("the input integer is odd")
```

Output:



```
Python 3.7.7 (bundled)
>>> %Run p4.py

  enter a integer1234232
  the input integer is even

>>> %Run p4.py

  enter a integer7786881
  the input integer is odd

>>>
```

Conclusion: Today I learn how write a program that reads an integer and determines and prints whether it’s odd or even.

NAME	Muhammad Arslan Raza
ROLL#	2020-EE-403

Lab 5: Problem Set 2

Objective: The objective of this problem set is how to write different programs on compiler

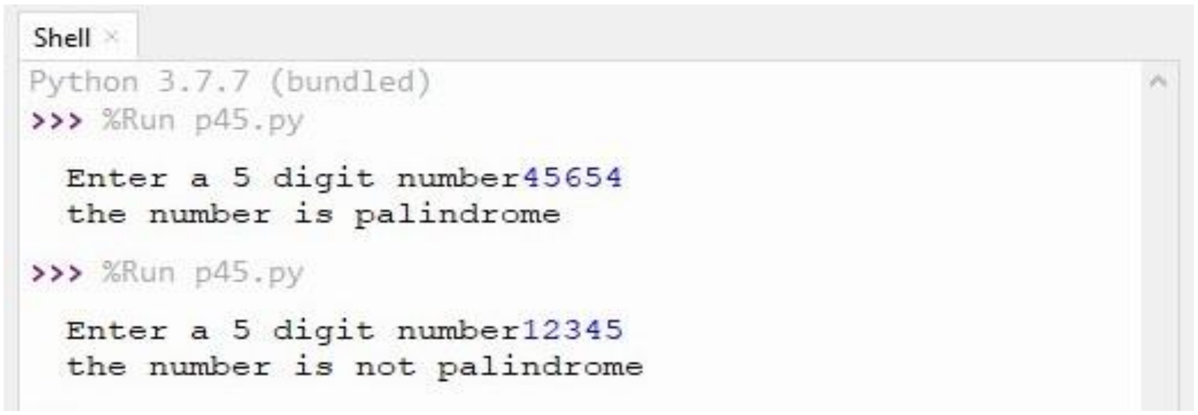
Task 1:

A palindrome is a number or a text phrase that reads the same backward as forward. For example, each of the following five-digit integers is a palindrome: 12321, 55555, 45554 and 11611. Write a program that reads in a five-digit integer and determines whether or not it’s a palindrome.

Code:

```
x=int(input("Enter a 5 digit number"))

a=x%10

y=int(x/10)

b=y%10

z=int(y/10)

c=z%10

w=z%10

w=int(z/10)

d=w%10

e=int(w/10)

if e==a and d==b:

    print("the number is palindrome")

else:

    print("the number is not palindrome")
```

Output:



```
Python 3.7.7 (bundled)
>>> %Run p45.py

  Enter a 5 digit number45654
  the number is palindrome

>>> %Run p45.py

  Enter a 5 digit number12345
  the number is not palindrome
```

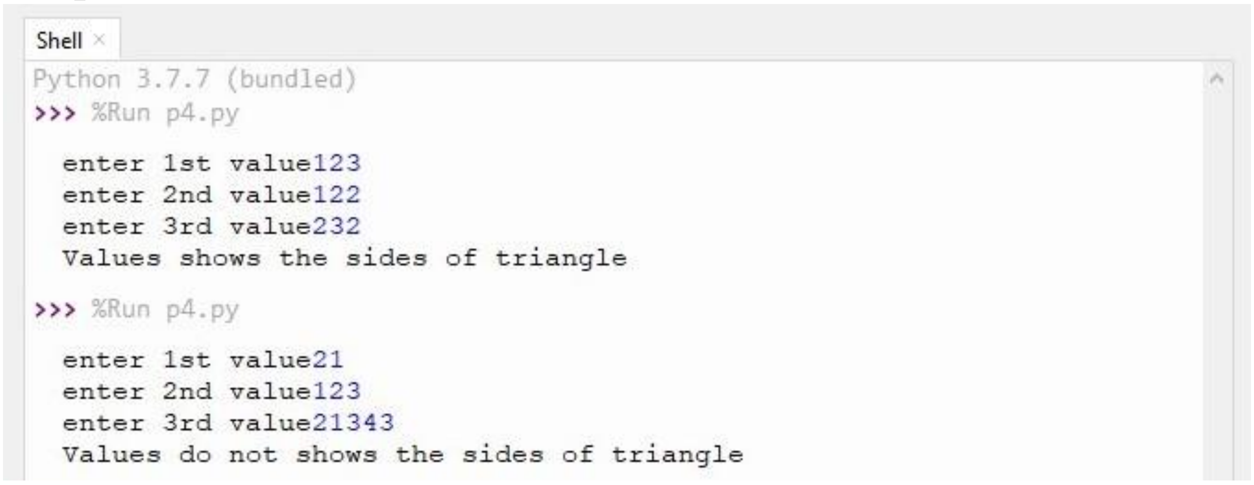
Task 2:

Write a program that reads three non-zero integer values and determines and prints whether they could represent the sides of a triangle.

Code:

```
x=int(input("enter 1st value"))

y=int(input("enter 2nd value"))

z=int(input("enter 3rd value"))

if x+y>=z and y+z>=x and x+z>=y:

    print("Values shows the sides of triangle")

else:

    print("Values do not shows the sides of triangle")
```

Output:



```
Python 3.7.7 (bundled)
>>> %Run p4.py

  enter 1st value123
  enter 2nd value122
  enter 3rd value232
  Values shows the sides of triangle

>>> %Run p4.py

  enter 1st value21
  enter 2nd value123
  enter 3rd value21343
  Values do not shows the sides of triangle
```

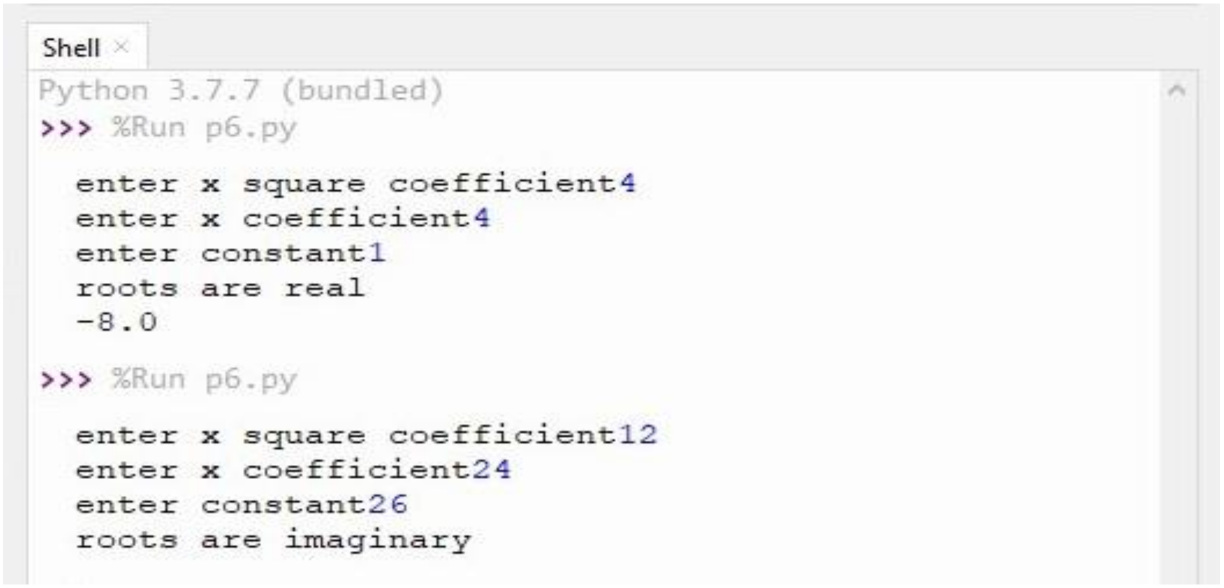
Task 3:

Implement a program which determines the roots of a quadratic equation (take coefficients from user). The program also determines if the roots of the equation are real or imaginary. Display the roots if they are real. Print a message on the output screen about the nature of the roots (*roots are real* or *roots are imaginary*).

Code:

```
import math
a=int(input("enter x square coefficient"))
b=int(input("enter x coefficient"))
c=int(input("enter constant"))
if b*b-4*a*c>=0:
    print("roots are real")
    y=b**2-4*a*c
    x1=(-b+(math.sqrt(y)))/2*a
    x2=(-b-(math.sqrt(y)))/2*a
    print(x)
else:
    print("roots are imaginary")
```

Output:



```
Shell
Python 3.7.7 (bundled)
>>> %Run p6.py

  enter x square coefficient4
  enter x coefficient4
  enter constant1
  roots are real
  -8.0

>>> %Run p6.py

  enter x square coefficient12
  enter x coefficient24
  enter constant26
  roots are imaginary
```

Conclusion:

Today I learn how to write a program that determine whether the input numbers are palindrome or not. Enter values shows the sides of triangle or not. And write a program that determine the roots of quadratic.

NAME	Muhammad Arslan Raza
ROLL#	2020-EE-403

Lab 5: Problem Set 2

Objective: The objective of this problem set is how to write different programs on compiler

Task 1:

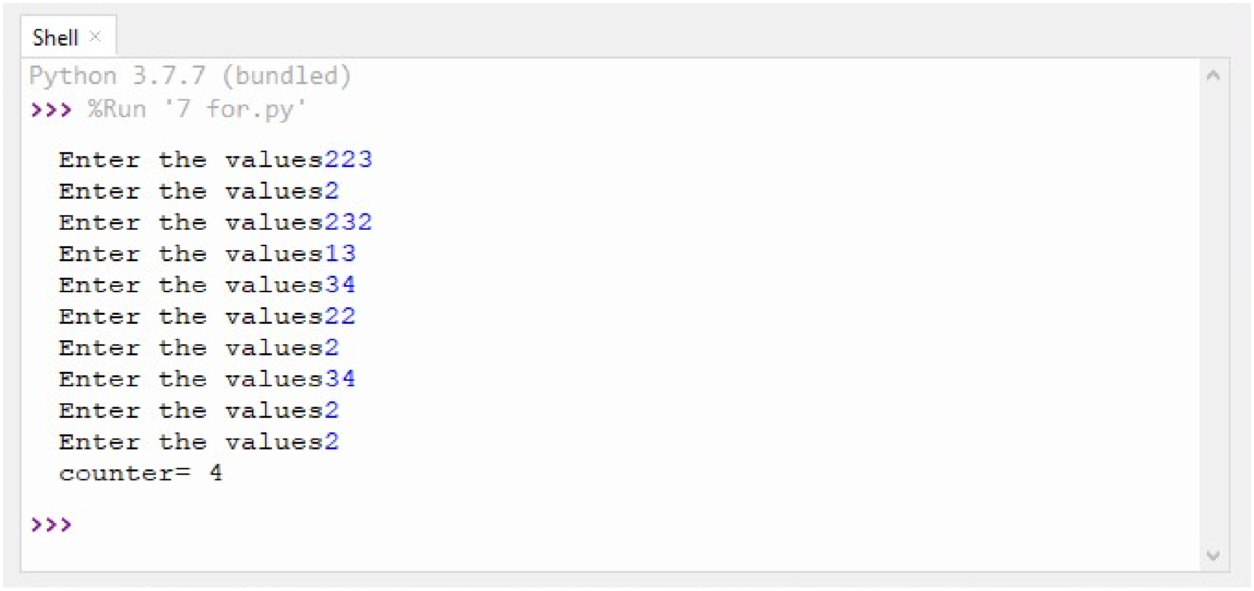
Implement a program that prompts the user to enter 10 integers (there can be repetition). The program will count the number of times the user enters the integer '2'. The count will be displayed on the output screen. Implement using both *while* and *for* loops.

With for loop:

Code:

```
counter=0

for i in range(0,10,1):

    a=int(input("Enter the values"))

    if a==2:

        counter=counter+1

print("counter=",counter)
```

Output:



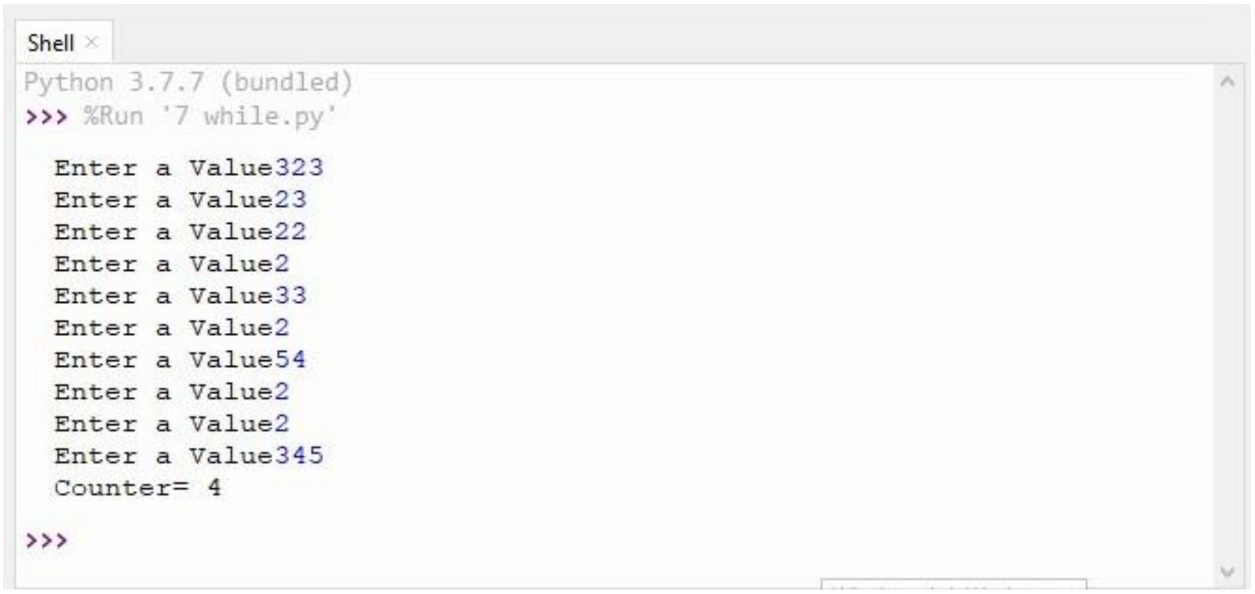
```
Shell ×
Python 3.7.7 (bundled)
>>> %Run '7 for.py'

  Enter the values223
  Enter the values2
  Enter the values232
  Enter the values13
  Enter the values34
  Enter the values22
  Enter the values2
  Enter the values34
  Enter the values2
  Enter the values2
  counter= 4

>>>
```

With while loop:

Code:

```
counter=0

x=0

while x<10:

    a=int(input("Enter a Value"))

    if a==2:

        counter=counter+1

    x=x+1

print("Counter=",counter)
```

Output:



```
Shell ×
Python 3.7.7 (bundled)
>>> %Run '7 while.py'

  Enter a Value323
  Enter a Value23
  Enter a Value22
  Enter a Value2
  Enter a Value33
  Enter a Value2
  Enter a Value54
  Enter a Value2
  Enter a Value2
  Enter a Value345
  Counter= 4

>>>
```

Task 2:

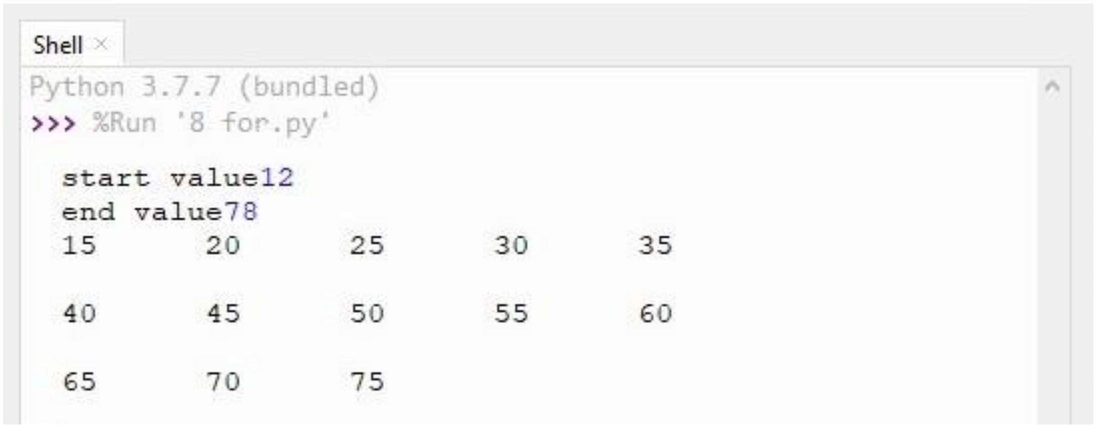
Implement a program that prints multiples of 5 in the range defined by user such that only 5 numbers are printed on one row. Implement using both *while* and *for* loops.

With for Loop:

Code:

```
a=int(input("start value"))

b=int(input("end value"))

count=0

i=a

for i in range(a,b):
    if i%5==0:
         print(i,end='\t')
         count=count+1
    if count==5:
        count=0
        print("\n")
```

Output:



Shell

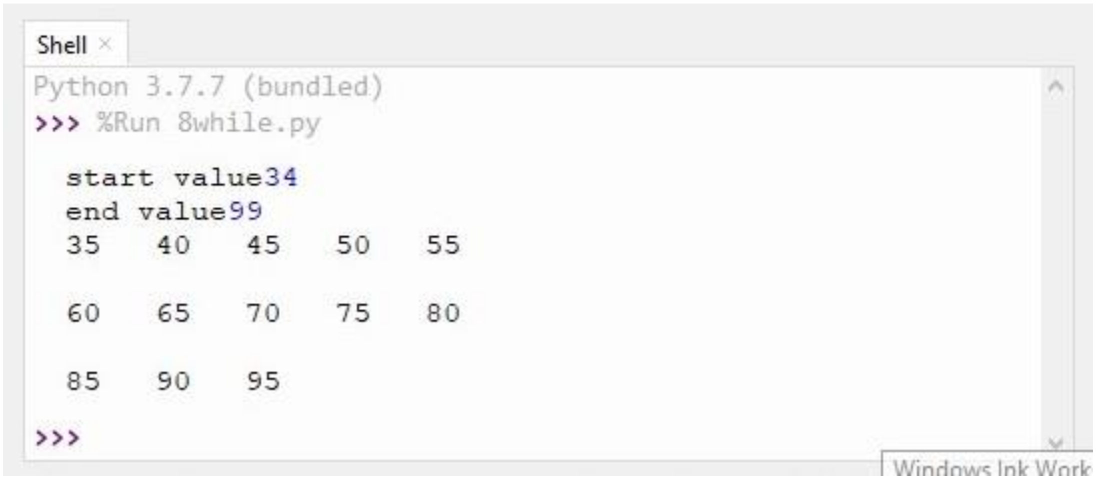
```
Python 3.7.7 (bundled)
>>> %Run '8 for.py'

  start value12
  end value78
  15      20      25      30      35

  40      45      50      55      60

  65      70      75
```

With while loop:

Code:

```
a=int(input("start value"))
b=int(input("end value"))
count=0
i=a
while i<b:
    if i%5==0:
        print(i," ",end=" ")
        count=count+1
    if count==5:
        count=0
        print("\n")
    i=i+1
```

Output:



```
Shell ×
Python 3.7.7 (bundled)
>>> %Run 8while.py

  start value34
  end value99
  35   40   45   50   55

  60   65   70   75   80

  85   90   95

>>>
```

Task 3:

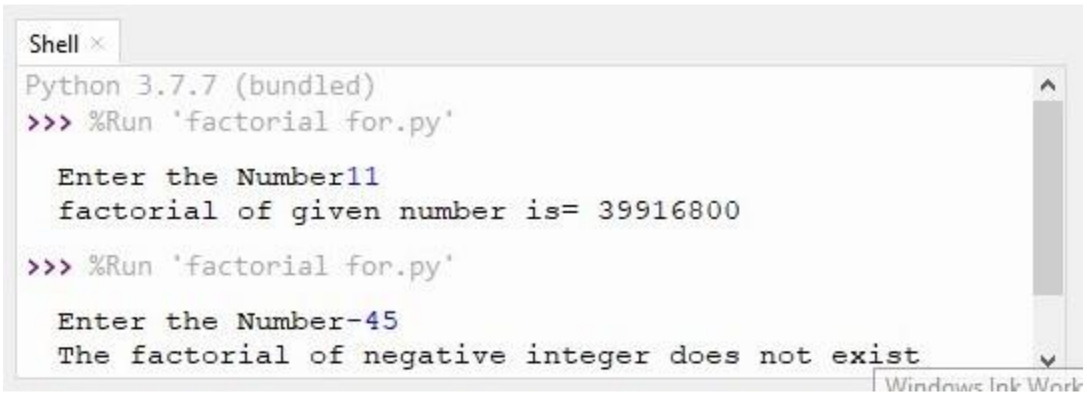
Take a number from the user and print its factorial. Implement using both *while* and *for* loops.

With for Loop:

Code:

```
x=int(input("Enter the Number"))

factorial=1

if x<0:

    print("The factorial of negative integer does not exist")

if x==0:

    print("The factorial of 0 is=",1)

else:

    for x in range(1,x+1):

        factorial=factorial*x

    print("factorial of given number=",factorial)
```

Output:



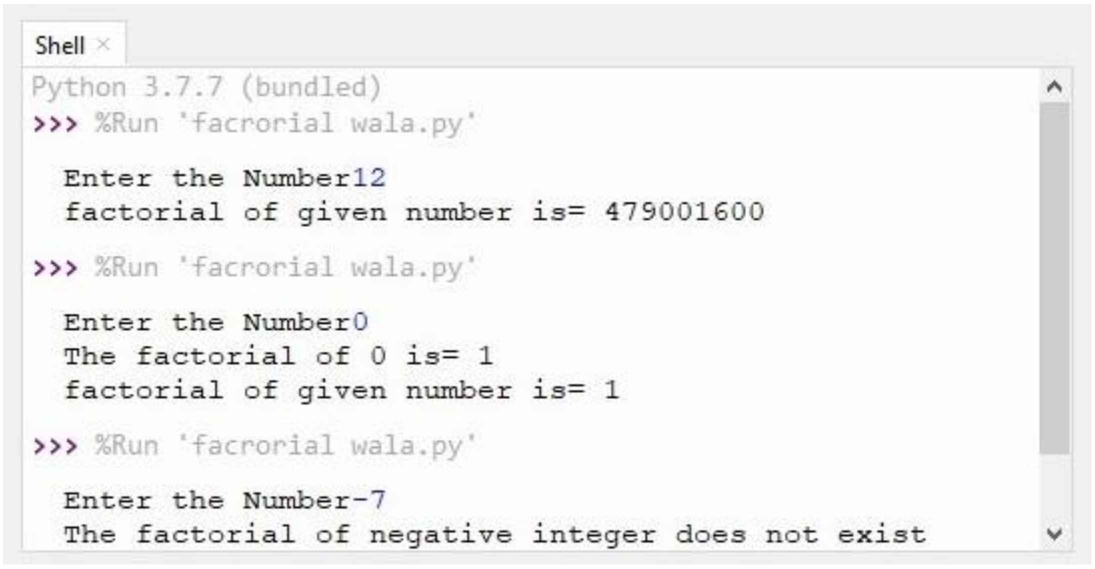
```
Shell ×
Python 3.7.7 (bundled)
>>> %Run 'factorial for.py'

  Enter the Number11
  factorial of given number is= 39916800

>>> %Run 'factorial for.py'

  Enter the Number-45
  The factorial of negative integer does not exist
```

With while loop:

Code:

```
x=int(input("Enter the Number"))
factorial=1
if x<0:
    print("The factorial of negative integer does not exist")
if x==0:
    print("The factorial of 0 is=",1)
else:
    while x>0:
        factorial=factorial*x
        x=x-1
    print("factorial of given number is=",factorial)
```

Output:



Shell ×

```
Python 3.7.7 (bundled)
>>> %Run 'facrorial wala.py'

  Enter the Number12
  factorial of given number is= 479001600

>>> %Run 'facrorial wala.py'

  Enter the Number0
  The factorial of 0 is= 1
  factorial of given number is= 1

>>> %Run 'facrorial wala.py'

  Enter the Number-7
  The factorial of negative integer does not exist
```

Conclusion:

Today I learn how to write a program that prompts the user to enter 10 integers. The program will count the number of times the user enters the integer ‘2’. The count will be displayed on the output screen, write a program that prints multiples of 5 in the range defined by user such that only 5 numbers are printed on one row. And write a program that determine the factorial of given number.

NAME	Muhammad Arslan Raza
ROLL#	2020-EE-403

Lab 7: Problem Set 2

Objective: The objective of this problem set is how to write different programs on compiler

Task 1:

Implement a code that estimates the value of mathematical constant ‘e’ according to the following formula:

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

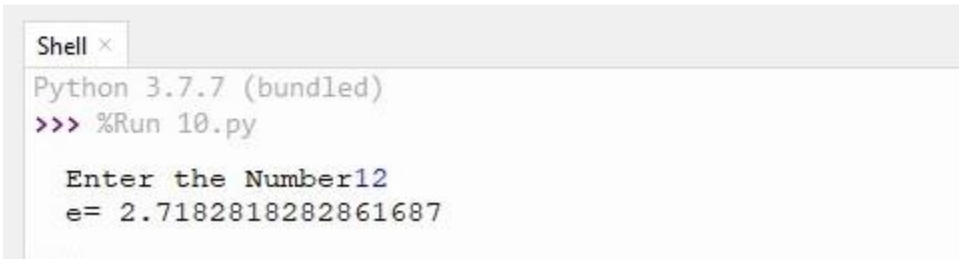
Adding more terms to calculate ‘e’ will make the value of ‘e’ more accurate. Prompt the user to enter the number of terms for the desired accuracy of the value of ‘e’

With for Loop:

Code:

```
x=int(input("Enter the Number"))

factorial=1

e=1

for i in range(1,x+1):

    factorial=factorial*i

    e=e+(1/factorial)

print("e=",e)
```

Output:



```
Shell ×
Python 3.7.7 (bundled)
>>> %Run 10.py

  Enter the Number12
  e= 2.7182818282861687
```

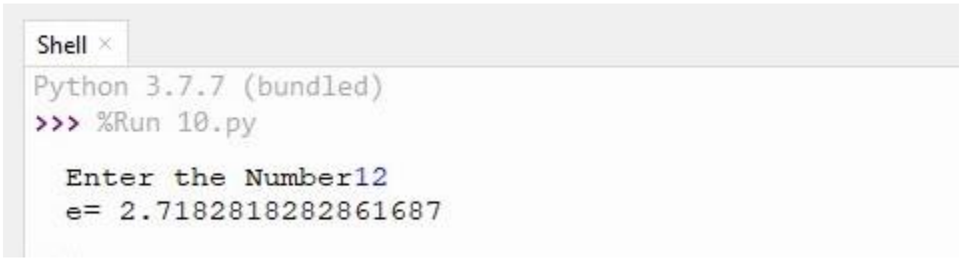
With while Loop:

Code:

```
x=int(input("Enter the Number"))

factorial=1

e=1

i=1
```

```
while i<x:
    factorial=factorial*i
    e=e+(1/factorial)
    i=i+1
print(e)
```

Output:



```
Shell ×
Python 3.7.7 (bundled)
>>> %Run 10.py

  Enter the Number12
  e= 2.7182818282861687
```

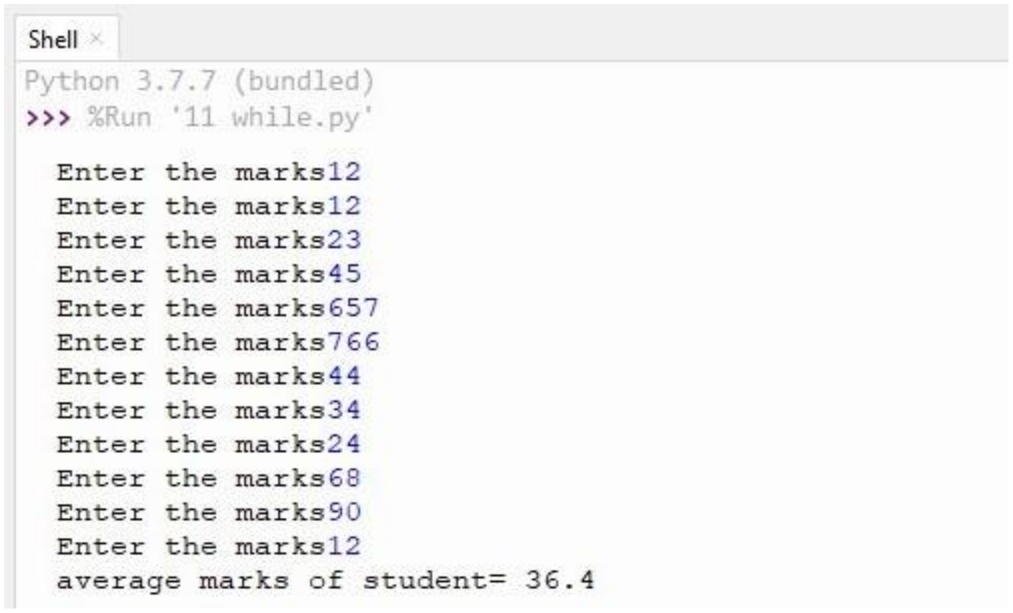
Task 2:

Ask the user to provide final exam marks of 10 students in *Introduction to Computing* course in order to calculate the average. Use *while* loop for this purpose, and use *break* statement to exit the loop when an invalid input (e.g. negative number) is given. Moreover, if marks greater than 100 are given, use *continue* statement to skip that number from average calculation.

Code:

```
n=0
sum=0
while n<10:
    x=int(input("Enter the marks"))
    if x<0:
        break
    if x>100:
        continue
    n=n+1
    sum=sum+x
print("average marks of student=",sum/10)
```

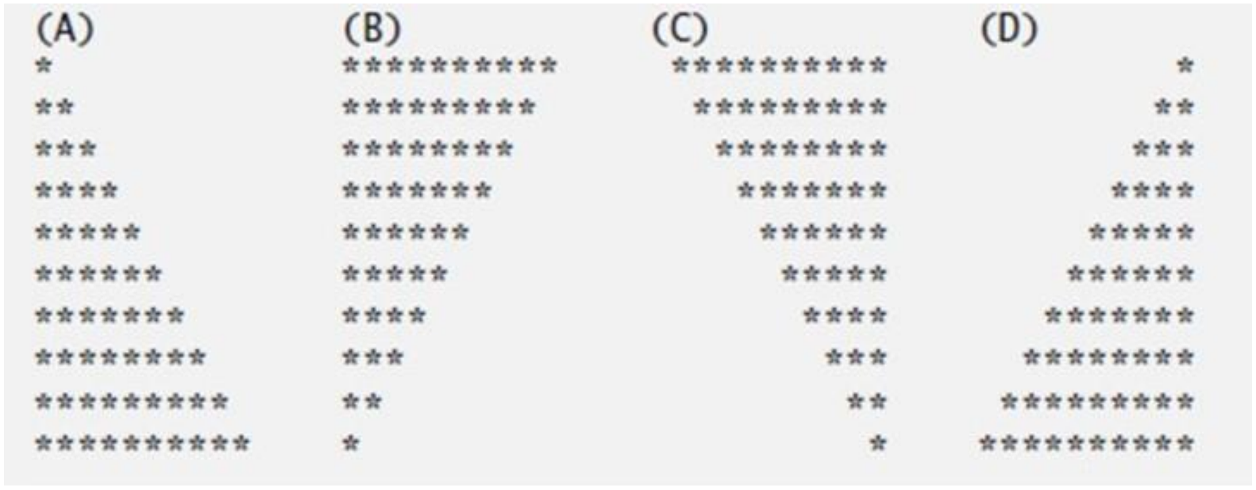
Output:



```
Python 3.7.7 (bundled)
>>> %Run '11 while.py'

  Enter the marks12
  Enter the marks12
  Enter the marks23
  Enter the marks45
  Enter the marks657
  Enter the marks766
  Enter the marks44
  Enter the marks34
  Enter the marks24
  Enter the marks68
  Enter the marks90
  Enter the marks12
  average marks of student= 36.4
```

Task 3:

Write programs that print following patterns separately



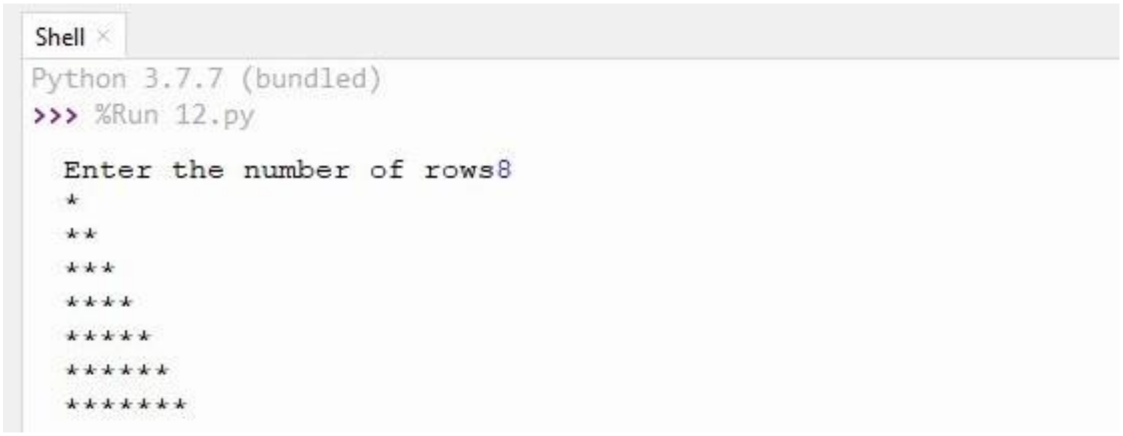
```
(A)             (B)             (C)             (D)
*               **********      **********               *
**              *********        *********              **
***             ********          ********             ***
****            *******            *******            ****
*****           ******              ******           *****
******          *****                *****          ******
*******         ****                  ****         *******
********        ***                    ***        ********
*********       **                      **       *********
**********      *                        *      **********
```

A:

Code:

```
x=int(input("Enter the number of rows"))
for i in range(1,x):
    for j in range(0,i):
        print("*",end="")
    print()
```

Output:



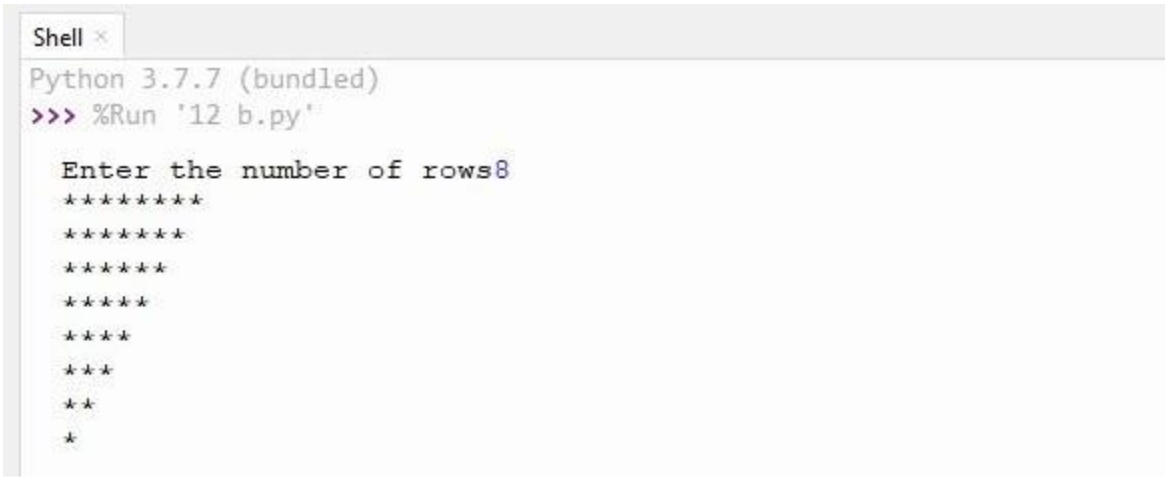
```
Shell ×
Python 3.7.7 (bundled)
>>> %Run 12.py

 Enter the number of rows8
 *
 **
 ***
 ****
 *****
 ******
 *******
```

B:

Code:

```
x=int(input("Enter the number of rows"))
for i in range(x,0,-1):
    for j in range(0,i):
        print("*",end="")
    print()
```

Output:



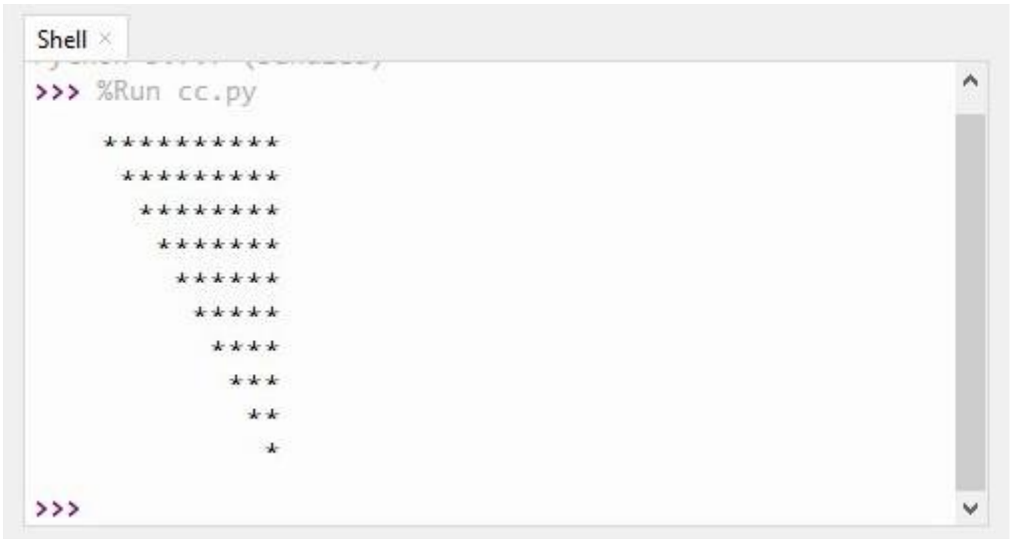
```
Shell ×
Python 3.7.7 (bundled)
>>> %Run '12 b.py'

 Enter the number of rows8
 ********
 *******
 ******
 *****
 ****
 ***
 **
 *
```

C:

Code:

```
i=1
while i<11:
    i=i+1
    k=0
    while k<i:
         k=k+1
         print(" ",end= "")
    j=11
    while j>i-1:
        j=j-1
        print("*",end= "")
    print()
```

Output:



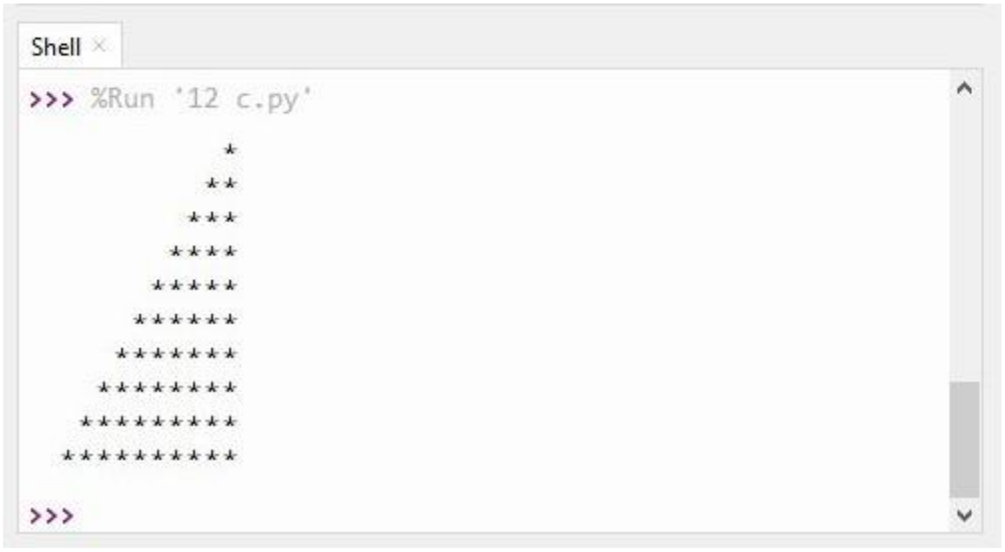
```
>>> %Run cc.py
    **********
     *********
      ********
       *******
        ******
         *****
          ****
           ***
            **
             *
>>>
```

D:

Code:

```
i=1
while i<11:
    i=i+1
    k=10
    while k>i-1:
         k=k-1
         print(" ",end="")
    j=0
    while j<i-1:
        j=j+1
        print("*",end="")
    print()
```

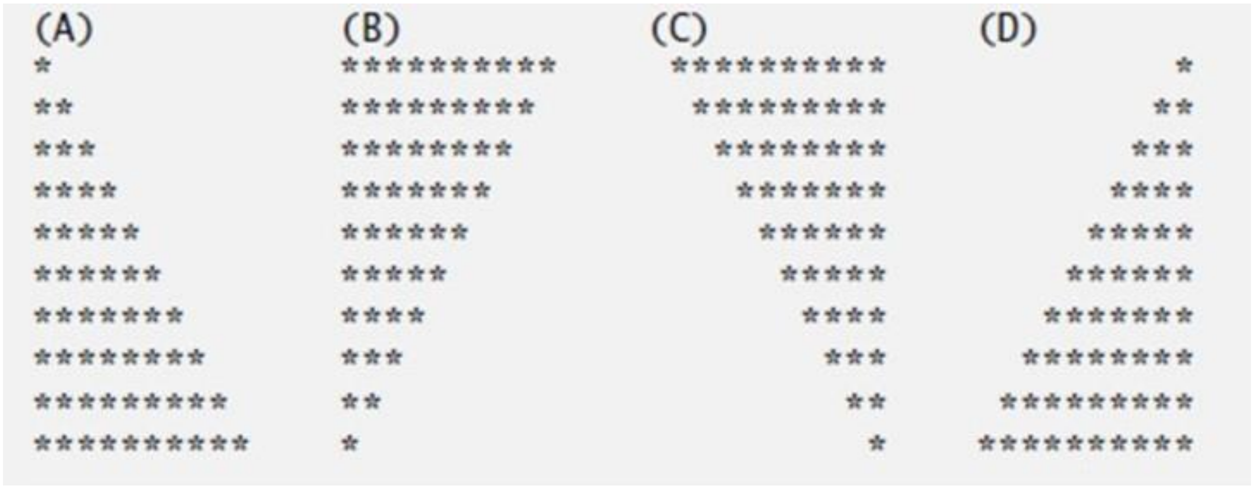
<u>Output</u>:



```
Shell
>>> %Run '12 c.py'
         *
        **
       ***
      ****
     *****
    ******
   *******
  ********
 *********
**********

>>>
```

<u>Conclusion</u>:

Today I learn how to write a program which calculate the value of “e” and implement which Ask the user to provide final exam marks of 10 students in *Introduction to Computing* course in order to calculate the average. Use *while* loop for this purpose, and use *break* statement to exit the loop when an invalid input (e.g negative number) is given. Moreover, if marks greater than 100 are given, use *continue* statement to skip that number from average calculation. And write programs that print following patterns separately.



```
(A)           (B)           (C)           (D)
*             **********    **********             *
**            *********      *********            **
***           ********        ********           ***
****          *******          *******          ****
*****         ******            ******         *****
******        *****              *****        ******
*******       ****                ****       *******
********      ***                  ***      ********
*********     **                    **     *********
**********    *                      *    **********
```

NAME	Muhammad Arslan Raza
ROLL#	2020-EE-403

Lab 8: Problem Set 3

Objective: The objective of this problem set is how to write different programs on compiler

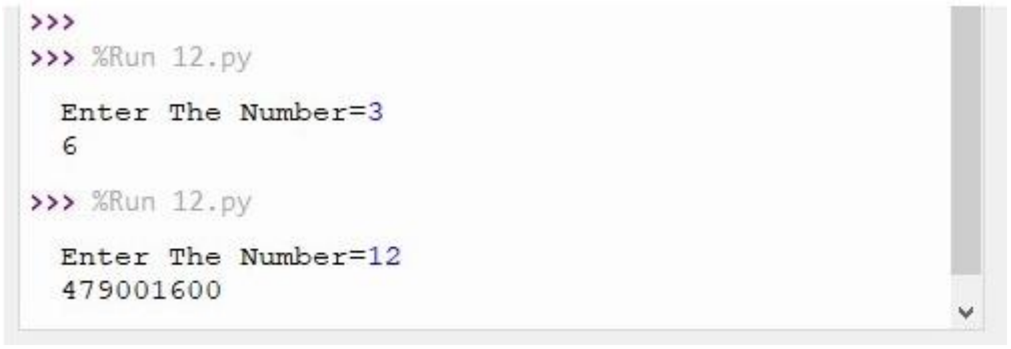
Task 1:

Implement the following function:
a) *factorial (n)*--- which takes a number as parameter and return its factorial.
Call the above function. Initialize an integer variable from the user and pass it as parameter to the above function. Then display the factorial

Code:

```
def factorial(n):
    factorial=1
    for i in range(1,x+1):
        factorial=factorial*i
    return(factorial)
x=int(input("Enter The Number=",))
print(factorial(x))
```

Output:



```
>>>
>>> %Run 12.py

 Enter The Number=3
 6

>>> %Run 12.py

 Enter The Number=12
 479001600
```

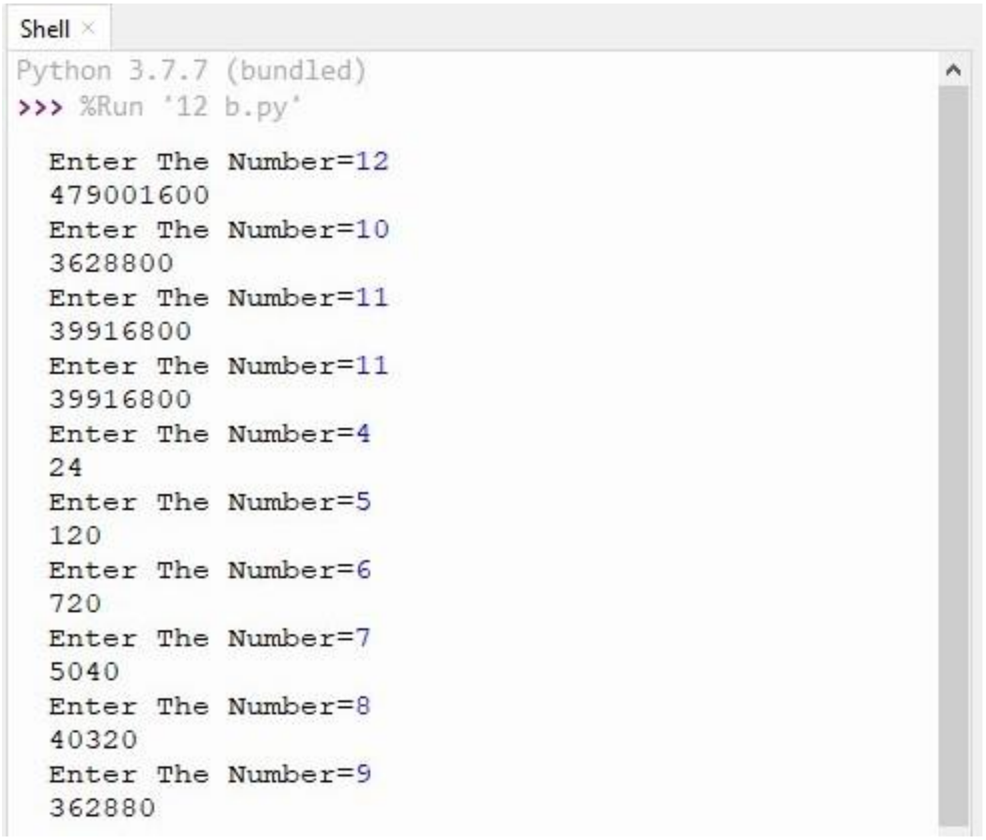
Task 2:

Modify the factorial program to call the factorial function 10 times while passing a user defined integer each time and displaying the corresponding factorial.

Code:

```
def factorial(n):
    factorial=1
    for i in range(1,x+1):
        factorial=factorial*i
    return(factorial)
for i in range(10):
    x=int(input("Enter The Number=",))
    print(factorial(x))
```

Output:



```
Shell
Python 3.7.7 (bundled)
>>> %Run '12 b.py'

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

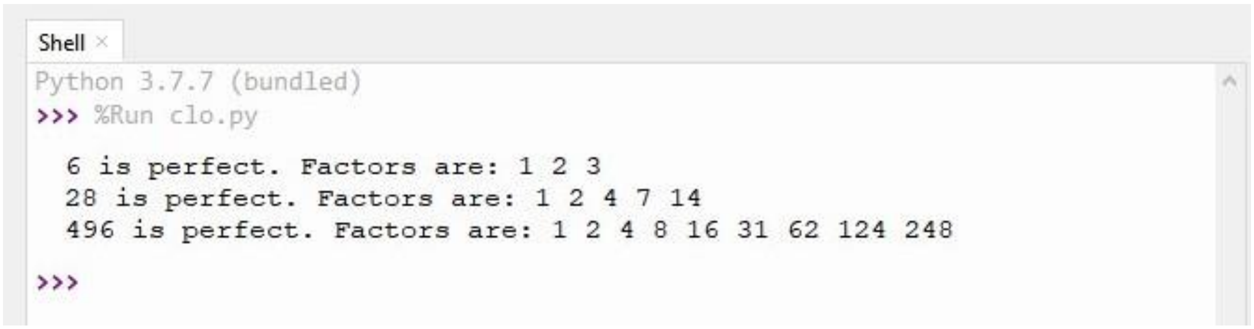
Task 3:

An integer number is said to be a perfect number if the sum of its factors, including 1 (but not the number itself), is equal to the number. For example, 6 is a perfect number, because 6 = 1 + 2 + 3. Write a function *perfect* that determines whether parameter *number* is a perfect number. Use this function in a program that determines and prints all the perfect numbers between 1 and 1000. Print the factors of each perfect number to confirm that the number is indeed perfect.

Code:

```
def is_perfect(x):
    sum = 0
    for n in range (1,x):
        if x % n == 0:
            sum =sum+n
    if sum == x:
        return True
    else:
        return False
def factors(x):
    factors = []
    for n in range (1,x):
        if x % n == 0:
            factors.append(n)
    return factors
for num in range(1,1000):
    if is_perfect(num) == True:
        print(num,'is perfect.',end = ' ')
        print ('Factors are:',end = ' ')
        for x in factors(num):
            print(x,end=' ')
        print()
```

Output:



```
Shell ×
Python 3.7.7 (bundled)
>>> %Run clo.py

  6 is perfect. Factors are: 1 2 3
  28 is perfect. Factors are: 1 2 4 7 14
  496 is perfect. Factors are: 1 2 4 8 16 31 62 124 248

>>>
```

Conclusion:

Today I learn how to write a program which takes a number as parameter and return its factorial. Call the above function. Initialize an integer variable from the user and pass it as parameter to the above function. Then display the factorial, Modify the factorial program to call the factorial function 10 times while passing a user defined integer each time and displaying the corresponding factorial. And An integer number is said to be a perfect number if the sum of its factors, including 1 (but not the number itself), is equal to the number. For example, 6 is a perfect number, because 6 = 1 + 2 + 3. Write a function *perfect* that determines whether parameter *number* is a perfect number. Use this function in a program that determines and prints all the perfect numbers between 1 and 1000. Print the factors of each perfect number to confirm that the number is indeed perfect.

NAME	Muhammad Arslan Raza
ROLL#	2020-EE-403

Lab 9: Problem Set 3

Objective: The objective of this problem set is how to write different programs on compiler

Task 1:

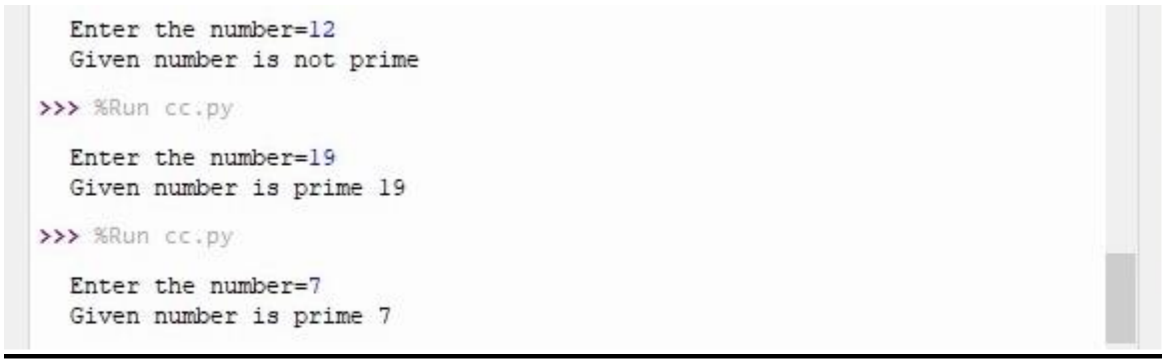
An integer greater than 1 is said to be prime if it is divisible by only 1 and itself. For example, 2, 3, 5 and 7 are prime numbers, but 4, 6, 8 and 9 are not.
i) Write a function that determines whether a number is prime.
ii) Use this function in a program that determines and prints all the prime numbers between 2 and 1,000.
iii) Initially, you might think that n/2 is the upper limit for which you must test to see whether a number is prime, but you need go only as high as the square root of n.
Rewrite the program and run it both ways to show that you get the same result.

i)

Code:

```
def print_prime(n):
    count=0
    for i in range(1,n):
        if n%i==0:
            count=count+1
    if count==1:
        print("Given number is prime",n)
    else:
        print("Given number is not prime")
num=int(input("Enter the number="))
print_prime(num)
```

Output:



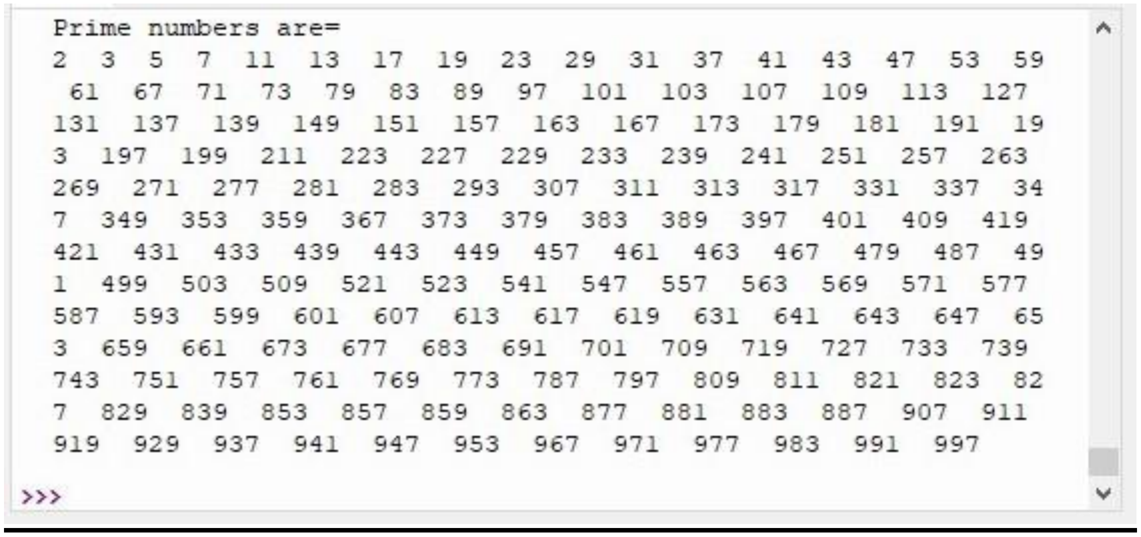
```
  Enter the number=12
  Given number is not prime
>>> %Run cc.py
  Enter the number=19
  Given number is prime 19
>>> %Run cc.py
  Enter the number=7
  Given number is prime 7
```

ii)

Code:

```
def print_prime(n):
    count=0
    for i in range(1,n):
        if n%i==0:
            count=count+1
    if count==1:
        print(n,end = '  ')
print('prime numbers are=')
for x in range(2,1000):
    print_prime(x)
```

Output:



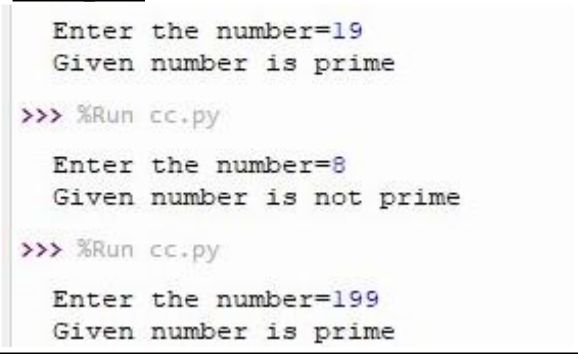
```
Prime numbers are=
2  3  5  7  11  13  17  19  23  29  31  37  41  43  47  53  59
 61  67  71  73  79  83  89  97  101  103  107  109  113  127
131  137  139  149  151  157  163  167  173  179  181  191  19
3  197  199  211  223  227  229  233  239  241  251  257  263
269  271  277  281  283  293  307  311  313  317  331  337  34
7  349  353  359  367  373  379  383  389  397  401  409  419
421  431  433  439  443  449  457  461  463  467  479  487  49
1  499  503  509  521  523  541  547  557  563  569  571  577
587  593  599  601  607  613  617  619  631  641  643  647  65
3  659  661  673  677  683  691  701  709  719  727  733  739
743  751  757  761  769  773  787  797  809  811  821  823  82
7  829  839  853  857  859  863  877  881  883  887  907  911
919  929  937  941  947  953  967  971  977  983  991  997

>>>
```

iii)

Code:

```
def print_prime(n):
    count=0
    y= int(n/2)
    for i in range(2,y+1):
        if n%i==0:
            count=count+1
    if count==0 and n!=1:
        print("Given number
is prime")
    else:
        print("Given number
is not prime")
num=int(input("Enter the
number="))
print_prime(num)
```

Output:

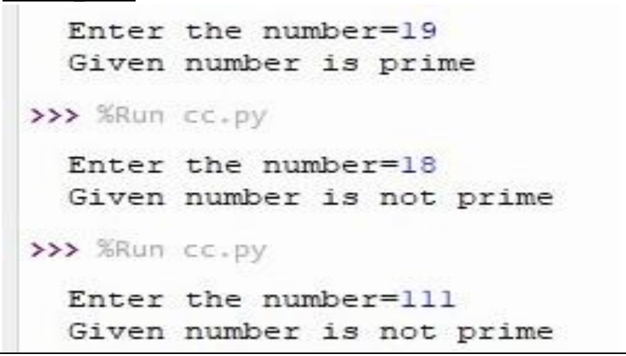


```
 Enter the number=19
 Given number is prime
>>> %Run cc.py
 Enter the number=8
 Given number is not prime
>>> %Run cc.py
 Enter the number=199
 Given number is prime
```

Code:

```
def print_prime(n):
    count=0
    y=int(n**1/2)
    for i in range(2,y+1):
        if n%i==0:
            count=count+1
    if count==0 and n!=1:
        print("Given number is
prime")
    else:
        print("Given number is
not prime")
num=int(input("Enter the
number="))
print_prime(num)
```

Output:



```
 Enter the number=19
 Given number is prime
>>> %Run cc.py
 Enter the number=18
 Given number is not prime
>>> %Run cc.py
 Enter the number=111
 Given number is not prime
```

Task 2:

Implement the following function:

a) *factors (n)*–displays all factors of the number *‘n’* passed as parameter.

b) *display (start, end)* –this function takes the start and end of a range as parameter and calls the above function *factors ()* to display the factors of all numbers in that range.
Now call the function *display ()* in a program. Take the start and end of a range from the user and pass it to the function to display the factors.

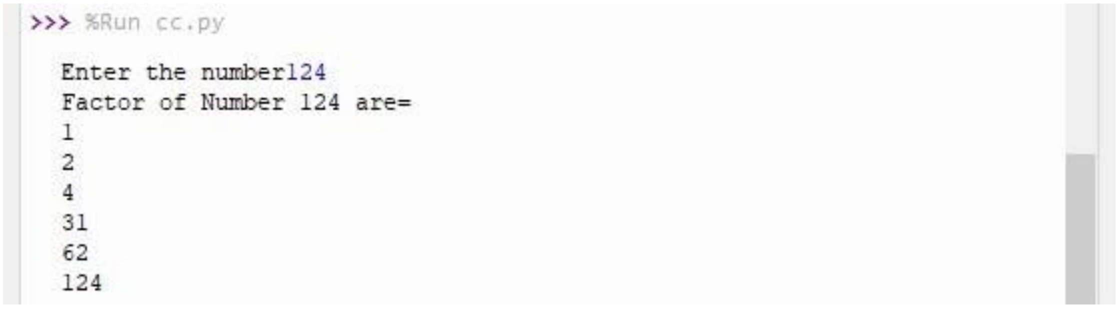
a)

Code:

```
def print_factor(n):

    print("Factor of Number",n,"are=")

    for i in range(1,n+1):

        if n%i==0:

            print(i)
```

```
num=int(input("Enter the number"))

print_factor(num)
```

Output:



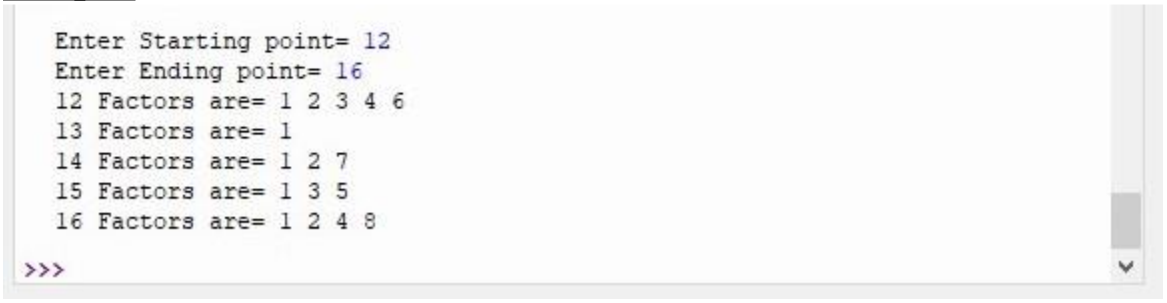
```
>>> %Run cc.py

  Enter the number124
  Factor of Number 124 are=
  1
  2
  4
  31
  62
  124
```

b)

Code:

```
def factors(x):

    for n in range (1,x):

        if x % n == 0:

            print(n,end = ' ')

def display(start,end):

    for n in range(start,end+1):

        print(n,'Factors are=',end = ' ')

        factors(n)

        print()

start = int(input('Enter Starting point= '))

end = int(input('Enter Ending point= '))

display(start,end)
```

Output:



```
  Enter Starting point= 12
  Enter Ending point= 16
  12 Factors are= 1 2 3 4 6
  13 Factors are= 1
  14 Factors are= 1 2 7
  15 Factors are= 1 3 5
  16 Factors are= 1 2 4 8

>>>
```

Conclusion:

Today I learn how to determine the prime number. And learn how to find out the factors of given numbers.

NAME	Muhammad Arslan Raza
ROLL#	2020-EE-403

Lab 9: Problem Set 3

Objective: The objective of this problem set is how to write different programs on compiler

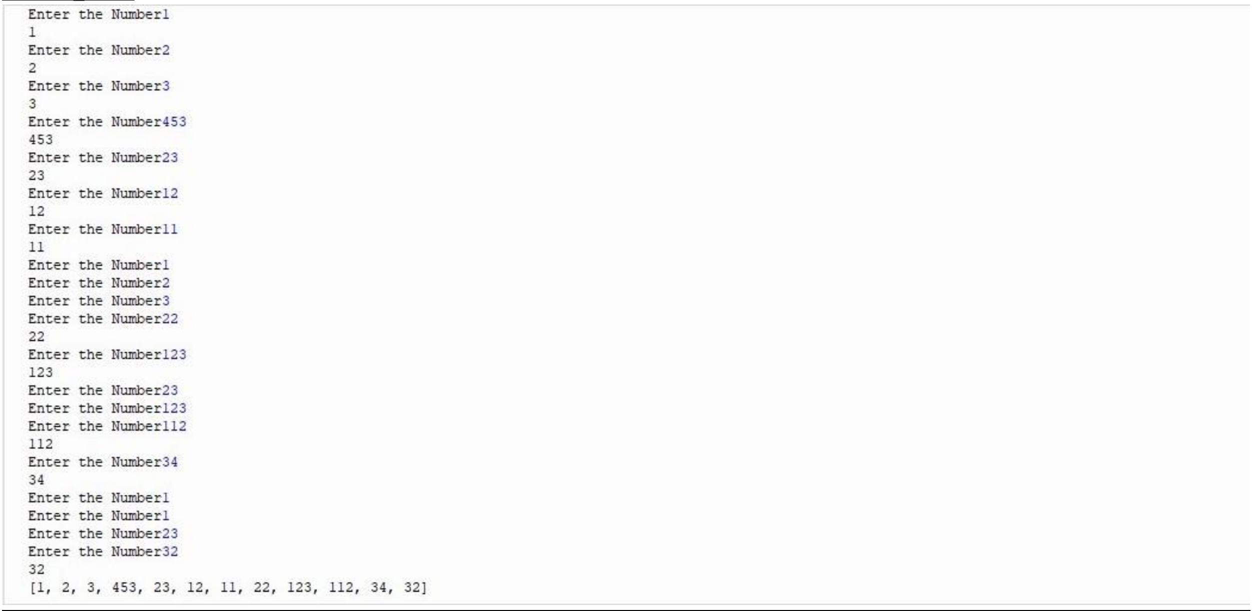
Task 1:

Use a list to solve the following problem: Read in 20 numbers. As each number is read, print it only if it is not a duplicate of a number already read.

Code:

```
L=[ ]
for i in range(20):
    x=int(input("Enter the Number"))
    if x not in L:
        L=L+[x]
        print(x)
print(L)
```

Output:



```
Enter the Number1
1
Enter the Number2
2
Enter the Number3
3
Enter the Number453
453
Enter the Number23
23
Enter the Number12
12
Enter the Number11
11
Enter the Number1
Enter the Number2
Enter the Number3
Enter the Number22
22
Enter the Number123
123
Enter the Number23
Enter the Number123
Enter the Number112
112
Enter the Number34
34
Enter the Number1
Enter the Number1
Enter the Number23
Enter the Number32
32
[1, 2, 3, 453, 23, 12, 11, 22, 123, 112, 34, 32]
```

Task 3:

A prime integer is any integer greater than 1 that is evenly divisible only by itself and 1. The Sieve of Eratosthenes is a method of finding prime numbers. It operates as follows:

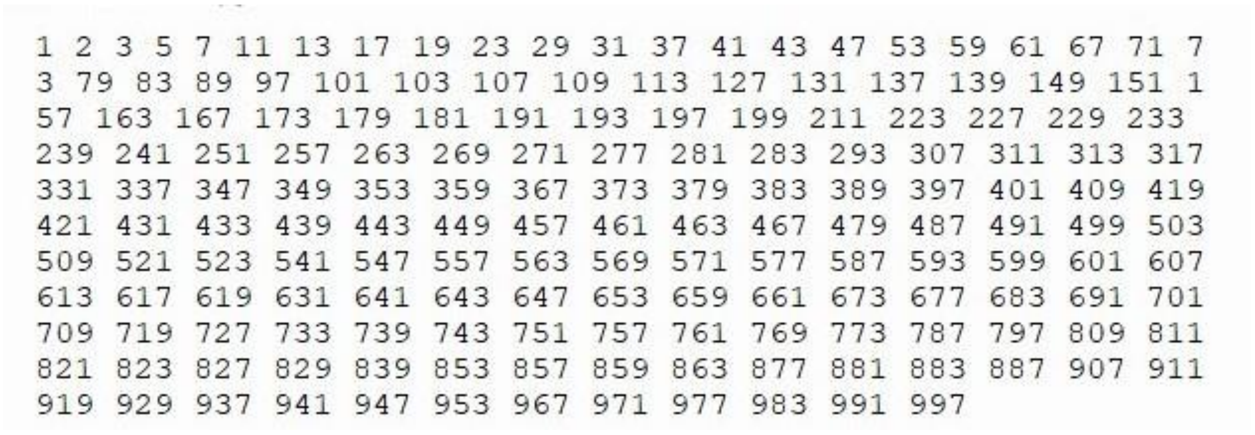
a) Create a list with all elements initialized to 1 (true). List elements with prime subscripts will remain 1. All other list elements will eventually be set to zero.
b) Starting with list element 2, every time a list element is found whose value is 1, loop through

the remainder of the list and set to zero every element whose subscript is a multiple of the subscript for the element with value 1. For list subscript 2, all elements beyond 2 in the list that are multiples of 2 will be set to zero (subscripts 4, 6, 8, 10, etc.); for list subscript 3, all elements beyond 3 in the list that are multiples of 3 will be set to zero (subscripts 6, 9, 12, 15, etc.); and so on.
When this process is complete, the list elements that are still set to 1 indicate that the subscript is a prime number. These subscripts can then be printed. Write a program that uses a list of 1000 elements to determine and print the prime numbers between 2 and 999. Ignore element 0 of the list

Code:

```
list=[]
for n in range (1000):
    list.append(1)
for j in range (2,1000):
    if list[j]==1:
        for m in range (j+1,1000):
            if m%j==0:
                list[m]=0
for k in range (1,1000):
    if list[k]==1:
        print(k,end =" ")
```

Output:



```
1 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 7
3 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 1
57 163 167 173 179 181 191 193 197 199 211 223 227 229 233
239 241 251 257 263 269 271 277 281 283 293 307 311 313 317
331 337 347 349 353 359 367 373 379 383 389 397 401 409 419
421 431 433 439 443 449 457 461 463 467 479 487 491 499 503
509 521 523 541 547 557 563 569 571 577 587 593 599 601 607
613 617 619 631 641 643 647 653 659 661 673 677 683 691 701
709 719 727 733 739 743 751 757 761 769 773 787 797 809 811
821 823 827 829 839 853 857 859 863 877 881 883 887 907 911
919 929 937 941 947 953 967 971 977 983 991 997
```

Conclusion:

Today In this lab I learn how to use a list to solve the different problems in python.

NAME	Muhammad Arslan Raza
ROLL#	2020-EE-403

Lab 11: Problem Set 4

Objective:

The objective of this problem set is how to write different programs on compiler

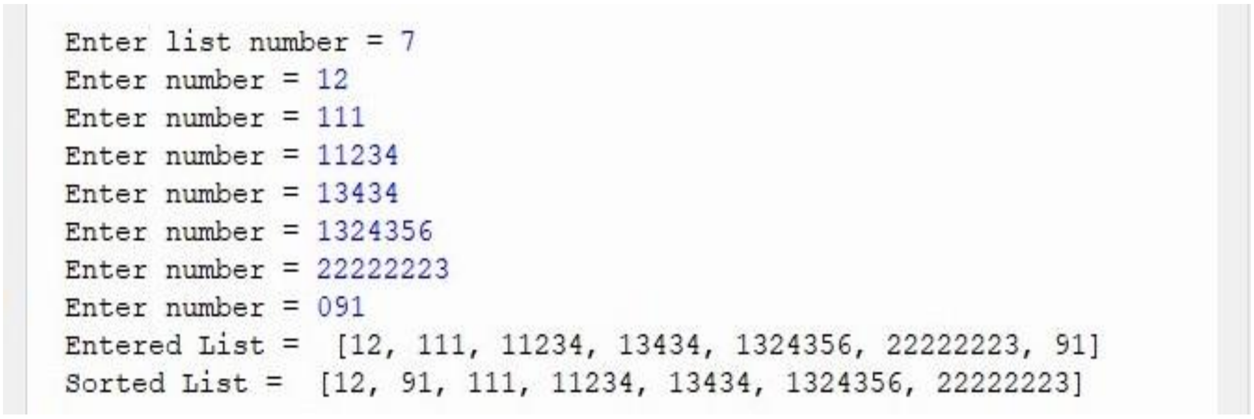
Task 1:

Sorting data (i.e., placing data into some particular order, such as ascending or descending) is one of the most important computing applications. Python lists provide a sort method. In this exercise, readers implement their own sorting function, using the bubble-sort method. In the bubble sort (or sinking sort), the smaller values gradually “bubble” their way upward to the top of the list like air bubbles rising in water, while the larger values sink to the bottom of the list. The process that compares each adjacent pair of elements in a list in turn and swaps the elements if the second element is less than the first element is called a pass. The technique makes several passes through the list. On each pass, successive pairs of elements are compared. If a pair is in increasing order, bubble sort leaves the values as they are. If a pair is in decreasing order, their values are swapped in the list. After the first pass, the largest value is guaranteed to sink to the highest index of a list. After the second pass, the second largest value is guaranteed to sink to the second highest index of a list, and so on. Write a program that uses function bubble Sort to sort the items in a list.

Code:

```
def bubbleSort(w):
    result=True
    while result:
        result=False
        for i in range (len(w)-1):
            if w[i]>w[i+1]:
                w[i],w[i+1]=w[i+1],w[i]
                result = True
    return w
l=int(input("Enter list number = "))
s=[]
for r in range (l):
    x=int(input("Enter number = "))
    s.append(x)
print ("Entered List = ",s)
print ("Sorted List = ",bubbleSort(s))
```

Output:



```
Enter list number = 7
Enter number = 12
Enter number = 111
Enter number = 11234
Enter number = 13434
Enter number = 1324356
Enter number = 22222223
Enter number = 091
Entered List =  [12, 111, 11234, 13434, 1324356, 22222223, 91]
Sorted List =  [12, 91, 111, 11234, 13434, 1324356, 22222223]
```

Conclusion:

Today in this lab I learn how to sort a given list with the use of bubbleSort function.

NAME	Muhammad Arslan Raza
ROLL#	2020-EE-403

Lab 12: Open Ended Lab (Searching & Sorting)

Objective:

The objective of this problem set is how to write different programs on compiler

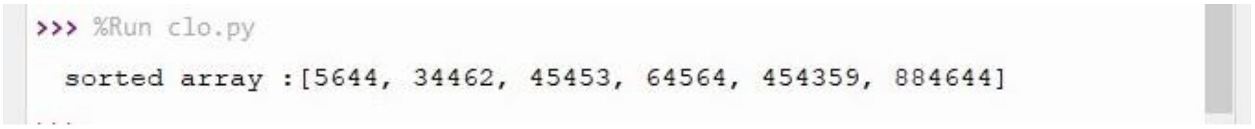
Task 1:

Write a Python program to sort a list of elements using the insertion sort algorithm.

Code:

```
def insertion_sort(l):
    for i in range(1,len(l)):
        value=l[i]
        index=i
        while index>0 and l[index-1]>value:
            l[index]=l[index-1]
            index=index-1
            l[index]=value
l=[45453,884644,454359,64564,34462,5644]
insertion_sort(l)
print("sorted array :" + str(l))
```

Output:



```
>>> %Run clo.py
  sorted array :[5644, 34462, 45453, 64564, 454359, 884644]
```

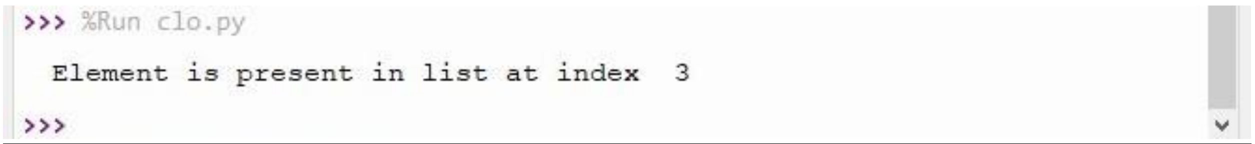
Task 2:

Write a Python program for binary search as it shows that specific element is present in list or not.

Code:

```
def binary_search (array,left,right,x ):
    if right >= left:
        mid=(right+left)//2
        if array[mid]==x:
            return mid
        elif array[mid]>x:
            return binary_search(array,left,mid-1,x)
        else:
            return binary_search(array,mid+1,right,x)
    else:
        return -1
array=[22,5,4,10,7]
x=10
result=binary_search(array,0,len(array)-1,x)
if result != -1 :
    print ("Element is present in list at index ", str(result))
else:
    print ("Element is not in the list")
```

Output:



```
>>> %Run clo.py
  Element is present in list at index  3
>>>
```

Conclusion:

Today in this lab I learn how to sort a list of elements using the insertion sort algorithm. And write a program for binary search as it shows that specific element is present in list or not.

Name:	Muhammad Arslan Raza
Roll no.	2020-EE-403

Lab # 13: Classes and Object-Oriented Programming

Objective:

The objective of this problem set is how to write different programs on compiler

Task # 1:

Create a class called RationalNumber for performing arithmetic with fractions. Write a driver program to test your class Use integer variables to represent the data of the class—the numerator and the denominator. Provide a constructor that enables an object of this class to be initialized when it is declared. The constructor should contain default values, in case no initializers are provided, and should store the fraction in reduced form (i.e., the fraction

2/4

would be stored in the object as 1 in the numerator and 2 in the denominator).

Provide methods for each of the following:

a) Adding two RationalNumbers. The result should be stored in reduced form.

b) Subtracting two RationalNumbers. The result should be stored in reduced form.

c) Multiplying two RationalNumbers. The result should be stored in reduced form.

d) Dividing two RationalNumbers. The result should be stored in reduced form.

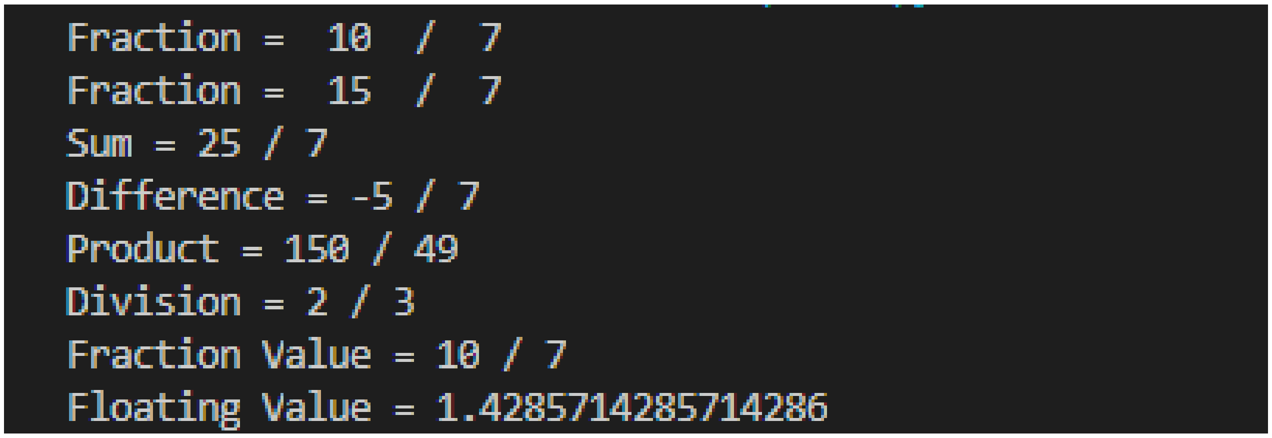
e) Printing RationalNumbers in the form a/b, where a is the numerator and b is the denominator.

f) Printing RationalNumbers in floating-point format.

Code:

```
#Creating Objects

class Rational():

    def __init__(self,n,d):

        self.n = n

        self.d = d

        print('Fraction = ',self.n," / ",self.d)

    def addition(self,other):

        sn = (self.n*(other.d) + other.n*(self.d))

        sd = (self.d * other.d)

        r = reduce(sn,sd)

        return "Sum = " + str(r[0]) + " / " + str(r[1])

    def subtraction(self,other):

        sn = (self.n*(other.d) - other.n*(self.d))

        sd = (self.d * other.d)

        r = reduce(sn,sd)

        return "Difference = " + str(r[0]) + " / " + str(r[1])

    def multiplication(self,other):

        sn = self.n * other.n

        sd = self.d * other.d

        r = reduce(sn,sd)

        return "Product = " + str(r[0]) + " / " + str(r[1])

    def division(self,other):

        sn = self.n*(other.d)

        sd = self.d * other.n

        r = reduce(sn,sd)
        return "Division = " + str(r[0]) + " / " + str(r[1])

    def get_frac(self):

        return "Fraction Value = " + str(self.n) + ' / ' + str(self.d)

    def get_float(self):

        return "Floating Value = " + str(self.n/self.d)

def reduce(n,d):

    if n < d:

        for k in range(abs(n),0,-1):

            if n % k == 0 and d % k == 0:

                n = int(n/k)

                d = int(d/k)

    else:

        for k in range(abs(d),0,-1):

            if n % k == 0 and d % k == 0:

                n = int(n/k)

                d = int(d/k)

    return(n,d)


#Driver Code

frac1 = Rational(10,7)

frac2 = Rational(15,7)

print(frac1.addition(frac2))

print(frac1.subtraction(frac2))

print(frac1.multiplication(frac2))

print(frac1.division(frac2))

print(frac1.get_frac())

print(frac1.get_float())
```

Output:



```
Fraction =  10  /  7
Fraction =  15  /  7
Sum = 25 / 7
Difference = -5 / 7
Product = 150 / 49
Division = 2 / 3
Fraction Value = 10 / 7
Floating Value = 1.4285714285714286
```

Task # 2:

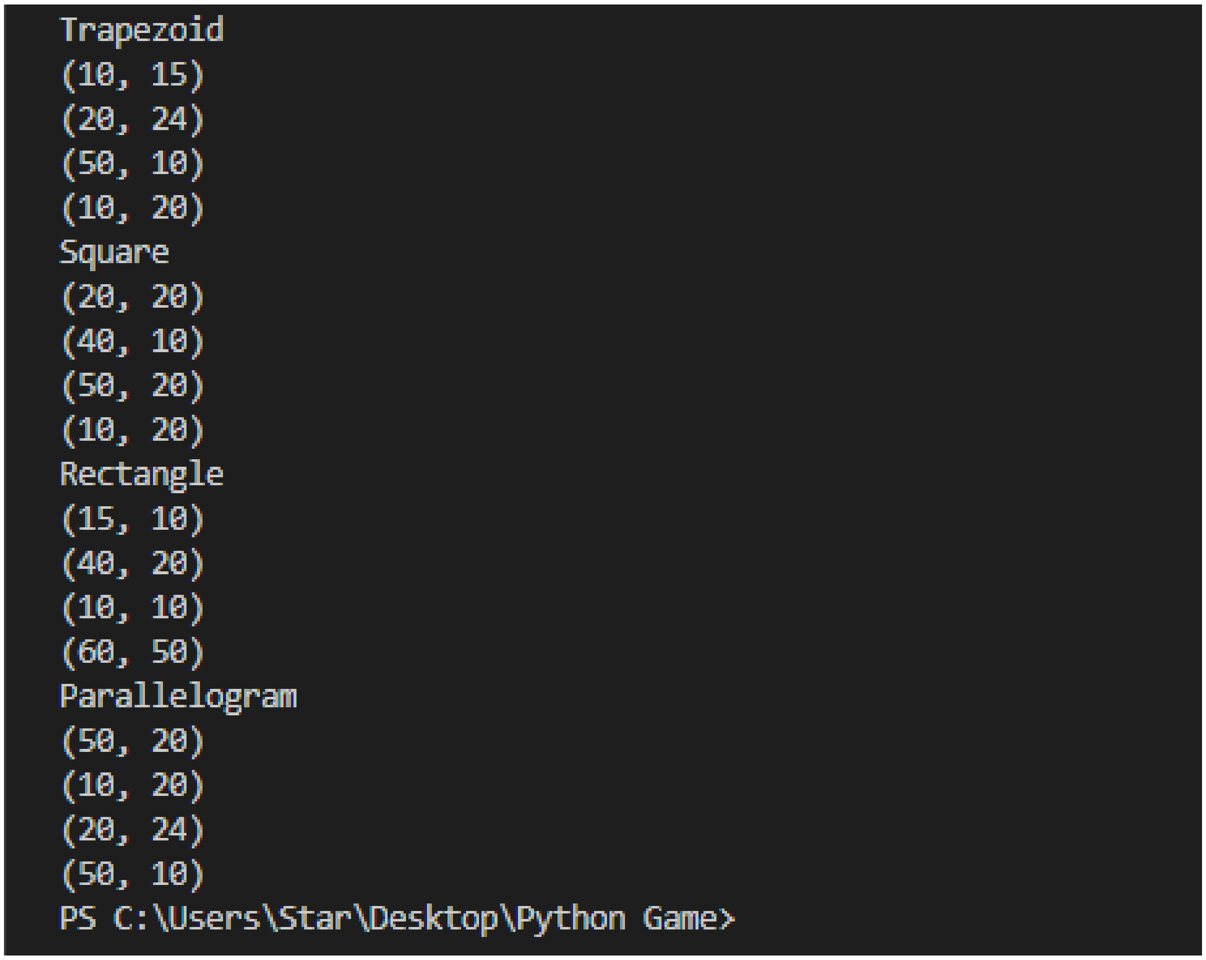
Write an inheritance hierarchy for class Quadrilateral, Trapezoid, Parallelogram, Rectangle and Square. Use Quadrilateral as the base class of the hierarchy. Make the hierarchy as deep (i.e., as many levels) as possible. The data of Quadrilateral should be the (x, y) coordinate pairs for the four endpoints of the Quadrilateral. Write a driver program that creates and displays objects of each of these classes.

Code:

```
#Parent class

class Quadilateral(object):

    def __init__(self,p1,p2,p3,p4):

        self.p1 = p1

        self.p2 = p2

        self.p3 = p3

        self.p4 = p4


    def __str__(self):

        s = str(self.p1) + "\n" + str(self.p2) + "\n"  +
str(self.p3) + "\n" + str(self.p4)
```

```
        return s

#Child classes
class Trapezoid(Quadilateral):
    pass
class Square(Quadilateral):
    pass
class Rectangle(Quadilateral):
    pass
class Parallelogram(Quadilateral):
    pass



#Driver code
trapezoid = Trapezoid((10,15),(20,24),(50,10),(10,20))
sqruare = Square((20,20),(40,10),(50,20),(10,20))
rectangle = Rectangle((15,10),(40,20),(10,10),(60,50))
parallelogram = Parallelogram((50,20),(10,20),(20,24),(50,10))

print("Trapezoid \n",trapezoid)
print("Square \n",sqr)
print("Rectangle \n",rectangle)
print("Parallelogram \n",parallelogram)
```

Output:



```
Trapezoid
(10, 15)
(20, 24)
(50, 10)
(10, 20)
Square
(20, 20)
(40, 10)
(50, 20)
(10, 20)
Rectangle
(15, 10)
(40, 20)
(10, 10)
(60, 50)
Parallelogram
(50, 20)
(10, 20)
(20, 24)
(50, 10)
PS C:\Users\Star\Desktop\Python Game>
```

Conclusion:

. Today in this lab I learn about Classes and Object-Oriented Programming.