

Batch Name: Summer Internship 2018

Enrolment No: R610217007

SAPID: 500062812

Name: DIVYANSHU SINGH

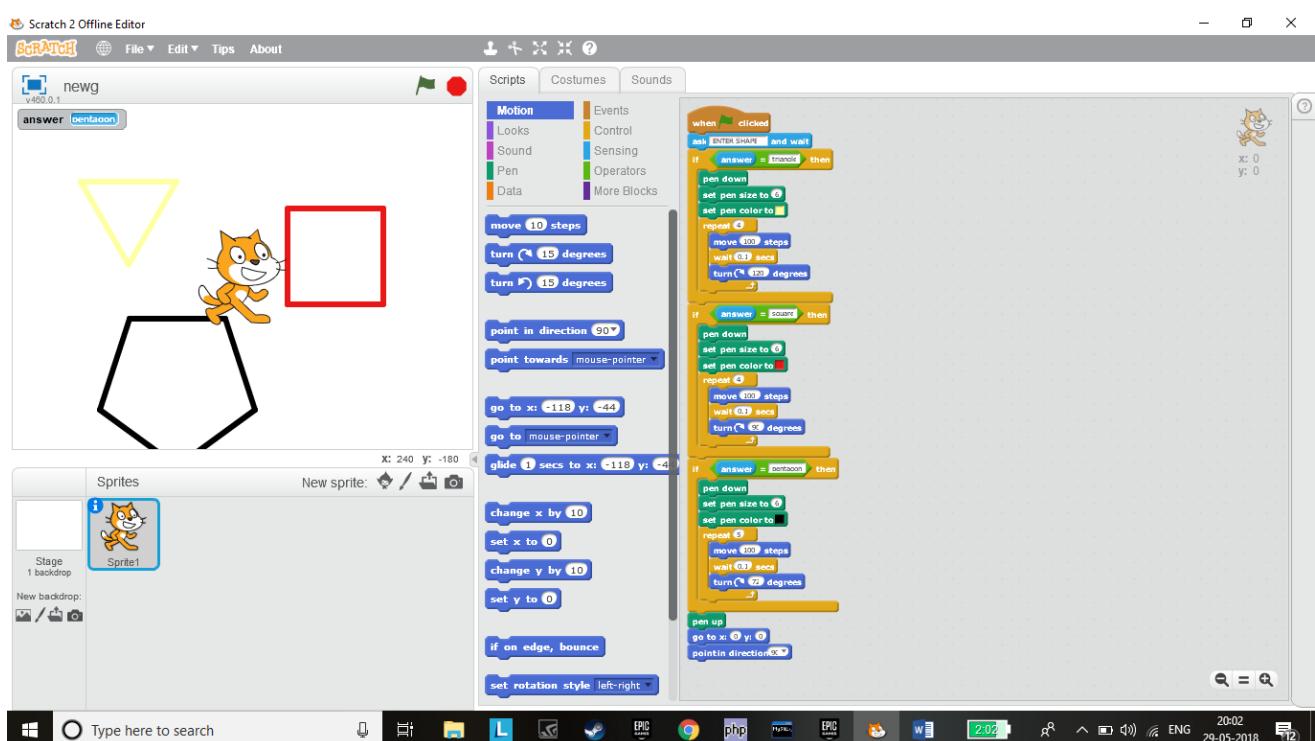
Sem: II

Branch: CSE -MAINFRAME TECHNOLOGY

FP5.0 Module-1 Assignments 1

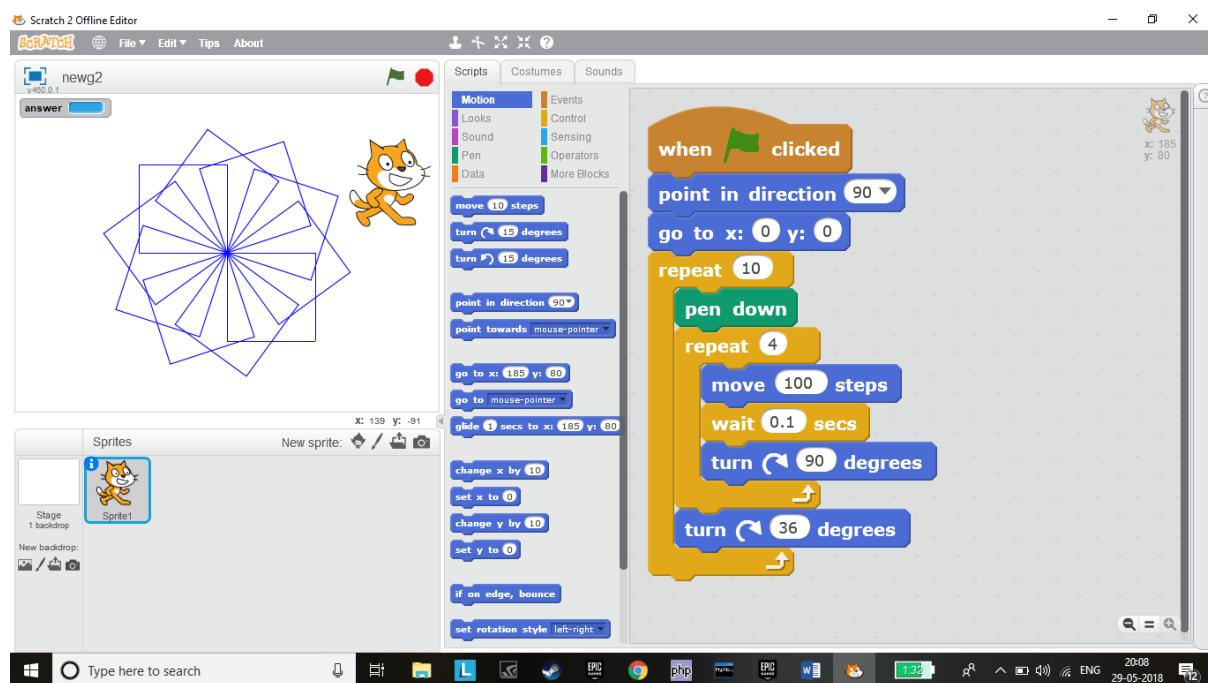
#Create the following using Scratch:

QUES1-Take a text input from the user as one of the three shape names – "square", "triangle" or "pentagon". Based on the input, draw either a red square, yellow triangle or black pentagon.

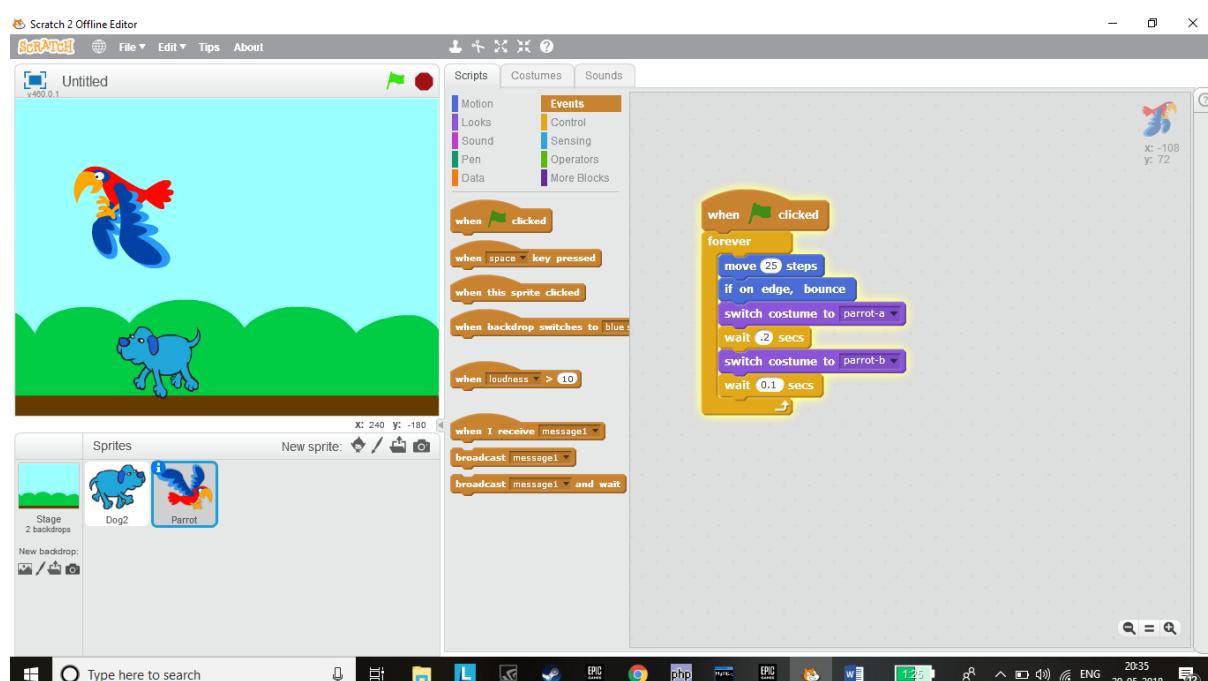
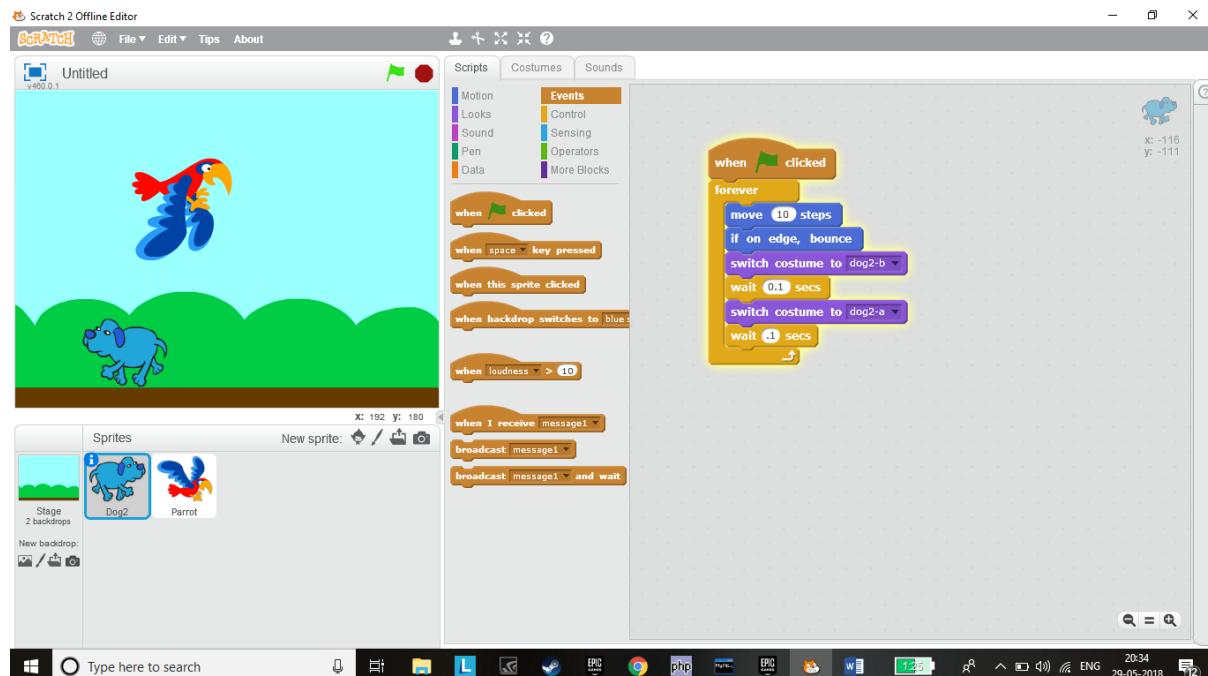


QUES 2. Create the following pattern using Scratch.

Hint: Create 10 squares each with an inclination of 36 degrees from the preceding square



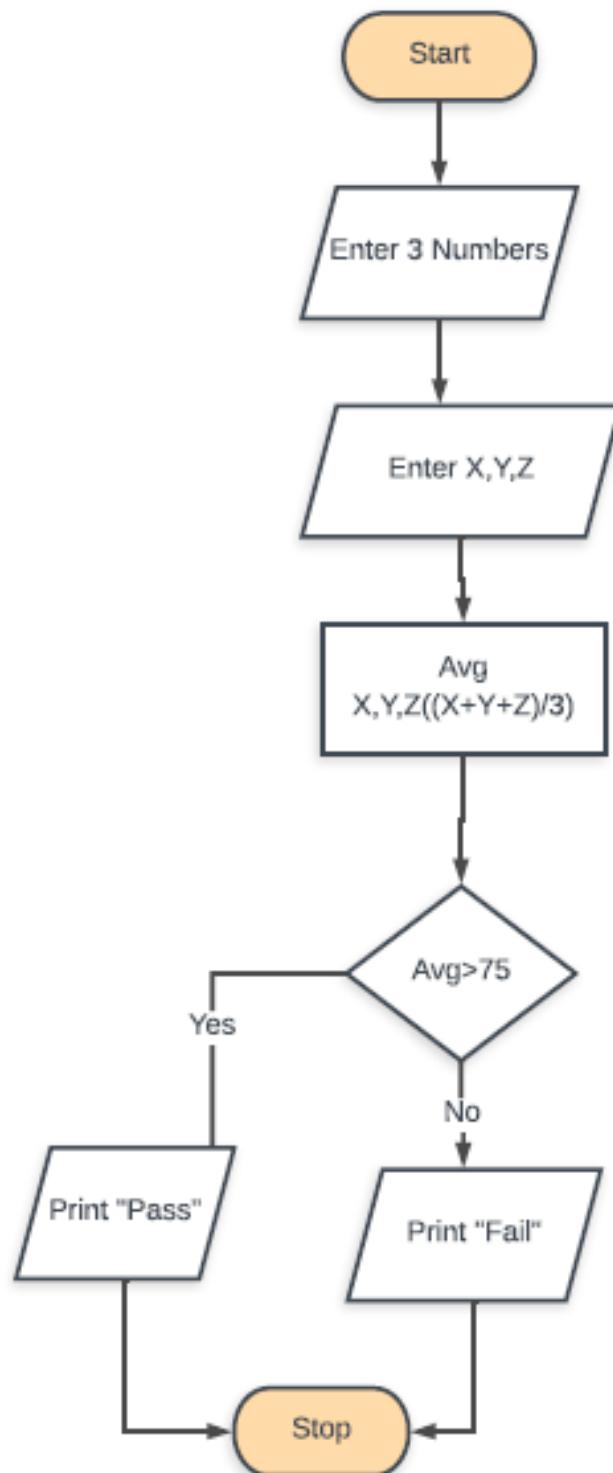
QUES 3. Create or import two Sprites and use your imagination to make them do different actions simultaneously. For example: "A bird is flying and a dog is walking at the same time."



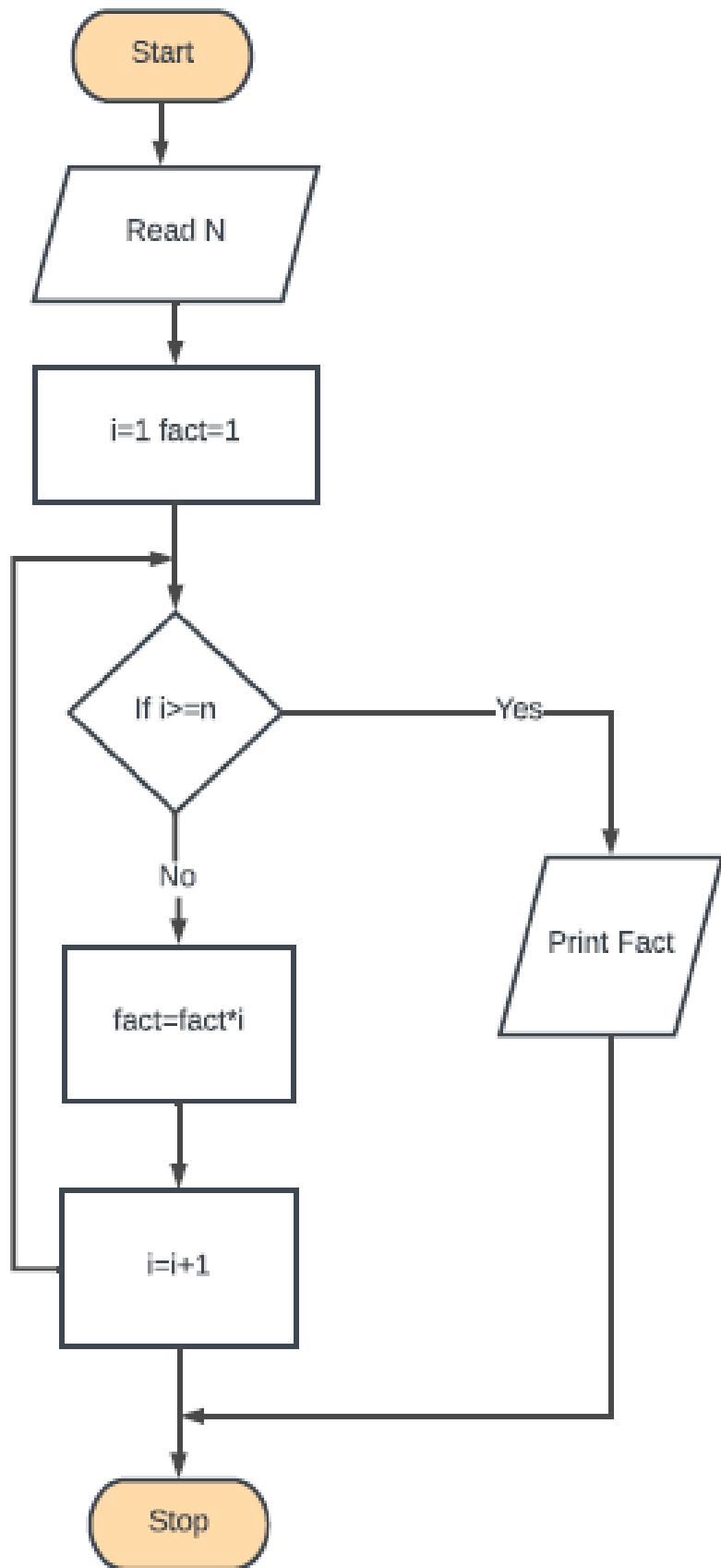
FP5.0 Module-1 Assignments 2

Create flowcharts for the following problems:

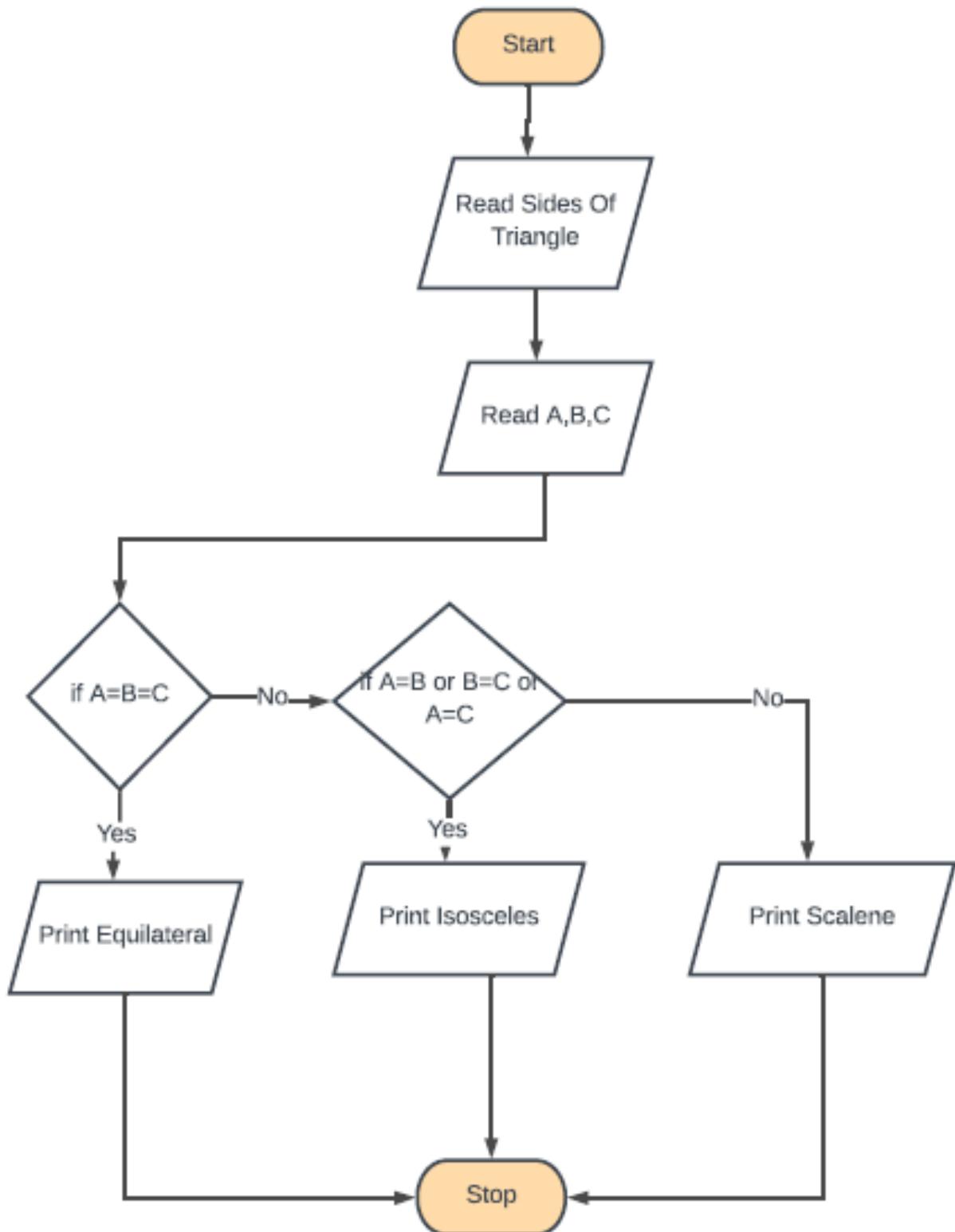
1. Calculate the average of three numbers. If average is greater than or equal to 75, print "Pass", else print "Fail".



2. Calculate and print the factorial of a number.



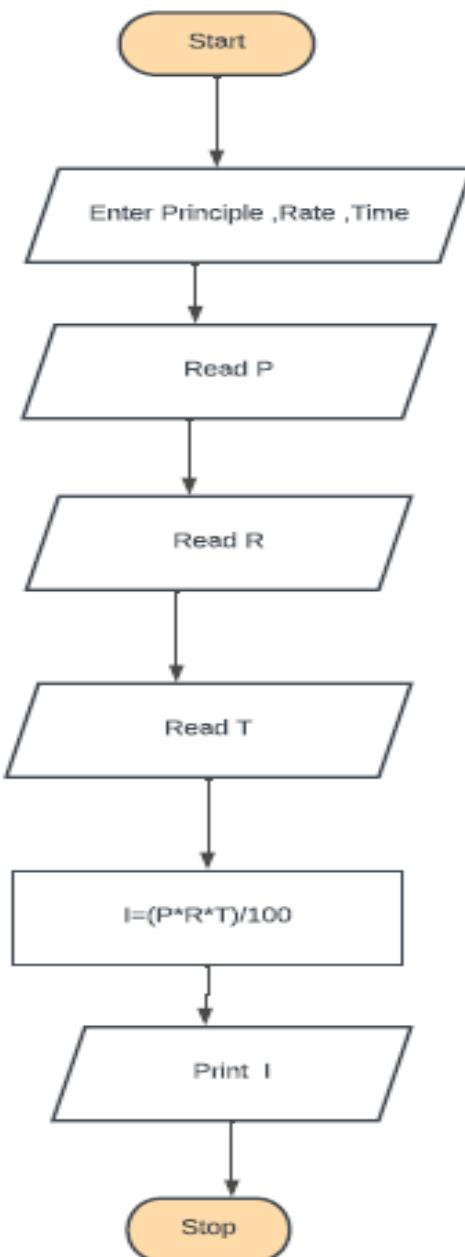
3. Accept the lengths of three sides of a triangle as input from the user. Based on the input, print if the given triangle is "Equilateral", "Isosceles" or "Scalene".



QUES4.Accept the values of principal amount, rate of interest and number of years as an input from the user. Calculate and print the simple interest.

Hint - Formula for simple interest:

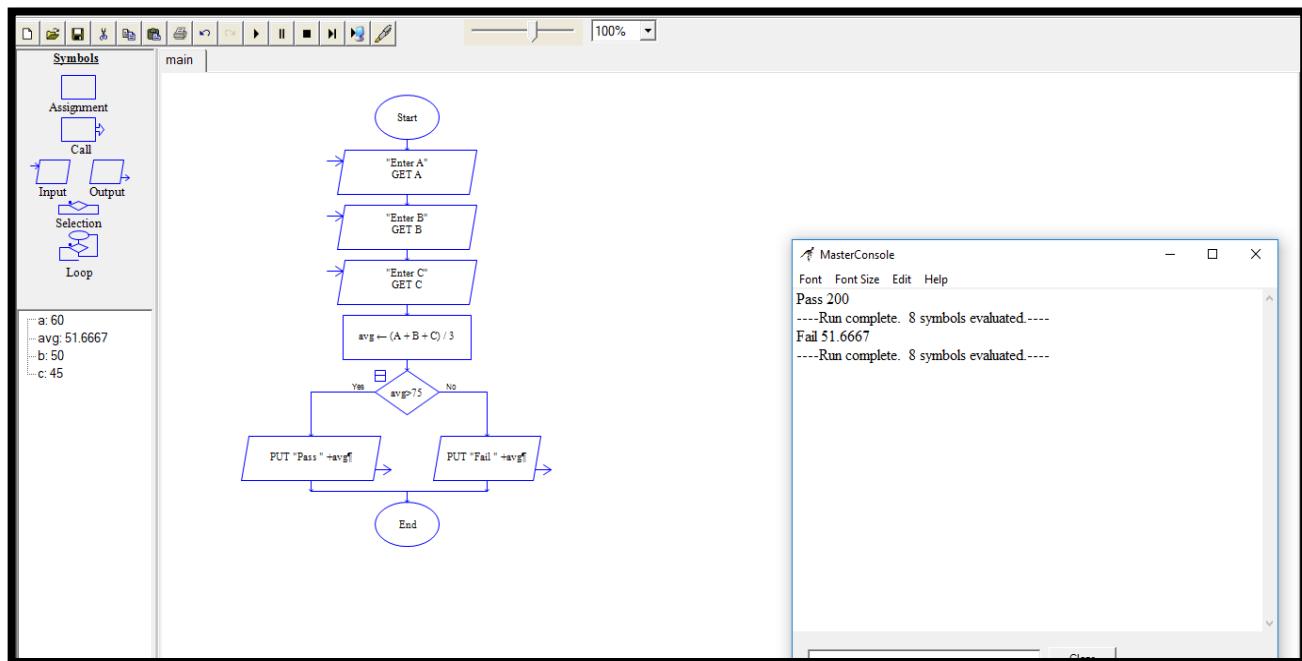
Simple Interest = (principal amount * rate of interest * number of years)/100



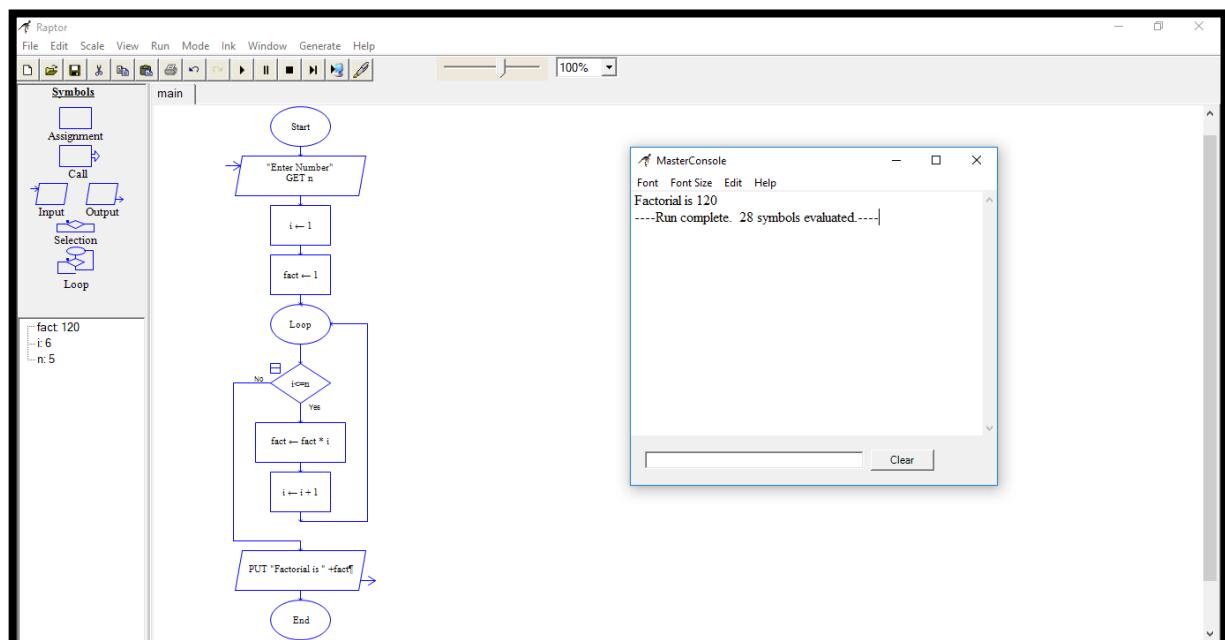
FP5.0 Module-1 Assignments 3

#Create flowcharts for the following problems:

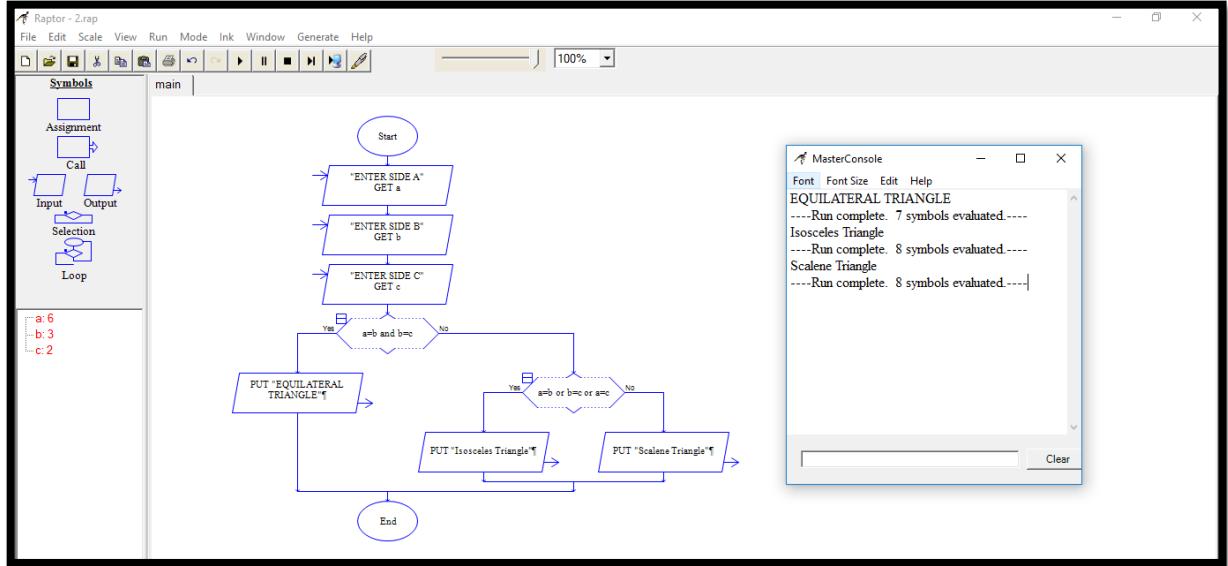
QUES1-Calculate the average of three numbers. If average is greater than or equal to 75, print "Pass", else print "Fail".



QUES2.Calculate and print the factorial of a number.

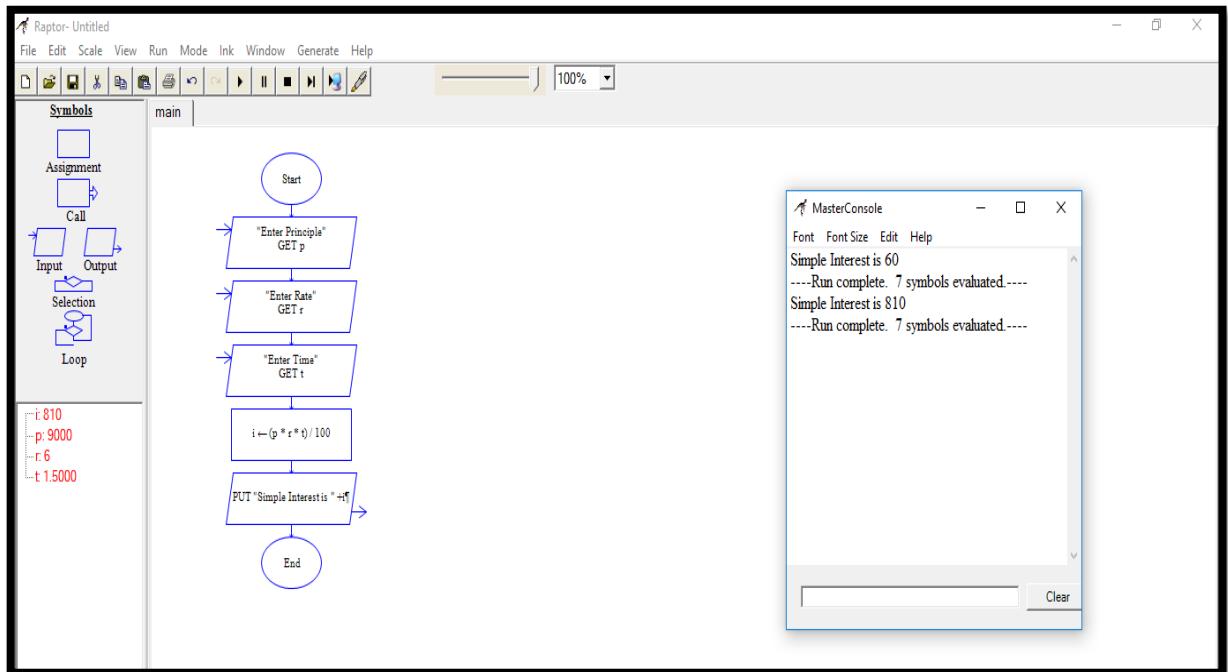


QUES3.Accept the lengths of three sides of a triangle as input from the user. Based on the input, print if the given triangle is "Equilateral", "Isosceles" or "Scalene".



QUES4.Accept the values of principal amount, rate of interest and number of years as an input from the user. Calculate and print the simple interest.

Hint - Formula for simple interest:Simple Interest = (principal amount * rate of interest * number of years)/100



FP5.0 Module-1 Assignments 4

Write Pseudo Code:

QUES1. To check whether a given number is even or odd.

```
1. START  
2. DISPLAY "Enter the Number - "  
3. READ number  
4. IF number MOD 2 = 0 THEN  
    DISPLAY "Number is Even"  
ELSE  
    DISPLAY "Number is Odd"  
END IF  
5. STOP
```

2. To print the multiples of 3 between 1 to 20.

```
1. START  
2. number = 1  
3. LOOP WHILE number<=20  
4. IF number MOD 3 = 0 THEN  
    DISPLAY number  
END IF  
5. number++  
6. END LOOP  
7. STOP
```

3. To find factorial of a given number.

```
1. START  
2. DISPLAY "Enter the Number - "  
3. READ number  
4. fact = 1  
5. LOOP FOR i<=number  
6. fact = fact * i  
7. i++  
END LOOP  
8. DISPLAY fact  
9. STOP
```

4. To calculate 'x' to the power of 'n' using a while loop.

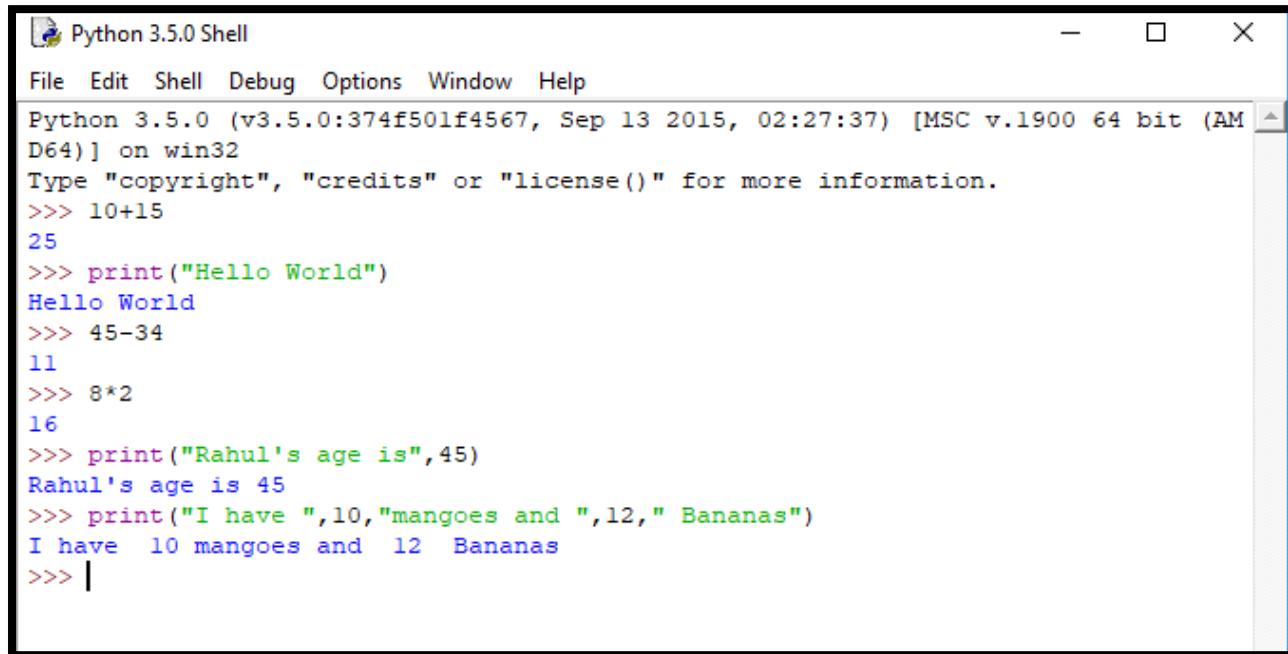
•Assume both 'x' and 'n' are positive whole numbers

1. START
2. DISPLAY "Enter the Number - "
3. READ number
4. DISPLAY "Enter Power"
5. READ power
6. result = 1
7. LOOP FOR power > 0
8. result = result * number
9. power--
- END LOOP
- 10.DISPLAY result
11. STOP

FP5.0 Module-1 Assignments 5

Open the Python IDLE and execute the following commands. Observe the output.

```
1)10 + 15  
2)print("Hello World")  
3)45 - 34  
4)8 * 2  
5)print("Rahul's age is", 45)  
6)print("I have", 10, "mangoes and", 12, "bananas")
```



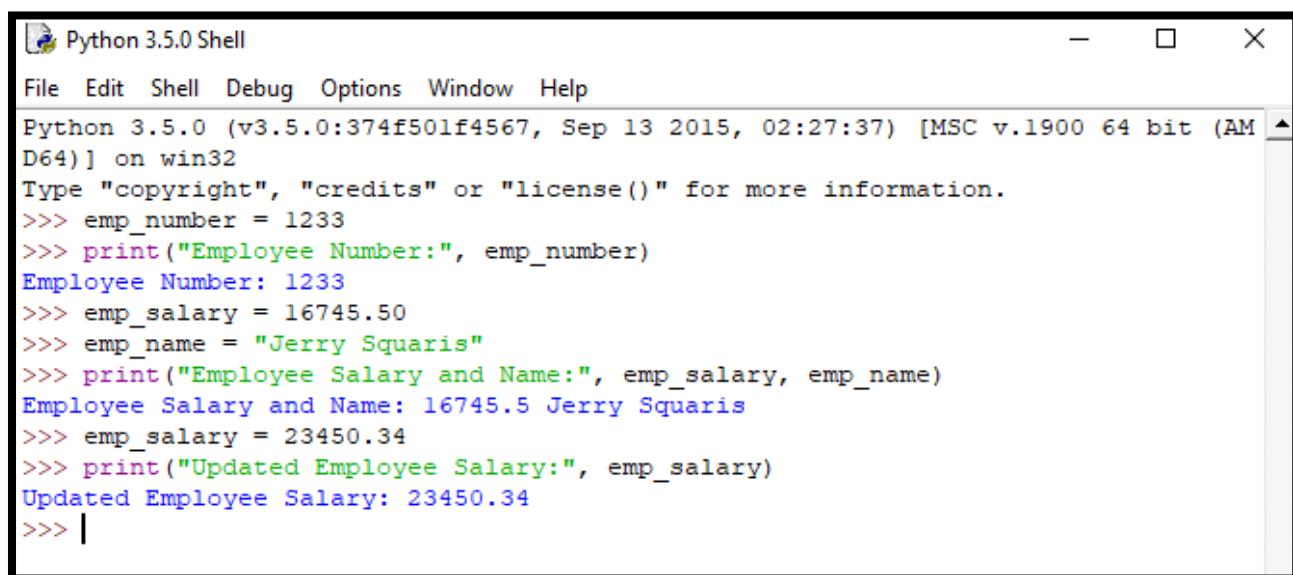
The screenshot shows the Python 3.5.0 Shell window. The title bar reads "Python 3.5.0 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the Python interpreter's prompt (>>>) followed by the execution of various commands and their results:

```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AM  
D64)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>> 10+15  
25  
>>> print("Hello World")  
Hello World  
>>> 45-34  
11  
>>> 8*2  
16  
>>> print("Rahul's age is",45)  
Rahul's age is 45  
>>> print("I have ",10,"mangoes and ",12," Bananas")  
I have 10 mangoes and 12 Bananas  
>>> |
```

FP5.0 Module-1 Assignments 6

Open Python IDLE and execute the following commands. Observe the output.

```
1.emp_number = 1233
2.print("Employee Number:", emp_number)
3.emp_salary = 16745.50
4.emp_name = "Jerry Squaris"
5.print("Employee Salary and Name:", emp_salary, emp_name)
6.emp_salary = 23450.34
7.print("Updated Employee Salary:", emp_salary)
```



The screenshot shows the Python 3.5.0 Shell window. The title bar says "Python 3.5.0 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the following text:

```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> emp_number = 1233
>>> print("Employee Number:", emp_number)
Employee Number: 1233
>>> emp_salary = 16745.50
>>> emp_name = "Jerry Squaris"
>>> print("Employee Salary and Name:", emp_salary, emp_name)
Employee Salary and Name: 16745.5 Jerry Squaris
>>> emp_salary = 23450.34
>>> print("Updated Employee Salary:", emp_salary)
Updated Employee Salary: 23450.34
>>> |
```

FP5.0 Module-1 Assignments 7

Execute the following Python statements in IDLE and observe the output:

- customer_id = 101
- type(customer_id)
- customer_name = "John"
- type(customer_name)
- bill_amount = 675.45
- type(bill_amoun)
- x = 5.3 + 0.9j
- type(x)
- print(customer_id, customer_name, bill_amount)
- print(x.real)
- print(x.imag + 3)
- Flag = True
- type(Flag)

```
•y = "False"
```

```
•type(y)
```

```
>>> customer_id = 101
>>> type(customer_id)
<class 'int'>
>>> customer_name = "John"
>>> type(customer_name)
<class 'str'>
>>> bill_amount = 675.45
>>> type(bill_amount )
<class 'float'>
>>> x = 5.3 + 0.9j
>>> type(x)
<class 'complex'>
>>> print(customer_id, customer_name, bill_amount)
101 John 675.45
>>> print(x.real)
5.3
>>> print(x.imag + 3)
3.9
>>> Flag = True
>>> type(Flag)
<class 'bool'>
>>> y = "False"
>>> type(y)
<class 'str'>
>>> |
```

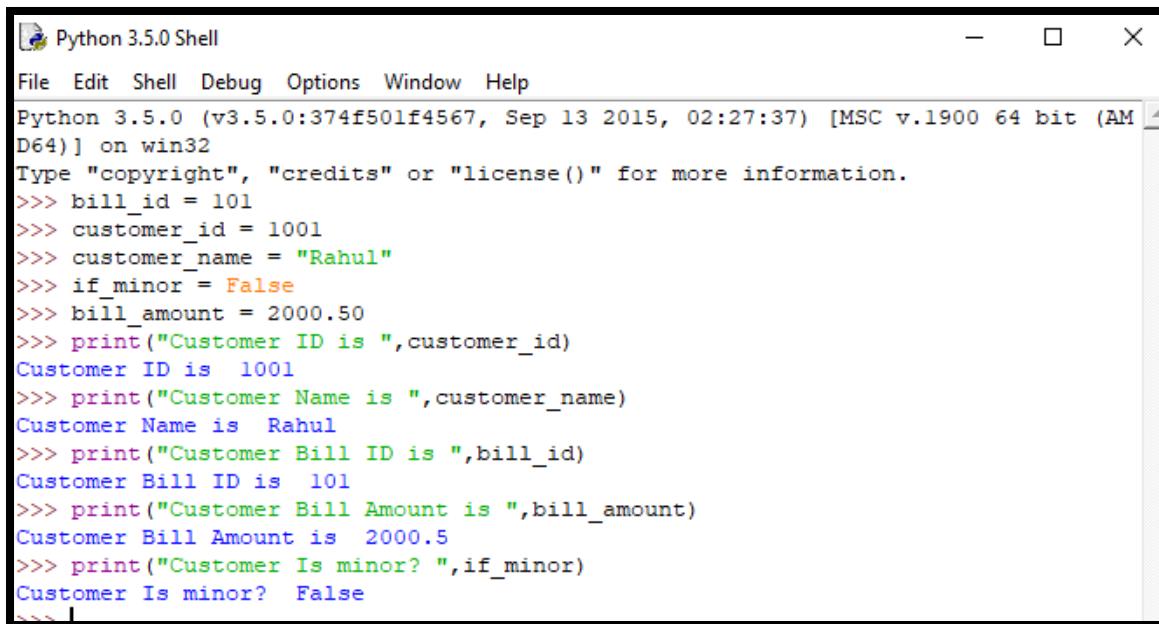
FP5.0 Module-1 Assignments 5

In a retail application, shopkeeper wants to keep a track of following details of a customer. Sample values are provided.

- bill_id = 101
- customer_id = 1001
- customer_name = "Rahul"
- if_minor = False
- bill_amount = 2000.50

Write a python program to store the details and display them.

```
#  
bill_id = 101  
customer_id = 1001  
customer_name = "Rahul"  
if_minor = False  
bill_amount = 2000.50  
print("Customer ID is ",customer_id)  
print("Customer Name is ",customer_name)  
print("Customer Bill ID is ",bill_id)  
print("Customer Bill Amount is ",bill_amount)  
print("Customer is minor?",if_minor)  
#
```



The screenshot shows a Python 3.5.0 Shell window. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The shell displays the following code and its output:

```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AM  
D64)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>> bill_id = 101  
>>> customer_id = 1001  
>>> customer_name = "Rahul"  
>>> if_minor = False  
>>> bill_amount = 2000.50  
>>> print("Customer ID is ",customer_id)  
Customer ID is 1001  
>>> print("Customer Name is ",customer_name)  
Customer Name is Rahul  
>>> print("Customer Bill ID is ",bill_id)  
Customer Bill ID is 101  
>>> print("Customer Bill Amount is ",bill_amount)  
Customer Bill Amount is 2000.5  
>>> print("Customer Is minor? ",if_minor)  
Customer Is minor? False  
>>>
```

FP5.0 Module-1 Assignments 9

Execute the following commands and observe the usage of different types of commenting styles.

```
i = 10 # creates an integer variable. This is a single line comment.
```

```
print("i =", i) # prints 10
```

```
""
```

Below code creates a Boolean variable in Python

(This is a multiple line comment)

```
""
```

```
s = True
```

```
print("s =", s) #prints True, Here, s is a Boolean variable with value True
```

```
"""
```

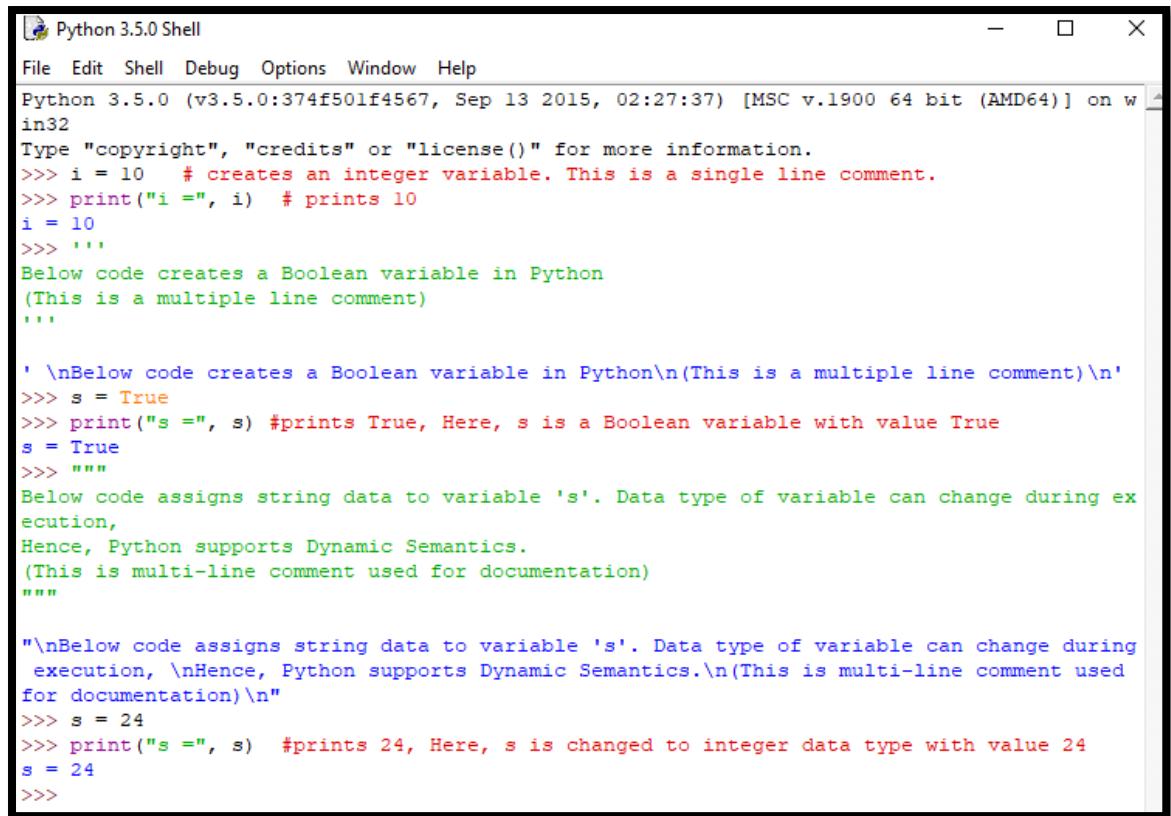
Below code assigns string data to variable 's'. Data type of variable can change during execution, Hence, Python supports Dynamic Semantics.

(This is multi-line comment used for documentation)

```
""
```

```
s = 24
```

```
print("s =", s) #prints 24, Here, s is changed to integer data type with value 24
```



The screenshot shows a Python 3.5.0 Shell window. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The title bar displays "Python 3.5.0 Shell". The main area contains the following Python code:

```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> i = 10    # creates an integer variable. This is a single line comment.
>>> print("i =", i)  # prints 10
i = 10
>>> """
Below code creates a Boolean variable in Python
(This is a multiple line comment)
"""

' \nBelow code creates a Boolean variable in Python\n(This is a multiple line comment)\n'
>>> s = True
>>> print("s =", s) #prints True, Here, s is a Boolean variable with value True
s = True
>>> """
Below code assigns string data to variable 's'. Data type of variable can change during execution,
Hence, Python supports Dynamic Semantics.
(This is multi-line comment used for documentation)
"""

'\nBelow code assigns string data to variable 's'. Data type of variable can change during execution,
\nHence, Python supports Dynamic Semantics.\n(This is multi-line comment used for documentation)\n'
>>> s = 24
>>> print("s =", s)  #prints 24, Here, s is changed to integer data type with value 24
s = 24
>>>
```

FP5.0 Module-1 Assignments 10

Write a Python program for the following requirements:

- Prompt the user to input two numbers num1 and num2
 - Increment num1 by 4 and num2 by 6

- Find and print the sum of new values of num1 and num2

Hint – Use type casting for converting the input into an integer.

```
#

>>> num1=int(input("ENTER NUM 1 \n"))
>>> num2=int(input("ENTER NUM 2 \n"))
>>> num1+=4
>>> num2+=6
>>> print("New Value of NUM 1=%2d \n New Value Of NUM 2=%2d \n Sum Is
%3d"%(num1,num2,num1+num2))
#
```

```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> num1=int(input("ENTER NUM 1 \n"))
ENTER NUM 1
6
>>> num2=int(input("ENTER NUM 2 \n"))
ENTER NUM 2
5
>>> num1+=4
>>> num2+=6
>>> print("New Value of NUM 1=%2d \n New Value Of NUM 2=%2d \n Sum Is %3d"%(num1,num2,num1+num2))
)
New Value of NUM 1=10
New Value Of NUM 2=11
Sum Is 21
>>> type(num1,num2)
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    type(num1,num2)
TypeError: type() takes 1 or 3 arguments
>>> type(num1)
<class 'int'>
>>> type(num2)
<class 'int'>
>>> |
```

FP5.0 Module-1 Assignments 11

1) Consider two variables 'a' and 'b' in Python such that $a = 4$ and $b = 5$. Swap the values of 'a' and 'b' without using a temporary variable. Print the values of 'a' and 'b' before and after swapping.

```
#  
a=4  
b=5  
print("Value of A=%d and B=%d"%(a,b))  
a,b=b,a  
print("Swapped Value of A=%d and B=%d"%(a,b))  
#
```

```
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a=4
>>> b=5
>>> print("Value of A=%d and B=%d"%(a,b))
Value of A=4 and B=5
>>> a,b=b,a
>>> print("Swapped Value of A=%d and B=%d"%(a,b))
Swapped Value of A=5 and B=4
>>> |
```

2) Consider the scenario of processing marks of a student in ABC Training Institute. John, the student of fifth grade takes exams in three different subjects. Create three variables to store the marks obtained by John in three subjects. Find and display the average marks scored by John. Now change the marks in one of the subjects and observe the output. Did the value of average change?

```
#
maths=96
eng=76
hindi=83
avg=(maths+eng+hindi)/3
print("Average marks is ",avg)
eng=84
print("Average marks is ",avg)
#
Value Of Average Did Not Change!!
```

```
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> maths=96
>>> eng=76
>>> hindi=83
>>> avg=(maths+eng+hindi)/3
>>> print("Average marks is ",avg)
Average marks is  85.0
>>> eng=84
>>> print("Average marks is ",avg)
Average marks is  85.0
>>> |
```

3) Given the value of radius of a circle, write a Python program to calculate the area and perimeter of the circle. Display both the values.

```
#  
r=float(input("ENTER RADIUS OF CIRCLE \n"))  
ar=3.14*r**2  
peri=2*3.14*r  
print("AREA OF CIRCLE IS %d Sq.cm"%ar)  
print("PERIMETER OF CIRCLE IS %d cm"%peri)  
#
```

The screenshot shows the Python 3.5.0 Shell window. The title bar says "Python 3.5.0 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The shell window displays the following text:

```
File Edit Shell Debug Options Window Help  
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AM  
D64)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>> r=float(input("ENTER RADIUS OF CIRCLE \n"))  
ENTER RADIUS OF CIRCLE  
12  
>>> ar=3.14*r**2  
>>> peri=2*3.14*r  
>>> print("AREA OF CIRCLE IS %d Sq.cm"%ar)  
AREA OF CIRCLE IS 452 Sq.cm  
>>> print("PERIMETER OF CIRCLE IS %d cm"%peri)  
PERIMETER OF CIRCLE IS 75 cm  
>>> |
```

4) The finance department of a company wants to compute the monthly pay of its employees. Monthly pay should be calculated as mentioned in the formula below. Display all the employee details. Monthly Pay = Number of hours worked in a week * Pay rate per hour * No. of weeks in a month

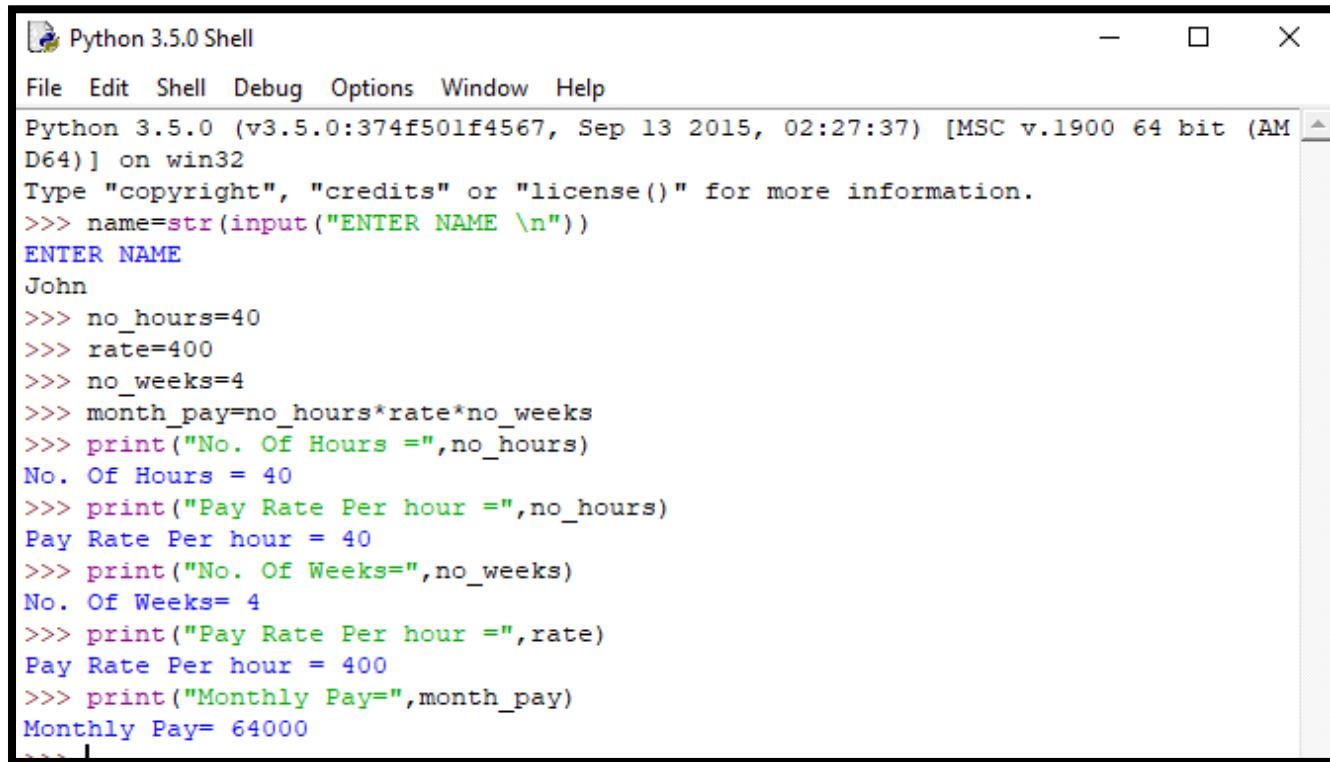
- The number of hours worked by the employee in a week should be considered as 40
- Pay rate per hour should be considered as Rs.400
- Number of weeks in a month should be considered as 4

```
#  
name=str(input("ENTER NAME \n"))  
no_hours=40  
rate=400  
no_weeks=4
```

```

month_pay=no_hours*rate*no_weeks
print("No. Of Hours =",no_hours)
print("No. Of Weeks=",no_weeks)
print("Pay Rate Per hour =",rate)
print("Monthly Pay=",month_pay)
#

```



The screenshot shows the Python 3.5.0 Shell window. The title bar says "Python 3.5.0 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The shell window displays the following code and its execution:

```

File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> name=str(input("ENTER NAME \n"))
ENTER NAME
John
>>> no_hours=40
>>> rate=400
>>> no_weeks=4
>>> month_pay=no_hours*rate*no_weeks
>>> print("No. Of Hours =",no_hours)
No. Of Hours = 40
>>> print("Pay Rate Per hour =",no_hours)
Pay Rate Per hour = 40
>>> print("No. Of Weeks=",no_weeks)
No. Of Weeks= 4
>>> print("Pay Rate Per hour =",rate)
Pay Rate Per hour = 400
>>> print("Monthly Pay=",month_pay)
Monthly Pay= 64000
>>>

```

FP5.0 Module-1 Assignments 12

Identify the sections of the given program where the coding standards are not followed and correct them.

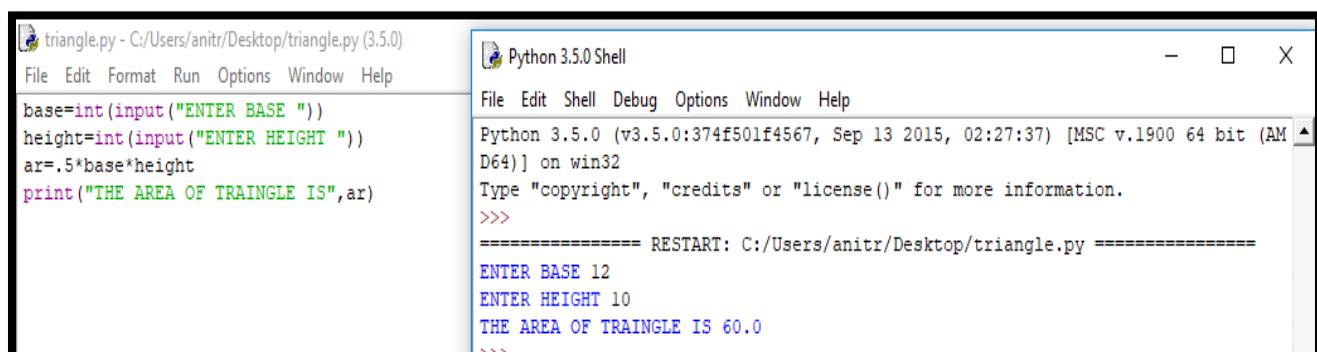
- itemNo =1005
- # item_no =1005
- unitprice = 250
- #unit_price =250
- quantity = 2
- #Correct
- amount=quantity*unitprice
- #amount = quantity * unit_price
- print("Item No:", itemNo)
- # print("Item No:", item_no)
- print("Bill Amount:", amount)
- #Correct

FP5.0 Module-1 Assignments 13

- Create a new file in Python IDLE as "triangle.py"
- Write a Python program to calculate and print the area of the triangle. Prompt the user to input the values for base and height of the triangle.
- Execute the program(use 'Run Module' under 'Run' tab) and observe the output.
- Close the file, open it again and execute it once more with different values. Observe the output.

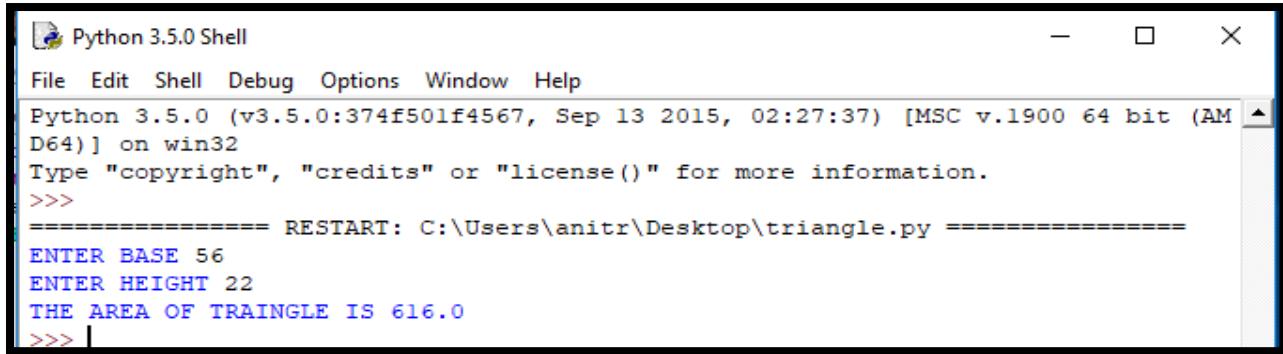
Hint – Use type casting for converting the input into an integer.
Area of a triangle = $\frac{1}{2} * \text{base} * \text{height}$

```
#  
base=int(input("ENTER BASE "))  
height=int(input("ENTER HEIGHT "))  
ar=.5*base*height  
print("THE AREA OF TRAINGLE IS",ar)  
#
```



The screenshot shows two windows side-by-side. On the left is the Python 3.5.0 Shell window, which displays the Python interpreter's prompt (>>>), the command to restart the application (===== RESTART: C:/Users/anitr/Desktop/triangle.py =====), and the user's input for base and height followed by the calculated area. On the right is the code editor window titled 'triangle.py - C:/Users/anitr/Desktop/triangle.py (3.5.0)', showing the Python script with variable names corrected from 'TRAINGLE' to 'TRIANGLE'.

```
triangle.py - C:/Users/anitr/Desktop/triangle.py (3.5.0)  
File Edit Format Run Options Window Help  
base=int(input("ENTER BASE "))  
height=int(input("ENTER HEIGHT "))  
ar=.5*base*height  
print("THE AREA OF TRIANGLE IS",ar)  
#  
  
Python 3.5.0 Shell  
File Edit Shell Debug Options Window Help  
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AM  
D64)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:/Users/anitr/Desktop/triangle.py =====  
ENTER BASE 12  
ENTER HEIGHT 10  
THE AREA OF TRIANGLE IS 60.0  
>>>
```



Python 3.5.0 Shell

File Edit Shell Debug Options Window Help

Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AM
D64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

===== RESTART: C:\Users\anitr\Desktop\triangle.py =====

ENTER BASE 56

ENTER HEIGHT 22

THE AREA OF TRAINGLE IS 616.0

>>> |

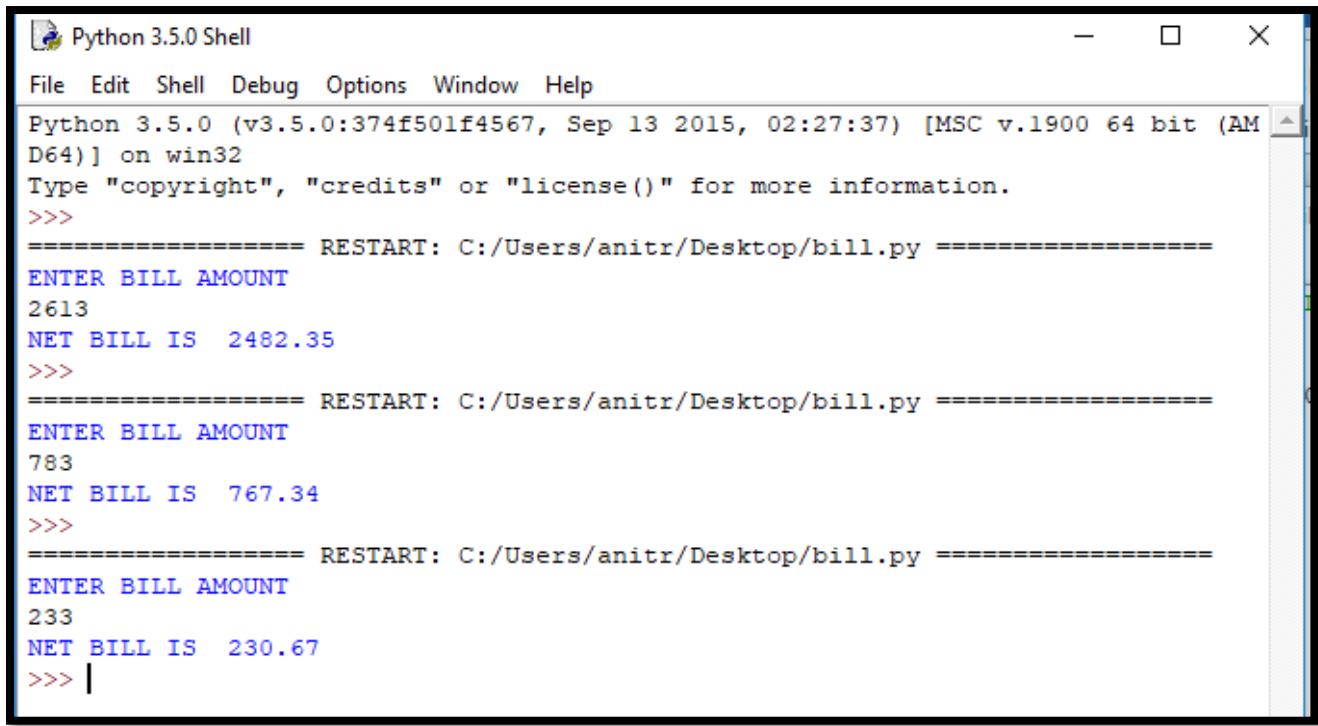
FP5.0 Module-1 Assignments 5

1) Consider the scenario of retail store management again. The store provides discount for all bill amounts based on the criteria below:

Bill Amount	Discount %
≥ 1000	5
$\geq 500 \text{ and } < 1000$	2
$> 0 \text{ and } < 500$	1

Write a Python program to find the net bill amount after discount. Observe the output with different values of bill amount. Assume that bill amount will be always greater than zero.

```
#  
bill=int(input("ENTER BILL AMOUNT\n"))  
if(bill>=1000):  
    net=bill-(.05*bill)  
elif(bill>=500) & (bill<1000):  
    net=bill-(.02*bill)  
else:  
    net=bill-(.01*bill)  
print("NET BILL IS ",net)  
#
```



```
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
=====
ENTER BILL AMOUNT
2613
NET BILL IS  2482.35
>>>
=====
ENTER BILL AMOUNT
783
NET BILL IS  767.34
>>>
=====
ENTER BILL AMOUNT
233
NET BILL IS  230.67
>>> |
```

**2) Extend the above program to validate the customer id.
Customer ids in the range of 101 and 1000 (both inclusive) should
only be considered valid.**

Note: Display appropriate error messages wherever applicable

```
#  
bill=int(input("ENTER BILL AMOUNT\n"))  
cust_id=int(input("ENTER CUSTOMER ID\n"))  
if(cust_id>=101)&(cust_id<=1000):  
    if(bill>=1000):  
        net=bill-(.05*bill)  
    elif(bill>=500) & (bill<1000):  
        net=bill-(.02*bill)  
    else:  
        net=bill-(.01*bill)  
    print("NET BILL IS ",net)  
else:  
    print("DISCOUNT NO AVAILABLE ")  
#
```

```
===== RESTART: C:/Users/anitr/Desktop/bill.py =====
ENTER BILL AMOUNT
698
ENTER CUSTOMER ID
123
NET BILL IS  684.04
>>>
===== RESTART: C:/Users/anitr/Desktop/bill.py =====
ENTER BILL AMOUNT
2563
ENTER CUSTOMER ID
100
DISCOUNT NO AVAILABLE
```

FP5.0 Module-1 Assignments 15

Implement the following in Python:

a. Display all even numbers between 50 and 80 (both inclusive) using "for" loop.

```
#  
num=0  
for num in range(50,82,2):  
    print(num)  
#
```

```
===== RESTART: C:/Users/anitr/Desktop/for.py =====  
50  
52  
54  
56  
58  
60  
62  
64  
66  
68  
70  
72  
74  
76  
78  
80
```

b. Add natural numbers up to n where n is taken as an input from user. Print the sum.

```
#  
num=int(input("ENTER NUMBER\n"))  
x=0  
sum=0  
while x<=num:  
    sum+=x  
    x+=1  
print("SUM IS ",sum)  
#
```

A screenshot of the Python 3.5.0 Shell window. The title bar says "Python 3.5.0 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main area shows the following text:

```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
=====
RESTART: C:/Users/anitr/Desktop/for.py =====
ENTER NUMBER
12
SUM IS 78
>>> |
```

c. Prompt the user to enter a number. Print whether the number is prime or not.

```
#  
num=int(input("ENTER NUMBER "))  
if num > 1:  
    for i in range(2,num):  
        if (num % i) == 0:  
            print(num,"is not a prime number")  
            break  
        else:  
            print(num,"is a prime number")  
    else:  
        print(num,"is not a prime number")  
#
```

A screenshot of the Python 3.5.0 Shell window. The title bar says "for.py - C:/Users/anitr/Desktop/for.py (3.5.0)". The menu bar includes File, *Python 3.5.0 Shell*, Edit, Shell, Debug, Options, Window, and Help. The main area shows the following text:

```
num
if
>>>
=====
RESTART: C:/Users/anitr/Desktop/for.py =====
ENTER NUMBER 21
21 is not a prime number
>>>
=====
RESTART: C:/Users/anitr/Desktop/for.py =====
ENTER NUMBER 667
667 is not a prime number
>>>
=====
RESTART: C:/Users/anitr/Desktop/for.py =====
ENTER NUMBER 13
13 is a prime number
>>> |
```

d. Print Fibonacci series till nth term where n is taken as an input from user.

Hint – Fibonacci series is a series of numbers in which each number is the sum of the two preceding numbers.

```
#  
n=int(input("ENTER VALUE OF N "))  
t1,t2=0,1  
print(t2)  
for i in range(1,n):  
    t3=t1+t2  
    t1=t2  
    t2=t3  
    print(t3)  
#
```

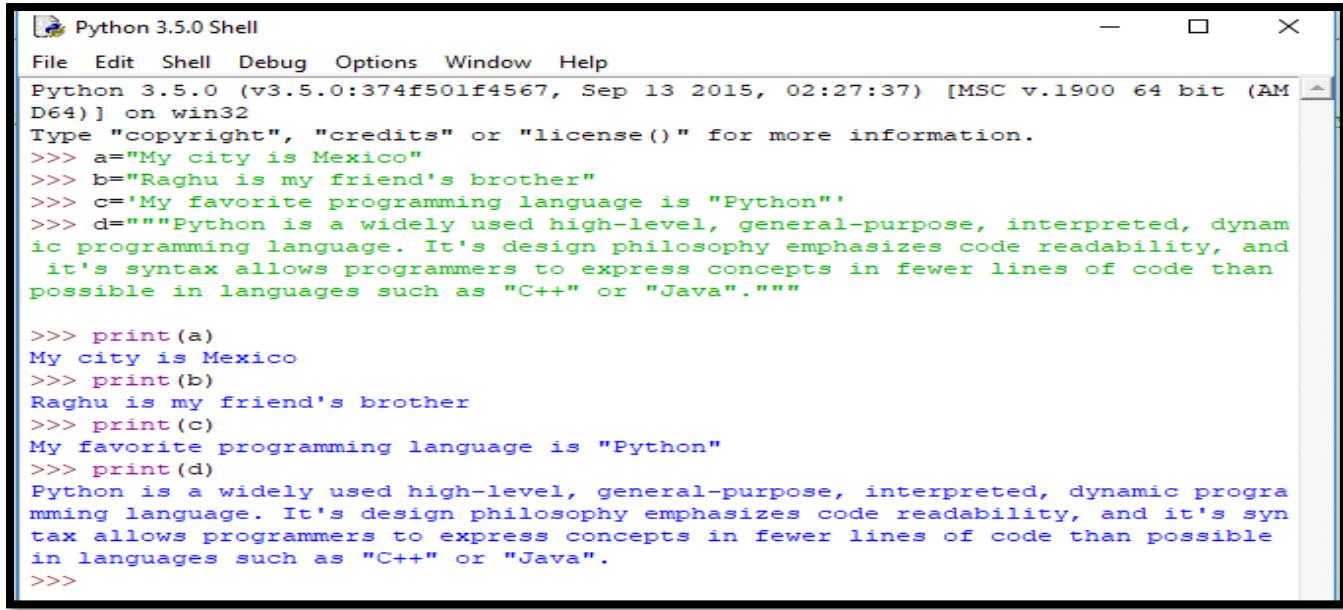
The screenshot shows the Python 3.5.0 Shell window. The title bar reads "Python 3.5.0 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The shell area displays the Python version information: "Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32". It also shows the copyright message: "Type "copyright", "credits" or "license()" for more information." A command prompt line starts with ">>>". Below it, a line of text indicates the script being run: "===== RESTART: C:/Users/anitr/Desktop/for.py =====". The user then enters "ENTER VALUE OF N 12" followed by a series of Fibonacci numbers: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144. The final command prompt ">>> |" is visible at the bottom.

FP5.0 Module-1 Assignments 16

Create four string variables a, b, c, d to store the following values and display them:

- My city is Mexico
- Raghu is my friend's brother
- My favorite programming language is "Python"
- Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. It's design philosophy emphasizes code readability, and it's syntax allows programmers to express concepts in fewer lines of code than possible in languages such as "C++" or "Java".

```
#  
a="My city is Mexico"  
b="Raghu is my friend's brother"  
c='My favorite programming language is "Python"'  
d="""Python is a widely used high-level, general-purpose, interpreted, dynamic programming language.  
It's design philosophy emphasizes code readability, and it's syntax allows programmers to express concepts  
in fewer lines of code than possible in languages such as "C++" or "Java"."""  
print(a)  
print(b)  
print(c)  
print(d)  
#
```



```
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a="My city is Mexico"
>>> b="Raghu is my friend's brother"
>>> c="My favorite programming language is "Python"
>>> d="""Python is a widely used high-level, general-purpose, interpreted, dynamic
ic programming language. It's design philosophy emphasizes code readability, and
it's syntax allows programmers to express concepts in fewer lines of code than
possible in languages such as "C++" or "Java."""
>>> print(a)
My city is Mexico
>>> print(b)
Raghu is my friend's brother
>>> print(c)
My favorite programming language is "Python"
>>> print(d)
Python is a widely used high-level, general-purpose, interpreted, dynamic pro
mning language. It's design philosophy emphasizes code readability, and it's syn
tax allows programmers to express concepts in fewer lines of code than possible
in languages such as "C++" or "Java".
>>>
```

FP5.0 Module-1 Assignments 17

- Accept a string as an input from the user. Check if the accepted string is palindrome or not.
- If the string is palindrome, print "String is palindrome", otherwise print "String is not palindrome".
- Also print the actual and the reversed strings.

Note – Ignore the case of characters.

Hint – A palindrome string remains the same if the characters of the string are reversed

```
#  
ch=str(input("ENTER STRING\n"))  
print("ACTUAL STRING",ch[0:])  
print("REVERSED STRING",ch[::-1])  
if(ch[:]==ch[::-1]):  
    print("STRING IS PALINDROME")  
else:  
    print("STRING IS NOT PALINDROME")  
#
```



The screenshot shows a Python 3.5.0 Shell window. The title bar says "Python 3.5.0 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the following code and its execution:

```
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:/Users/anitr/Desktop/new.py =====
ENTER STRING
hello
ACTUAL STRING hello
REVERSED STRING olleh
STRING IS NOT PALINDROME
>>> ===== RESTART: C:/Users/anitr/Desktop/new.py =====
ENTER STRING
malayalam
ACTUAL STRING malayalam
REVERSED STRING malayalam
STRING IS PALINDROME
>>> |
```

FP5.0 Module-1 Assignments 18

Accept two strings 'string1' and 'string2' as an input from the user. Generate a resultant string, such that it is a concatenated string of all upper case alphabets from both the strings in the order they appear. Print the actual and the resultant strings.
Note: Each character should be checked if it is a upper case alphabet and then it should be concatenated to the resultant string.

Sample Input:

string1: I Like C

string2: Mary Likes Python

Output: ILCMLP

#

```
string1=str(input("ENTER STRING 1 \n"))
string2=str(input("ENTER STRING 2 \n"))
ch=string1+string2
print("ACTUAL STRING:",ch)
k=""
for i in ch:
    if(i.isupper()):
        k+=i
print("RESULTANT STRING:",k)

#
```

```
===== RESTART: C:/Users/anitr/Desktop/new.py =====
ENTER STRING 1
I Like C
ENTER STRING 2
Mary Likes Python
ACTUAL STRING: I Like CMary Likes Python
RESULTANT STRING: ILCMLP
>>> |
```

FP5.0 Module-1 Assignments 19

**Given a string containing both upper and lower case alphabets.
Write a Python program to count the number of
occurrences of each alphabet(case insensitive) and display the
same.**

Sample Input: ABaBCbGc

Sample Output:
2A

3B

2C

1G

```
#  
ch=str(input("ENTER STRING \n"))  
print("ACTUAL STRING:",ch)  
count=0  
k={}  
for i in ch:  
    if(i!=' '):  
        k[i]=ch.count(i)  
for d in k:  
    print(k[d],d)  
#
```

```
>>>  
===== RESTART: C:/Users/anitr/Desktop/new.py ======  
ENTER STRING  
HELLO this is Anirudh  
ACTUAL STRING: HELLO this is Anirudh  
1 H  
1 d  
1 E  
2 s  
1 u  
1 A  
1 t  
1 O  
3 i  
2 L  
1 r  
2 h  
1 n  
>>> |
```

FP5.0 Module-1 Assignments 20

Write a Python program to accept a string 'accepted_string'. Generate a resultant string 'resultant_string' such that 'resultant_string' should contain all characters at the even position of 'accepted_string' (ignoring blank spaces). Display 'resultant_string' in reverse order.

accepted_string: An apple a day keeps the doctor away
resultant_string: Aapedyepteotrwy

expected_output: ywrtoetpeydepaA

Hint: String starts from index 0, hence the starting character is at even position.

```
#  
accepted_str=str(input("ENTER STRING \n"))  
print("ACTUAL STRING:",accepted_str)  
accepted_str=accepted_str.replace(' ','')  
resultant_str=accepted_str[::2]  
print("RESULTANT STRING:",resultant_str)  
print("REVERSED STRING:",resultant_str[::-1])  
#
```

```
===== RESTART: C:/Users/anitr/Desktop/new.py =====  
ENTER STRING  
AN APPLE A DAY  
ACTUAL STRING: AN APPLE A DAY  
RESULTANT STRING: AAPEDY  
REVERSED STRING: YDEPAA  
>>> |
```

FP5.0 Module-1 Assignments 21

Write a Python program to generate first 'n' Fibonacci numbers where 'n' is accepted as an input from the user.

Store the generated Fibonacci numbers in a list and display the output.

Sample input: 5

Sample output: [0, 1, 1, 2, 3]

```
#  
n=int(input("ENTER VALUE OF N "))  
t= [0, 1]  
for i in range(2,n):  
    t.append(t[i-1]+t[i-2])  
print(t)  
#
```

```
===== RESTART: C:/Users/anitr/Desktop/new.py =====
ENTER VALUE OF N 5
[0, 1, 1, 2, 3]
>>>
===== RESTART: C:/Users/anitr/Desktop/new.py =====
ENTER VALUE OF N 10
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
>>>
```

FP5.0 Module-1 Assignments 22

The "Variety Retail Store" sells different varieties of Furniture to the customers. The list of furniture available with its respective cost is given below:

The furniture and its corresponding cost should be stored as a list. A customer can order any furniture in any quantity (the name and quantity of the furniture will be provided). If the required furniture is available in the furniture list(given above) and quantity to be purchased is greater than zero, then bill amount should be calculated.

In case of invalid values for furniture required by the customer and quantity to be purchased, display appropriate error message and consider bill amount to be 0. Initialize required furniture and quantity with different values and test the results.

Write a Python program to calculate and display the bill amount to be paid by the customer based on the furniture bought and quantity purchased.

```
#  
furni=['Sofa Set','Dining Table','T.V. Stand','Cupboard']  
cost=[20000,8500,4599,13920]  
item=input("ENTER FURNITURE\\n")  
qty=int(input("ENTER QUANTITY\\n"))  
if(qty>0) & (item in furni):  
    for i in range(0,len(furni)):  
        if(item==furni[i]):  
            print(furni[i])  
            print("TOTAL COST IS ",qty*cost[i])  
else:  
    print("ENTER VALID INPUT")  
#
```



```
Python 3.5.0 Shell  
File Edit Shell Debug Options Window Help  
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AM  
D64)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:/Users/anitr/Desktop/LIST.py ======  
ENTER FURNITURE  
Cupboard  
ENTER QUANTITY  
3  
Cupboard  
TOTAL COST IS  41760  
>>>  
===== RESTART: C:/Users/anitr/Desktop/LIST.py ======  
ENTER FURNITURE  
Cupboard  
ENTER QUANTITY  
0  
ENTER VALID INPUT  
>>>  
===== RESTART: C:/Users/anitr/Desktop/LIST.py ======  
ENTER FURNITURE  
Stand  
ENTER QUANTITY  
3  
ENTER VALID INPUT  
>>> |
```

FP5.0 Module-1 Assignments 23

Consider the list of courses opted by a Student "John" and available electives at ABC Training Institute:

courses = ("Python Programming", "RDBMS", "Web Technology", "Software Engg.")

electives = ("Business Intelligence", "Big Data Analytics")

Write a Python Program to satisfy business requirements mentioned below:

1.List the number of courses opted by John.

2.List all the courses opted by John.

3.John is also interested in elective courses mentioned above. Print the updated tuple including electives.

```
#  
courses = ("Python Programming", "RDBMS", "Web Technology", "Software Engg.")  
electives = ("Business Intelligence", "Big Data Analytics")  
print("NUMBER OF COURSES OPTED ARE ",len(courses))  
print("ALL OPTED COURSES ARE ",courses)  
all_course=courses+electives  
print("OPTED COURSES PLUS ELECTIVES ",all_course)
```

```
>>>  
===== RESTART: C:/Users/anitr/Desktop/LIST.py ======  
NUMBER OF COURSES OPTED ARE 4  
ALL OPTED COURSES ARE ('Python Programming', 'RDBMS', 'Web Technology', 'Software Engg.')  
OPTED COURSES PLUS ELECTIVES ('Python Programming', 'RDBMS', 'Web Technology', 'Software Engg.', 'Business Intelligence', 'Big Data Analytics')  
>>> |
```

FP5.0 Module-1 Assignments 24

Given below is a dictionary 'customer_details' representing customer details from a Retail Application. Customer Id is the key and Customer Name is the value.

**customer_details = { 1001 : "John", 1004 : "Jill", 1005: "Joe",
1003 : "Jack" }**

Write Python code to perform the operations mentioned below:

a) Print details of customers.

b) Print number of customers.

c) Print customer names in ascending order.

d) Delete the details of customer with customer id = 1005 and print updated dictionary.

e) Update the name of customer with customer id = 1003 to "Mary" and print updated dictionary.

f) Check whether details of customer with customer id = 1002 exists in the dictionary.

```
#  
customer_details = { 1001 : "John", 1004 : "Jill", 1005: "Joe", 1003 : "Jack" }  
print(customer_details)  
print(len(customer_details))  
for i in sorted(customer_details):  
    print(customer_details[i])  
del customer_details[1005]  
print(customer_details)
```

```

d={1003 : "Mary"}
customer_details.update(d)
print(customer_details)
key=1002
print(key)
if key in customer_details:
    print("Key is present and value of the key is:")
    print(key)
else:
    print("Key isn't present!")
#

```

```

=====
RESTART: C:/Users/anitr/Desktop/LIST.py =====
{1001: 'John', 1003: 'Jack', 1004: 'Jill', 1005: 'Joe'}
4
John
Jack
Jill
Joe
{1001: 'John', 1003: 'Jack', 1004: 'Jill'}
{1001: 'John', 1003: 'Mary', 1004: 'Jill'}
1002
Key isn't present!

```

FP5.0 Module-1 Assignments 25

Consider a scenario from ABC Training Institute. The given table shows the marks scored by students of grade XI in Python Programming course.

Write a Python program to meet the requirements mentioned below:

- Display the name and marks for every student.**
- Display the top two scorers for the course.**
- Display class average of this course.**

Hint- Implement the solution using a dictionary.

```

#
details= { "John" : 86.5, "Jill" : 84.5, "Joe" : 80.5, "Jack" : 91.2 , "Harry" : 72.1}
print(details)
print(max(details, key=details.get))

```

```

sm=0
for i,j in details.items():
    sm+=j
avg=sm/len(details)
print(avg)
#

```

```

=====
RESTART: C:/Users/anitr/Desktop/LIST.py =====
{'John': 86.5, 'Jill': 84.5, 'Jack': 91.2, 'Joe': 80.5, 'Harry': 72.1}
HIGHEST SCORER: Jack
AVERAGE MARKS: 82.96
>>>

```

FP5.0 Module-1 Assignments 26

Consider the scenario from "Variety Retail Store" discussed in 'List' section. The list of furniture available with its respective cost is given below:

A customer can order any furniture in any quantity. If the required furniture is available in the furniture list(given above) and quantity to be purchased is greater than zero, then bill amount should be calculated. In case of invalid values for furniture required by the customer and quantity to be purchased, display appropriate error message and consider bill amount to be 0. Initialize required furniture and quantity with different values and test the results.

Calculate and display the bill amount to be paid by the customer based on the furniture bought and quantity purchased. Implement the given scenario using:

1) List of tuples

```

#
lis=[('Sofa Set',20000),('Dining Table',8500),('T.V. Stand',4599),('Cupboard',13920)]
print(lis)
item=input("ENTER FURNITURE\n")
qty=int(input("ENTER QUANTITY\n"))
if(qty>0):

```

```

for i in range(0,len(lis)):
    if(item in lis[i]):
        print(lis[i])
        print("TOTAL COST IS ",qty*lis[i])
else:
    print("ENTER VALID INPUT")
#

```

```

===== RESTART: C:/Users/anitr/Desktop/LIST.py =====
[('Sofa Set', 20000), ('Dining Table', 8500), ('T.V. Stand', 4599), ('Cupboard', 13920)]
ENTER FURNITURE
Cupboard
ENTER QUANTITY
3
('Cupboard', 13920)
TOTAL COST IS  41760
>>>

```

2)Dictionary

```

#
lis={'Sofa Set':20000,'Dining Table':8500,'T.V. Stand':4599,'Cupboard':13920}
print(lis)
item=input("ENTER FURNITURE\n")
qty=int(input("ENTER QUANTITY\n"))
if(qty>0):
    for i,j in lis.items():
        if(item==i):
            print(i)
            print("TOTAL COST IS ",qty*j)
else:
    print("ENTER VALID INPUT")
#

```

```

===== RESTART: C:/Users/anitr/Desktop/LIST.py =====
{'Sofa Set': 20000, 'T.V. Stand': 4599, 'Cupboard': 13920, 'Dining Table': 8500}
ENTER FURNITURE
Cupboard
ENTER QUANTITY
3
Cupboard
TOTAL COST IS  41760
>>> |

```

FP5.0 Module-1 Assignments 27

Consider a scenario from ABC Training Institute. Given below are two Sets representing the names of students enrolled for a particular course:

```
java_course = {"John", "Jack", "Jill", "Joe"}  
python_course = {"Jake", "John", "Eric", "Jill"}
```

Write a Python program to list the number of students enrolled for:

- 1) Python course
- 2) Java course only
- 3) Python course only
- 4) Both Java and Python courses
- 5) Either Java or Python courses but not both
- 6) Either Java or Python courses

```
#  
java_course = {"John", "Jack", "Jill", "Joe"}  
python_course = {"Jake", "John", "Eric", "Jill"}  
print("STUDENTS IN JAVA:",java_course)  
print("STUDENTS IN PYTHON:",python_course)  
print("STUDENTS IN JAVA ONLY:",java_course - python_course)  
print("STUDENTS IN PYTHON ONLY:",python_course - java_course)  
print("STUDENTS IN PYTHON & JAVA:",python_course & java_course)  
print("STUDENTS IN PYHTON OR JAVA BUT NOT BOTH:",python_course ^ java_course)  
print("STUDENTS IN PYTHON OR JAVA:",python_course | java_course)  
#
```

```
=====  
===== RESTART: C:/Users/anitr/Desktop/LIST.py ======  
STUDENTS IN JAVA: {'John', 'Jack', 'Jill', 'Joe'}  
STUDENTS IN PYTHON: {'Jake', 'John', 'Jill', 'Eric'}  
STUDENTS IN JAVA ONLY: {'Jack', 'Joe'}  
STUDENTS IN PYTHON ONLY: {'Jake', 'Eric'}  
STUDENTS IN PYTHON & JAVA: {'John', 'Jill'}  
STUDENTS IN PYHTON OR JAVA BUT NOT BOTH: {'Jack', 'Eric', 'Jake', 'Joe'}  
STUDENTS IN PYTHON OR JAVA: {'Jack', 'Eric', 'Jake', 'John', 'Jill', 'Joe'}  
>>>
```

FP5.0 Module-1 Assignments 28

Using functions, re-write and execute Python program to:

1. Add natural numbers upto n where n is taken as an input from user.

```
#  
num=int(input("ENTER NUMBER\n"))  
def total(num):  
    x=sum=0  
    while x<=num:  
        sum+=x  
        x+=1  
    print("SUM IS ",sum)  
total(num)  
#
```

```
===== RESTART: C:/Users/anitr/Desktop/LIST.py ======  
ENTER NUMBER  
6  
SUM IS  21  
>>>
```

2. Print Fibonacci series till nth term (Take input from user).

Note: You have implemented these programs using loops earlier.

```
#  
n=int(input("ENTER VALUE OF N "))  
def fibo(num):  
  
    t1,t2=0,1  
    print(t2)  
    for i in range(1,n):  
        t3=t1+t2  
        t1=t2  
        t2=t3  
        print(t3)  
fibo(n)  
#
```

```
===== RESTART: C:/Users/anitr/Desktop/LIST.py =====
ENTER VALUE OF N 10
1
1
2
3
5
8
13
21
34
55
>>>
```

FP5.0 Module-1 Assignments 29

At an airport, a traveler is allowed entry into the flight only if he clears the following checks:

- 1.Baggage Check
- 2.Immigration Check
- 3.Security Check

The logic for the check methods are given below:

`check_baggage (baggage_weight)`

•returns True if `baggage_weight` is greater than or equal to 0 and less than or equal to 40. Otherwise returns False.

`check_immigration (expiry_year)`

•returns True if `expiry_year` is greater than or equal to 2001 and less than or equal to 2025. Otherwise returns False.

`check_security(noc_status)`

•returns True if `noc_status` is 'valid' or 'VALID', for all other values return False.

`traveler()`

- Initialize the traveler Id and traveler name and invoke the functions `check_baggage()`, `check_immigration()` and `check_security()` by passing required arguments.
- Refer the table below for values of arguments.
- If all values of `check_baggage()`, `check_immigration()` and `check_security()` are true,

display traveler_id and traveler_name
display "Allow Traveler to fly!" Otherwise,
display traveler_id and traveler_name
display "Detain Traveler for Re-checking!"
Invoke the traveler() function. Modify the values of different variables in traveler() function and observe the output.

```
#  
def check_baggage (baggage_weight):  
    if (baggage_weight>=0) & (baggage_weight<=40):  
        return True  
    else:  
        return False  
def check_immigration(expiry_year):  
    if (expiry_year>=2001) & (expiry_year<=2025):  
        return True  
    else:  
        return False  
def check_security(noc_status):  
    if (noc_status=='valid') | (noc_status=='VALID'):  
        return True  
    else:  
        return False  
def traveler():  
    traveler_id=int(input("ENTER TRAVELER ID \n"))  
    traveler_name=input("ENTER TRAVELER NAME \n")  
    baggage_weight=int(input("ENTER BAG WEIGHT\n"))  
    expiry_year=int(input("ENTER TRAVEL YEAR\n"))  
    noc_status=input("ENTER NOC STATUS(VALID)\n")  
    if(check_baggage (baggage_weight) is True)&(check_immigration(expiry_year) is  
True)&(check_security(noc_status) is True):  
        print("CUSTOMER ID: ",traveler_id)  
        print("CUSTOMER NAME: ",traveler_name)  
        print("ALLOW TO FLY")  
    else:  
        print("CUSTOMER ID: ",traveler_id)  
        print("CUSTOMER NAME: ",traveler_name)  
        print("DETAIN TRAVELER FOR RE-CHECK")  
traveler()  
#
```



The image shows a screenshot of the Python 3.5.0 Shell window. The title bar reads "Python 3.5.0 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the following text:

```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AM  
D64)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:/Users/anitr/Desktop/LIST.py =====  
ENTER TRAVELER ID  
1001  
ENTER TRAVELER NAME  
Jim  
ENTER BAG WEIGHT  
35  
ENTER TRAVEL YEAR  
2019  
ENTER NOC STATUS(VALID)  
VALID  
CUSTOMER ID: 1001  
CUSTOMER NAME: Jim  
ALLOW TO FLY  
>>>
```

FP5.0 Module-1 Assignments 30

Consider the pseudo code for generating Fibonacci series using Recursion:

FIBO (number)

1. **if (number = 0) then**
2. **return (0)**
3. **else if (number = 1) then**
4. **return (1)**
5. **else**
6. **return FIBO(number - 1) + FIBO(number - 2)**
7. **end if**

Write a program in Python to implement the same using Recursion and execute it in Eclipse. Print appropriate error message if the user enters negative number as input.

```
#  
def fibo(number):  
    if(number == 0):  
        return (0)  
    elif(number == 1):  
        return (1)  
    else:
```

```

        return fibo(number - 1) + fibo(number - 2)
num=int(input("ENTER NUMBER"))
if num<0:
    print("ENTER VALID INPUT")
else:
    for i in range(num):
        print(fibo(i))
#

```

The screenshot shows the PyUnit IDE interface. The code editor window displays a Python script named FIBO.py. The code defines a function fibo that calculates the nth Fibonacci number using recursion. It also prompts the user for input, handles negative numbers by printing an error message, and prints the first n Fibonacci numbers. The code is annotated with a creation date of 15-Jun-2018 and an author name, anitr.

```

Created on 15-Jun-2018
@author: anitr
'''

def fibo(number):
    if(number == 0):
        return (0)
    elif(number == 1):
        return (1)
    else:
        return fibo(number - 1) + fibo(number - 2)
num=int(input("ENTER NUMBER"))
if num<0:
    print("ENTER VALID INPUT")
else:
    for i in range(num):
        print(fibo(i))

```

The PyUnit interface has tabs for PyUnit and Console. The Console tab shows the output of running the script. The user enters '10' as input. The program outputs the first 10 Fibonacci numbers: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34.

FP5.0 Module-1 Assignments 32

Write a Python program to:

- 1.read a file.
- 2.add backslash (\) before every double quote in the file contents.
- 3.write it to another file in the same folder.
- 4.print the contents of both the files.

For example:

If the first file is 'TestFile1.txt' with text as:

Jack said, "Hello Pune".

The output of the file 'TestFile2.txt' should be:

Jack said,\\"Hello Pune\\".

```
#  
f=open("C:\\\\Users\\\\anitr\\\\Desktop\\\\testfile1.txt","r+")  
data = f.read()  
print(data)  
f1=open("C:\\\\Users\\\\anitr\\\\Desktop\\\\testfile2.txt","w+")  
f1.write(data.replace('\\', '\\\\\\'))  
f1.close()  
f.close()  
f2=open("C:\\\\Users\\\\anitr\\\\Desktop\\\\testfile2.txt", "r+")  
data2=f2.read()  
print(data2)  
f2.close()  
#
```

The screenshot shows the PyCharm IDE interface. The top part is the code editor with the following content:

```
Created on 15-Jun-2018  
  
@author: anitr  
  
f=open("C:\\\\Users\\\\anitr\\\\Desktop\\\\testfile1.txt","r+")  
data = f.read()  
print(data)  
f1=open("C:\\\\Users\\\\anitr\\\\Desktop\\\\testfile2.txt","w+")  
f1.write(data.replace('\\', '\\\\\\'))  
f1.close()  
f.close()  
f2=open("C:\\\\Users\\\\anitr\\\\Desktop\\\\testfile2.txt", "r+")  
data2=f2.read()  
print(data2)  
f2.close()
```

The bottom part is the terminal window titled "Console" showing the execution results:

```
PyUnit Console <terminated> A:\\Python1\\Module1\\file.py  
Jack said, "Hello Pune"  
Jack said, \\"Hello Pune\\"
```

FP5.0 Module-1 Assignments 33

**Consider a file 'courses.txt' in D Drive with the following details:
Write a program to read the file and store the courses in Python variables as a:**

- 1) Dictionary (Sample - {0: 'Java', 1: 'Python', 2:'Javascript' 3:'PHP'})
- 2) List (Sample - ['Java', 'Python', 'Javascript', 'PHP'])

```

#
lis=[]
dic={}
i=0
f=open("C:\\Users\\anitr\\Desktop\\course.txt","r+")
for line in f:
    i+=1
    lis.append(line)
    dic[line]=i
print(lis)
print(dic)
f.close()
#

```

```

'''Created on 15-Jun-2018
@author: anitr
'''

lis=[]
dic={}
i=0
f=open("C:\\Users\\anitr\\Desktop\\course.txt","r+")
for line in f:
    i+=1
    lis.append([line])
    dic[line]=i
print(lis)
print(dic)
f.close()

<
PyUnit Console <terminated> A:\\Python1\\Module1\\file2.py
['Java\\n', 'Python\\n', 'Javascript\\n', 'PHP']
{'Python\\n': 2, 'Java\\n': 1, 'Javascript\\n': 3, 'PHP': 4}

```

FP5.0 Module-1 Assignments 34

Consider a file 'student_details.txt' in D Drive with the details of students in ABC institute – student id and name:

Write a program to read the file and store the student records in Python variable as:

- 1)List of lists**
- 2)List of dictionaries**

```
#lis=[]
lis2=[]
dic={}
i=0
f=open("C:\\Users\\anitr\\Desktop\\student_details.txt", "r+")
for line in f:
    temp=line.split()
    i+=1
    lis.append(temp)
    dic[line]=i
print(lis)
lis2.append(dic)
print(lis2)
f.close()
#
```

The screenshot shows a Python code editor with the following content:

```
Created on 15-Jun-2018

@author: anitr
'''

lis=[]
lis2=[]
dic={}
i=0
f=open ("C:\\Users\\anitr\\Desktop\\student_details.txt", "r+")
for line in f:
    temp=line.split()
    i+=1
    lis.append(temp)
    dic[line]=i
print(lis)
lis2.append(dic)
print(lis2)
f.close()

<
```

Below the code editor is a terminal window titled "Console". It shows the command "<terminated> A:\\Python1\\Module1\\file2.py" and the output:

```
[['101', 'Rahul'], ['102', 'Julie'], ['103', 'Helena'], ['104', 'Kally']]
[{'103 Helena \\n': 3, '104 Kally': 4, '101 Rahul\\n': 1, '102 Julie\\n': 2}]
```

FP5.0 Module-1 Assignments 35

Consider a file 'rhyme.txt' in D Drive with following text:

Write a Python program to count the words in the file using a dictionary (use space as a delimiter). Find unique words and the count of their occurrences(ignoring case). Write the output in another file "words.txt" at the same location

```
#  
dic={}  
dic2={}  
i,count=0,0  
f=open("C:\\Users\\anitr\\Desktop\\rhyme.txt","r+")  
f1=open("C:\\Users\\anitr\\Desktop\\words.txt","w+")  
for line in f:  
    for temp in line.split():  
        i+=1  
        dic[temp]=i  
        temp=temp.lower()  
        dic2[temp]=i  
f1.write("TOTAL NO. OF WORDS: "+str(i)+"\n")  
f1.write(str(dic)+"\n")  
f1.write("UNIQUE WORDS: "+str(len(dic2))+"\n")  
for x in dic2:  
    f1.write(x+"\n")  
f.close()  
f1.close()  
f3=open("C:\\Users\\anitr\\Desktop\\words.txt","r+")  
print(f3.read())  
f3.close()  
#
```

```
Ru PyUnit Console ✘  
<terminated> A:\Python1\Module1\file2.py  
TOTAL NO. OF WORDS: 29  
{'the': 28, 'ride': 15, 'is': 13, 'fun': 11, 'it': 12, 'jingle': 24, 'In': 16, 'Jingle': 26, 'sleigh': 21, 'all': 27, 'open': 20}  
UNIQUE WORDS: 18  
the  
open  
is  
fun  
it  
jingle  
bells  
sleigh  
all  
to  
horse  
oh  
what  
one  
in  
way  
a  
ride
```

FP5.0 Module-1 Assignments 36

Assume the following Python code:

```
mylist=[1,2,3,"4",5]
sum=0
for i in mylist:
    sum+=i
print(sum)
print(mylist[5])
```

Rewrite the code to handle the exceptions raised. Print appropriate error messages wherever applicable.

```
#  
mylist=[1,2,3,"4",5]  
sum=0  
for i in mylist:  
    try:  
        sum+=i  
    except TypeError:  
        print("Type Error")  
print(sum)  
try:  
    print(mylist[5])  
except IndexError:  
    print("Index Error Occured")  
#
```

The screenshot shows the PyCharm IDE interface. The code editor window displays the provided Python script. The terminal window at the bottom shows the execution results: it prints 'Type Error' when attempting to add a string to an integer, and 'Index Error Occured' when trying to access a non-existent index in the list.

```
'''  
Created on 15-Jun-2018  
@author: anitrmylist=[1,2,3,"4",5]  
sum=0  
for i in mylist:  
    try:  
        sum+=i  
    except TypeError:  
        print("Type Error")  
print(sum)  
try:  
    print(mylist[5])  
except IndexError:  
    print("Index Error Occured")  
  
PyUnit Console <terminated> A:\Python1\Module1\exception.py  
Type Error  
11  
Index Error Occured
```

FP5.0 Module-1 Assignments 37

You have already created a Python program to implement the following in file handling section:

- 1.read a file.
- 2.add backslash (\) before every double quote in the file contents.
- 3.write it to another file in the same folder.
- 4.print the contents of both the files.

Modify your code to implement Exception handling. Print appropriate error messages wherever applicable

```
#  
try:  
    f=open("C:\\\\Users\\\\anitr\\\\Desktop\\\\testfile1.txt","r+")  
    data = f.read()  
    print(data)  
except FileNotFoundError:  
    print("FileNotFoundException\\n File Not FOUND")  
else:  
    f1=open("C:\\\\Users\\\\anitr\\\\Desktop\\\\testfile2.txt","w+")  
    f1.write(data.replace('"', '\"'))  
    f1.close()  
    f.close()  
try:  
    f2=open("C:\\\\Users\\\\anitr\\\\Desktop\\\\testfile2.txt","r+")  
except FileNotFoundError:  
    print("FileNotFoundException\\n File Not FOUND")  
else:  
    data2=f2.read()  
    print(data2)  
    f2.close()  
#
```

```
@author: anitr
'''
try:
    f=open("C:\\Users\\anitr\\Desktop\\testfile1.txt", "r+")
    data = f.read()
    print(data)
except FileNotFoundError:
    print("FileNotFoundException\\n File Not FOUND")
else:
    f1=open("C:\\Users\\anitr\\Desktop\\testfile2.txt", "w+")
    f1.write(data.replace('\\', '\\\\'))
    f1.close()
    f.close()
try:
    f2=open("C:\\Users\\anitr\\Desktop\\testfile2.txt", "r+")
except FileNotFoundError:
    print("FileNotFoundException\\n File Not FOUND")
else:
    data2=f2.read()
    print(data2)
    f2.close()

<
PyUnit Console ✎
terminated> A:\Python1\Module1\exception2.py
FileNotFoundException
File Not FOUND
FileNotFoundException
File Not FOUND
```

FP5.0 Module-1 Assignments 32

You have already executed the Python program given below in Functions section:

•Add natural numbers up to n where n is taken as an input from user.

Do appropriate exception handling in the code and observe the output by providing invalid input values.

```
# 
try:
    num=int(input("ENTER NUMBER\\n"))
except:
    print("ERROR!!!\\nENTER A NATURAL NUMBER ONLY")
else:
    x=0
    sum=0
    while x<=num:
```

```
sum+=x  
x+=1  
print("SUM IS ",sum)
```

```
'''  
Created on 15-Jun-2018  
  
@author: anitr  
'''  
  
try:  
    num=int(input("ENTER NUMBER\n"))  
except:  
    print("ERROR!!!\nENTER A NATURAL NUMBER ONLY")  
else:  
    x=0  
    sum=0  
    while x<=num:  
        sum+=x  
        x+=1  
    print("SUM IS ",sum)
```

The screenshot shows a Python code editor window. At the top, there's a status bar with icons for file operations. Below it is a toolbar with tabs for "PyUnit" and "Console". The main area contains the Python script. In the "Console" tab, the script is run, and the output shows an attempt to enter a non-integer value, which results in the error message "ERROR!!! ENTER A NATURAL NUMBER ONLY".

FP5.0 Module-1 Assignments 39

Refer to the following assignment which you have already executed in Functions section. Modify your code to implement Exception Handling and display appropriate error message wherever applicable.

At an airport, a traveler is allowed entry into the flight only if he clears the following checks:

- 1.Baggage Check
- 2.Immigration Check
- 3.Security Check

The logic for the check methods are given below:

check_baggage (baggage_weight)

- returns True if baggage_weight is greater than or equal to 0 and less than or equal to 40. Otherwise returns False.

check_immigration (expiry_year)

- returns True if expiry_year is greater than or equal to 2001 and less than or equal to 2025. Otherwise returns False.

check_security(noc_status)

- returns True if noc_status is 'valid' or 'VALID', for all other values return False.

traveler()

- Initialize the traveler Id and traveler name and invoke the functions check_baggage(), check_immigration() and check_security() by passing required arguments.
 - Refer the table below for values of arguments.
 - If all values of check_baggage(), check_immigration() and check_security() are true, display traveler_id and traveler_name display "Allow Traveler to fly!" Otherwise, display traveler_id and traveler_name display "Detain Traveler for Re-checking!"
- Invoke the traveler() function. Modify the values of different variables in traveler() function and observe the output

```
#  
def check_baggage (baggage_weight):  
    if (baggage_weight>=0) & (baggage_weight<=40):  
        return True  
    else:  
        return False  
def check_immigration(expiry_year):  
    if (expiry_year>=2001) & (expiry_year<=2025):  
        return True  
    else:  
        return False  
def check_security(noc_status):  
    if (noc_status=='valid') | (noc_status=='VALID'):  
        return True  
    else:  
        return False  
def traveler():  
    try:  
        traveler_id=int(input("ENTER TRAVELER ID \n"))
```

```

except ValueError:
    print("VALUE ERROR!!!\n NUMBER ONLY")
traveler_name=input("ENTER TRAVELER NAME \n")
try:
    baggage_weight=int(input("ENETR BAG WEIGHT\n"))
except ValueError:
    print("VALUE ERROR!!!\n NUMBER ONLY")
try:
    expiry_year=int(input("ENETR TRAVEL YEAR\n"))
except ValueError:
    print("VALUE ERROR!!!\n NUMBER ONLY")
noc_status=input("ENETR NOC STATUS(VALID)\n")
if(check_baggage(baggage_weight) is True)&(check_immigration(expiry_year) is
True)&(check_security(noc_status) is True):
    print("CUSTOMER ID: ",traveler_id)
    print("CUSTOMER NAME: ",traveler_name)
    print("ALLOW TO FLY")
else:
    print("CUSTOMER ID: ",traveler_id)
    print("CUSTOMER NAME: ",traveler_name)
    print("DETAIN TRAVELER FOR RE-CHECK")
traveler()
#

```

```

<terminated> A:\Python1\Module1\exception4.py
ENTER TRAVELER ID
fdf
VALUE ERROR!!!
NUMBER ONLY
ENTER TRAVELER NAME
101
ENETR BAG WEIGHT
fdf
VALUE ERROR!!!
NUMBER ONLY
ENETR TRAVEL YEAR
fsf
VALUE ERROR!!!
NUMBER ONLY
ENETR NOC STATUS(VALID)
Valid

```

FP5.0 Module-1 Assignments 40

- **Create a module "number_checker.py" which has following 2 functions:**
 - **is_prime(num) : this function returns true if the input number is prime**
 - **is_even(num): this function returns true if the input number is even**
- **Create another Python module "test_module.py".**
- **Invoke the functions "is_prime(num)" and "is_even(num)" in "test_module.py".**
- **Observe the results.**

Hint: Import "number_checker.py" module in "test_module.py" before using it's functions.

Number_checker.py

```
#  
def is_prime(num):  
    if num > 1:  
        for i in range(2,num):  
            if (num % i) == 0:  
                print(num,"is not a prime number")  
                break  
            else:  
                print(num,"is a prime number")  
                break  
    else:  
        print(num,"is not a prime number")  
def is_even(num):  
    if(num%2==0):  
        print(num," IS EVEN")  
    else:  
        print(num," IS ODD")  
#  
Test_module.py  
#  
import number_checker
```

```
n=int(input("ENTER NUMBER \n"))
number_checker.is_even(n)
number_checker.is_prime(n)
#
def is_prime(num):
    if num > 1:
        for i in range(2,num):
            if (num % i) == 0:
                print(num, "is not a prime number")
                break
            else:
                print(num, "is a prime number")
                break
        else:
            print(num, "is not a prime number")
def is_even(num):
    if(num%2==0):
        print(num, " IS EVEN")
    else:
        print(num, " IS ODD")
```

checker_module ✘ number_checker

```
import number_checker
n=int(input("ENTER NUMBER \n"))
number_checker.is_even(n)
number_checker.is_prime(n)
```

PyUnit ✘ Console ✘

```
<terminated> A:\Python1\Module1\checker_module.py
ENTER NUMBER
13
13  IS ODD
13 is a prime number
```

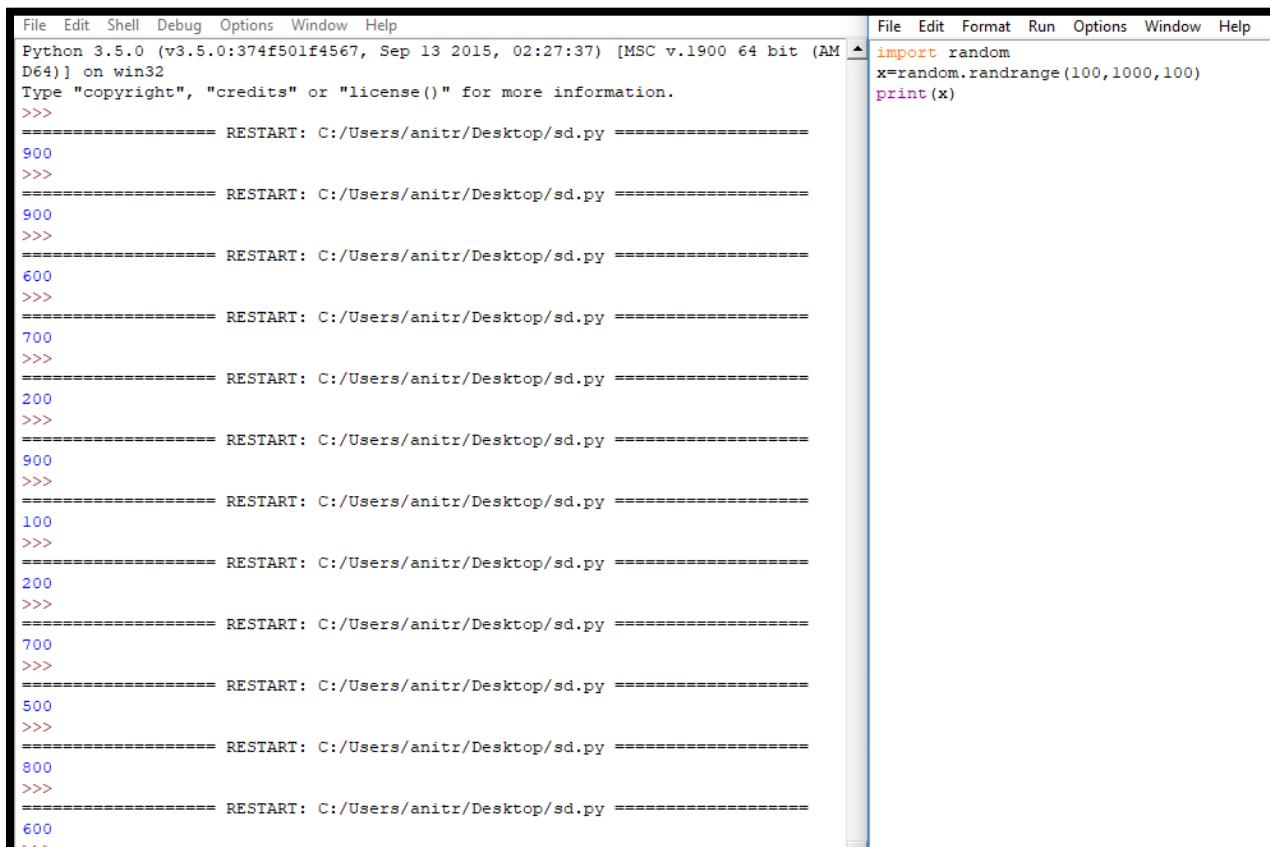
FP5.0 Module-1 Assignments 41

- Write a Python program to randomly print any of the below numbers:

100,200,300,400,500,600,700,800,900,1000

Execute the program 10 times and verify if the number generated in every output is one out of the numbers given in the list above.

```
#  
import random  
x=random.randrange(100,1000,100)  
print(x)  
#
```

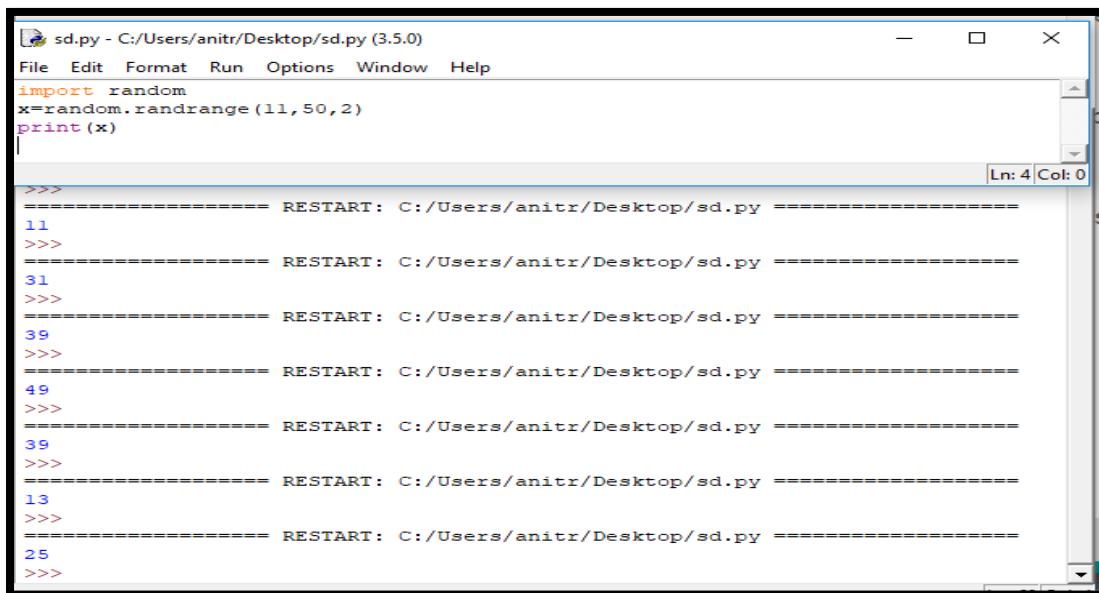


The image shows a screenshot of a Python terminal window. The terminal is running Python 3.5.0 on Windows. It displays 10 separate runs of the provided code, each printing a different random integer between 100 and 1000. The output is as follows:

```
File Edit Shell Debug Options Window Help  
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AM  
D64)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py ======  
900  
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py ======  
900  
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py ======  
600  
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py ======  
700  
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py ======  
200  
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py ======  
900  
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py ======  
100  
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py ======  
200  
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py ======  
700  
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py ======  
500  
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py ======  
800  
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py ======  
600
```

- Write a Python program to print a random odd numbers between 10 and 50.

```
#  
import random  
x=random.randrange(11,50,2)  
print(x)  
#
```



The screenshot shows a Python IDLE window. The script file 'sd.py' contains the following code:

```
sd.py - C:/Users/anitr/Desktop/sd.py (3.5.0)  
File Edit Format Run Options Window Help  
import random  
x=random.randrange(11,50,2)  
print(x)
```

The execution output shows the program running multiple times, each time printing a different odd number between 11 and 49. The output is as follows:

```
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py =====  
11  
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py =====  
31  
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py =====  
39  
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py =====  
49  
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py =====  
39  
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py =====  
13  
>>>  
===== RESTART: C:/Users/anitr/Desktop/sd.py =====  
25  
>>>
```

FP5.0 Module-1 Assignments 42

Write a Python program for rolling a dice on clicking enter key.
The program should run infinitely until user enters 'q'.
Hint: To implement a dice, you can randomly print a number in the range 1-6.

```
#  
import random  
leave=0
```

```

while leave!='q':
    print("PRESS ENTER TO ROLL DICE")
    input()
    print(random.randint(1,6))
    print("ENTER Q TO EXIT")
    leave=input()
#

```

```

sd.py - C:/Users/anitr/Desktop/sd.py (3.5.0)
File Edit Format Run Options Window Help
import random
leave=0
while leave!='q':
    print("PRESS ENTER TO ROLL DICE")
    input()
    print(random.randint(1,6))
    print("ENTER Q TO EXIT")
    leave=input()

Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit
(AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/anitr/Desktop/sd.py =====
PRESS ENTER TO ROLL DICE

1
ENTER Q TO EXIT

PRESS ENTER TO ROLL DICE

5
ENTER Q TO EXIT

PRESS ENTER TO ROLL DICE

3
ENTER Q TO EXIT

PRESS ENTER TO ROLL DICE

1
ENTER Q TO EXIT
q
>>> |

```

FP5.0 Module-1 Assignments 43

If area of one wall of a cubical wooden box is 16 units, write a Python program to display the volume of the box.

Note:

Area of a cube with side 'a' is ' $a^{}2$ '.**

Volume of the cube can be computed as ' $a^{}3$ '.**

Hint: Make use of 'sqrt' and 'pow' functions from math module.

```

#
import math
ar=16
l=math.sqrt(ar)
print("AREA OF A SIDE OF CUBE ",ar)
print("SIDE OF CUBE IS ",l)
arc=6*math.pow(l,2)
vol=math.pow(l,3)
print("AREA OF CUBE IS ",arc)
print("VOLUME OF CUBE IS ",vol)

```

#

The screenshot shows the Python 3.5.0 IDE interface. The top window is titled "sd.py - C:/Users/anitr/Desktop/sd.py (3.5.0)" and contains the following Python code:

```
import math
ar=16
l=math.sqrt(ar)
print("AREA OF A SIDE OF CUBE ",ar)
print("SIDE OF CUBE IS ",l)
arc=6*math.pow(l,2)
vol=math.pow(l,3)
print("AREA OF CUBE IS ",arc)
print("VOLUME OF CUBE IS ",vol)
```

The bottom window is titled "Python 3.5.0 Shell" and shows the execution of the script:

```
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/anitr/Desktop/sd.py =====
AREA OF A SIDE OF CUBE  16
SIDE OF CUBE IS  4.0
AREA OF CUBE IS  96.0
VOLUME OF CUBE IS  64.0
>>> |
```

A status bar at the bottom right indicates "Ln: 9 Col: 4".

FP5.0 Module-1 Assignments 44

The ABC Institute offers vocational courses to students in multiple areas e.g. theatre, classical singing, traditional dance forms, Bollywood dance, literature and so on. A student can enroll for zero to all courses.

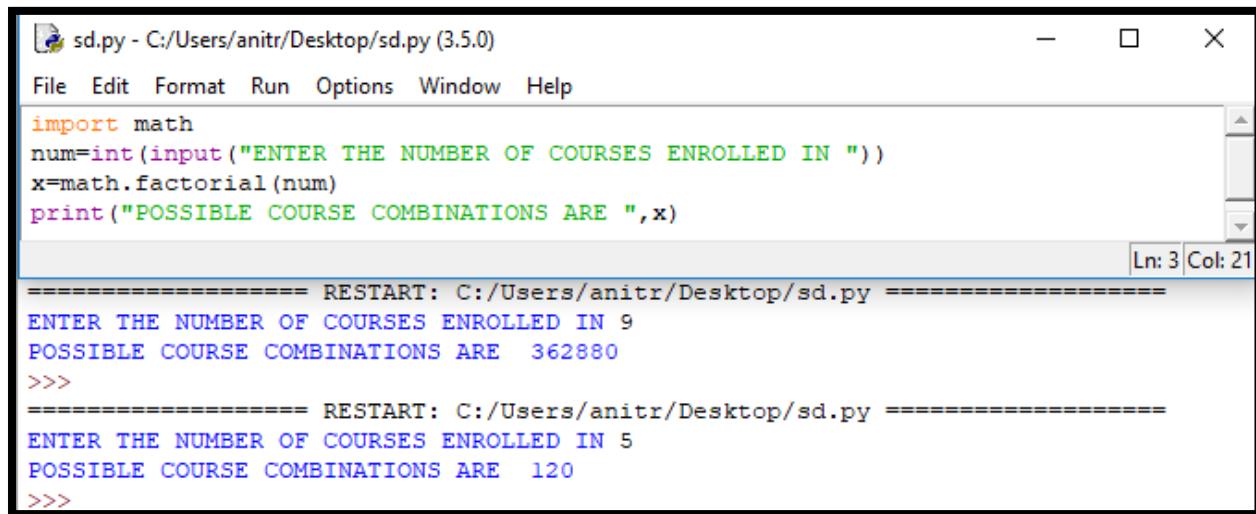
Write a Python function that takes the number of courses as an input and returns the total number of different course combinations, a student can opt for. (Make use of functions available in math module)

Hint: if no_of_courses = 2, possible number of combinations are 2! i.e. 2 if no_of_courses = 3, possible number of combinations are 3! i.e. 6 and so on.

```

#
import math
num=int(input("ENTER THE NUMBER OF COURSES ENROLLED IN "))
x=math.factorial(num)
print("POSSIBLE COURSE COMBINATIONS ARE ",x)
#

```



```

sd.py - C:/Users/anitr/Desktop/sd.py (3.5.0)
File Edit Format Run Options Window Help
import math
num=int(input("ENTER THE NUMBER OF COURSES ENROLLED IN "))
x=math.factorial(num)
print("POSSIBLE COURSE COMBINATIONS ARE ",x)

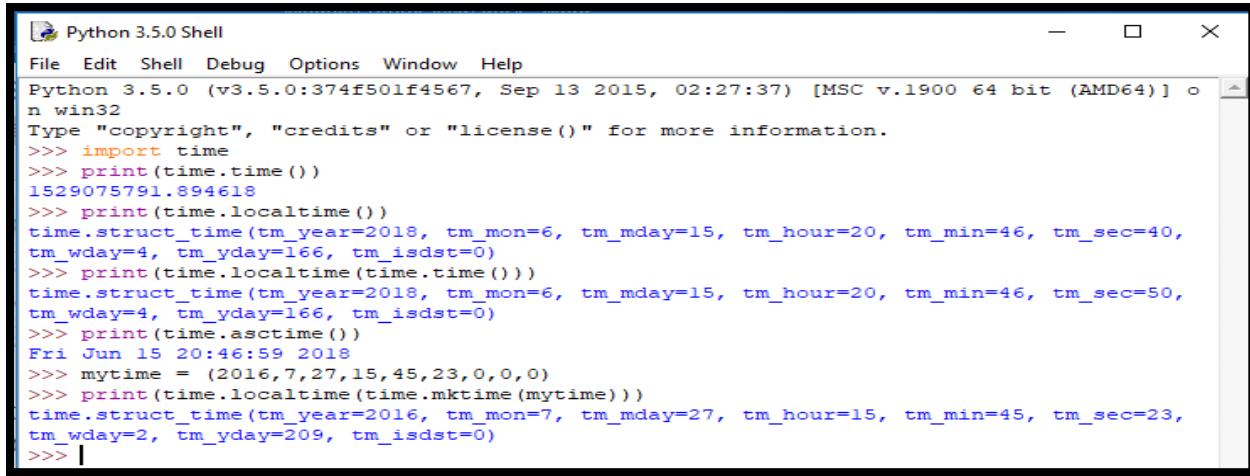
=====
RESTART: C:/Users/anitr/Desktop/sd.py =====
ENTER THE NUMBER OF COURSES ENROLLED IN 9
POSSIBLE COURSE COMBINATIONS ARE  362880
>>>
===== RESTART: C:/Users/anitr/Desktop/sd.py =====
ENTER THE NUMBER OF COURSES ENROLLED IN 5
POSSIBLE COURSE COMBINATIONS ARE  120
>>>

```

FP5.0 Module-1 Assignments 45

Execute the following code and observe the output.

- 1.import time
- 2.print(time.time())
- 3.print(time.localtime())
- 4.print(time.localtime(time.time()))
- 5.print(time.asctime())
- 6.mytime = (2016,7,27,15,45,23,0,0,0)
- 7.print(time.localtime(time.mktime(mytime)))



The screenshot shows a Windows window titled "Python 3.5.0 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, Help. The status bar at the bottom says "Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] o n win32". The main area contains Python code demonstrating the time module:

```
Type "copyright", "credits" or "license()" for more information.
>>> import time
>>> print(time.time())
1529075791.894618
>>> print(time.localtime())
time.struct_time(tm_year=2018, tm_mon=6, tm_mday=15, tm_hour=20, tm_min=46, tm_sec=40,
tm_wday=4, tm_yday=166, tm_isdst=0)
>>> print(time.localtime(time.time()))
time.struct_time(tm_year=2018, tm_mon=6, tm_mday=15, tm_hour=20, tm_min=46, tm_sec=50,
tm_wday=4, tm_yday=166, tm_isdst=0)
>>> print(time.asctime())
Fri Jun 15 20:46:59 2018
>>> mytime = (2016,7,27,15,45,23,0,0,0)
>>> print(time.localtime(time.mktime(mytime)))
time.struct_time(tm_year=2016, tm_mon=7, tm_mday=27, tm_hour=15, tm_min=45, tm_sec=23,
tm_wday=2, tm_yday=209, tm_isdst=0)
>>> |
```

FP5.0 Module-1 Assignments 46

Consider a Python string:

`cust_details = "Hello John, your customer id is j181"`

- 1) Find, if the name of the customer is preceded by a pattern "Hello " or "hello " (Observe a space after the word)? If pattern is found, print the searched result.
- 2) Find, if the given string ends with a pattern containing only one alphabet followed by three numbers? If pattern is found, print the searched result.
- 3) Replace the word starting with "j" followed by three numbers to only the number(remove the alphabet).
- 4) Replace the word "id" with "ID". The output of the above code is "Hello John, your customer ID is 181"

```
#  
import re  
cust_details = "Hello John, your customer id is j181"  
matched=re.search("Hello ",cust_details)  
print(matched.group())  
#matched=re.search("hello ",cust_details)  
#print(matched.group())  
matched=re.search("j\d{3}",cust_details)  
print(matched.group())  
new_str=re.sub("j","",cust_details)  
print(new_str)  
final_str=re.sub("id","ID",new_str)  
print(final_str)  
#
```

The screenshot shows a Windows-style application window titled "re.py - C:/Users/anitr/Desktop/re.py (3.5.0)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains the following Python script:

```
import re
cust_details = "Hello John, your customer id is j181"
matched=re.search("Hello ",cust_details)
print(matched.group())
#matched=re.search("hello ",cust_details)
#print(matched.group())
matched=re.search("j\d{3}",cust_details)
print(matched.group())
new_str=re.sub("j","",cust_details)
print(new_str)
final_str=re.sub("id","ID",new_str)
print(final_str)
```

The terminal output shows the execution of the script:

```
===== RESTART: C:/Users/anitr/Desktop/re.py =====
Hello
j181
Hello John, your customer id is 181
Hello John, your customer ID is 181
>>>
```

A status bar at the bottom right indicates "Ln: 6 Col: 1".