Lab report no: 01

Lab report name: Introduction to Python

Objectives:

1. Setup python environment for programing,
2. Learn the basics of python,
3. Create and run basic examples using python.

Theory:

Definition of python: Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object- oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

Main Features of Python: The main features of Python are:

- **Simple**: Python is a simple and minimalistic language. This pseudo-code nature of Python is one of its greatest strengths.
- **Easy to Learn:** Python is extremely easy to get started. Python has an extraordinarily simple syntax.
- **Free and Open Source:** Python is an example of FLOSS (Free/Libré and Open Source Software). In simple terms, i can freely distribute copies of this software, read it's source code, make changes to it, use pieces of it in new free programs, and i know that

can do these things. FLOSS is based on the concept of a community which shares knowledge.

- **High-level Language:** When i write programs in Python, I never need to bother about the low-level details such as managing the memory used by your program, etc.
- **Portable:** Due to its open-source nature, Python has been ported (i.e. changed to make it work on) to many platforms. All Python programs can work on any of these platforms without requiring any changes.
- **Multi-Plarform:** Python can be used on Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and even Pocket PC.
- **Interpreted:** Python does not need compilation to binary .I just run the program directly from the source code. Internally, Python converts the source code into an intermediate form called byte codes and then translates this into the native language of my computer and then runs it.
- **Object Oriented:** Python supports procedure-oriented programming as well as object oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In

object oriented languages, the program is built around objects which combine data and functionality.

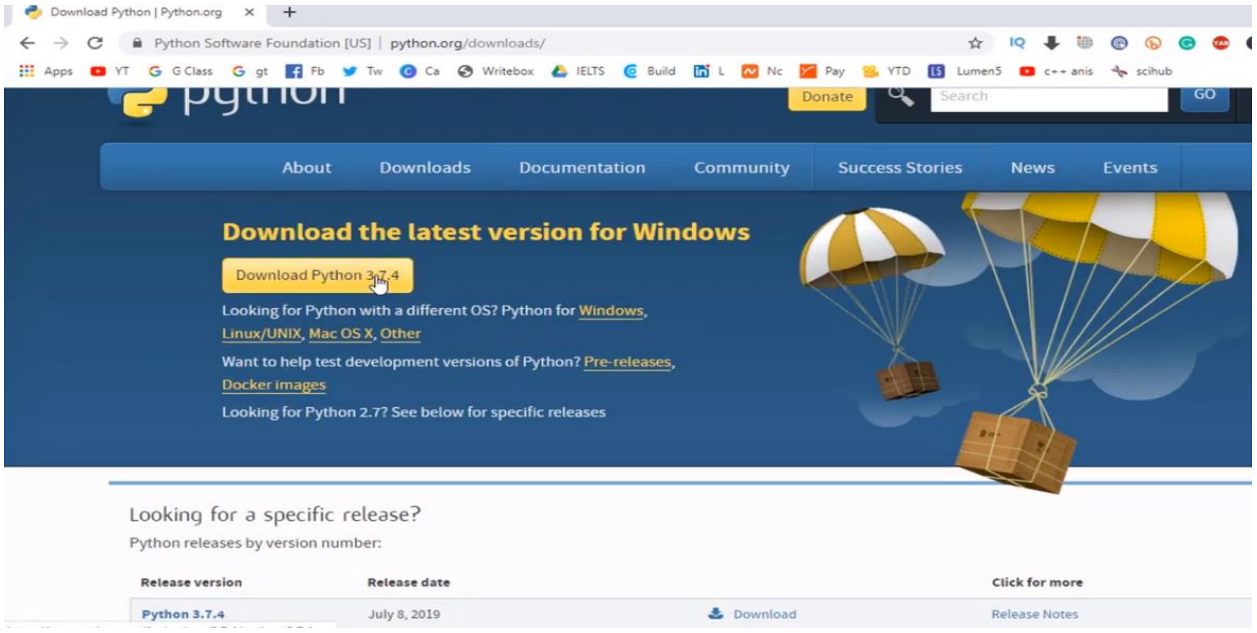
Extensive Libraries: The Python Standard Library is huge indeed. It can help our do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, ftp, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), and other system-dependent stuff. Remember, all this is always available wherever Python is installed.

Methodology: Setup of Python Environment

First install: 1.Python

2. pyCharm

Step-1**:** Download the latest version python 3.7.4


Download Python | Python.org
Python Software Foundation [US] | python.org/downloads/
Apps YT G Class gt Fb Tw Ca Writebox IELTS Build L Nc Pay YTD Lumen5 c++ anis scihub
python
Donate
Search
GO
About
Downloads
Documentation
Community
Success Stories
News
Events
Download Python 3.7.4

Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases, Docker images

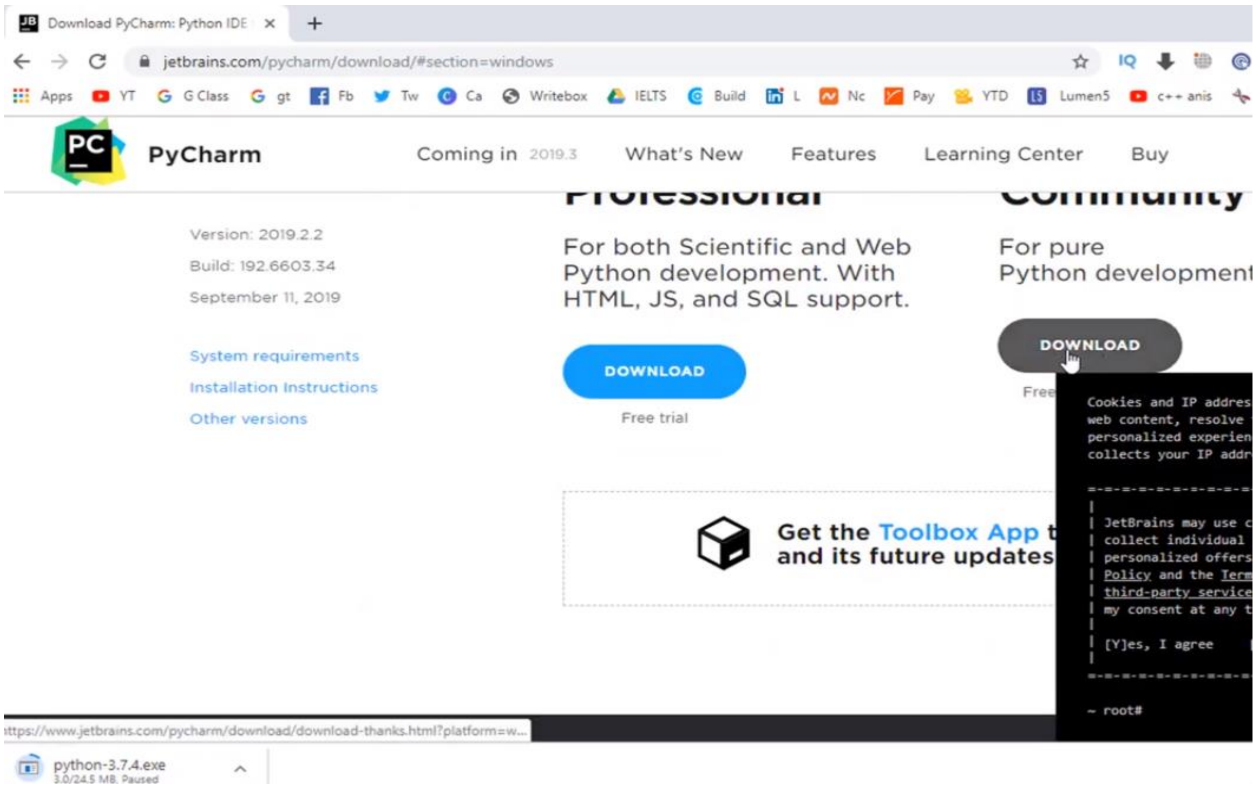
Looking for Python 2.7? See below for specific releases

Looking for a specific release?

Python releases by version number:

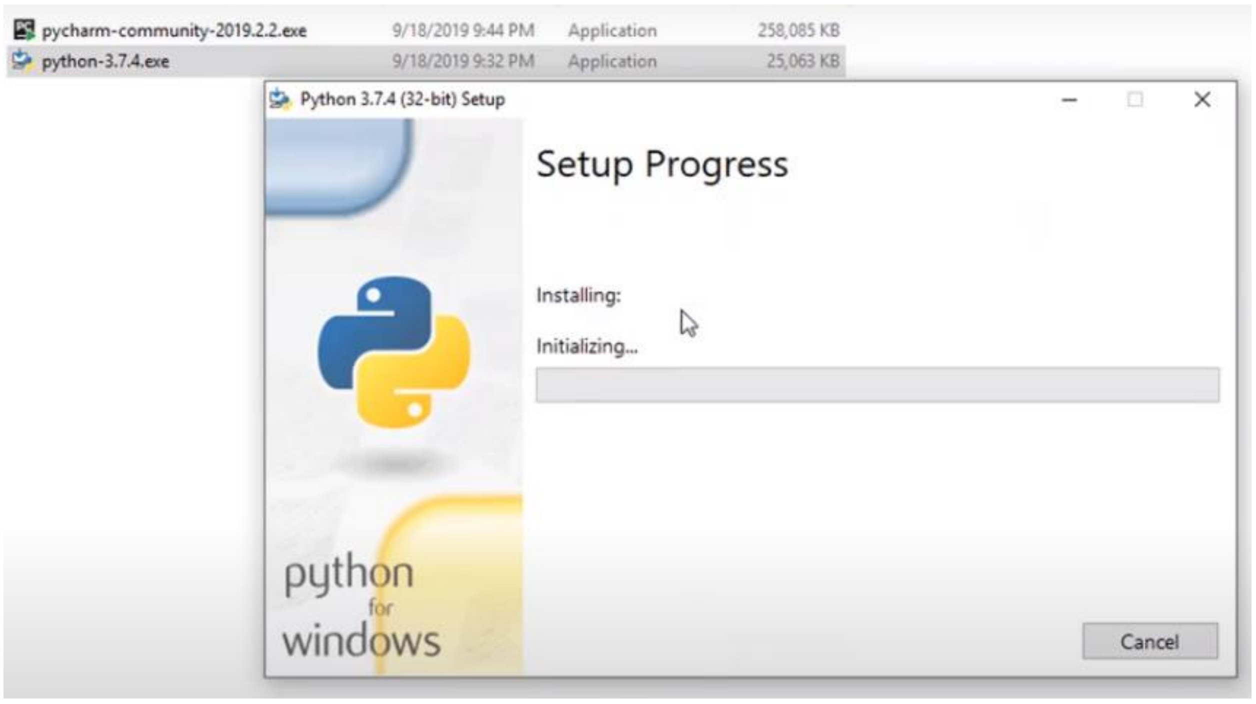
Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes

Step-2: Download pyCharm community version.



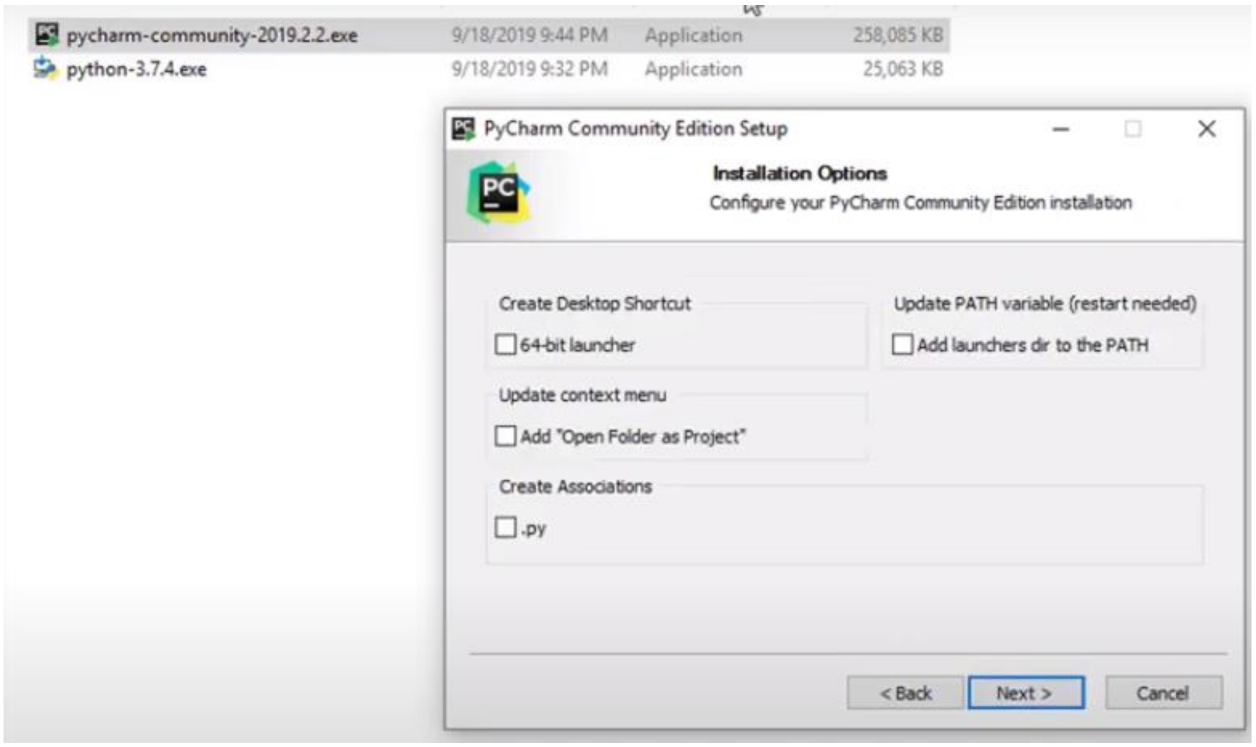
```
Cookies and IP addres
web content, resolve
personalized experien
collects your IP addr

=-=-=-=-=-=-=-=-=-=-=
|
| JetBrains may use c
| collect individual
| personalized offers
| Policy and the Term
| third-party service
| my consent at any t
|
| [Y]es, I agree
|
=-=-=-=-=-=-=-=-=-=-=

~ root#
```

Step-3: python 3.7.4 setup process



pycharm-community-2019.2.2.exe	9/18/2019 9:44 PM	Application	258,085 KB
python-3.7.4.exe	9/18/2019 9:32 PM	Application	25,063 KB

Step-4: pyCharm setup and select the option (64-bit... and .py)



pycharm-community-2019.2.2.exe	9/18/2019 9:44 PM	Application	258,085 KB
python-3.7.4.exe	9/18/2019 9:32 PM	Application	25,063 KB

PyCharm Community Edition Setup

Installation Options

Configure your PyCharm Community Edition installation

Create Desktop Shortcut

☐ 64-bit launcher

Update PATH variable (restart needed)

☐ Add launchers dir to the PATH

Update context menu

☐ Add "Open Folder as Project"

Create Associations

☐ .py

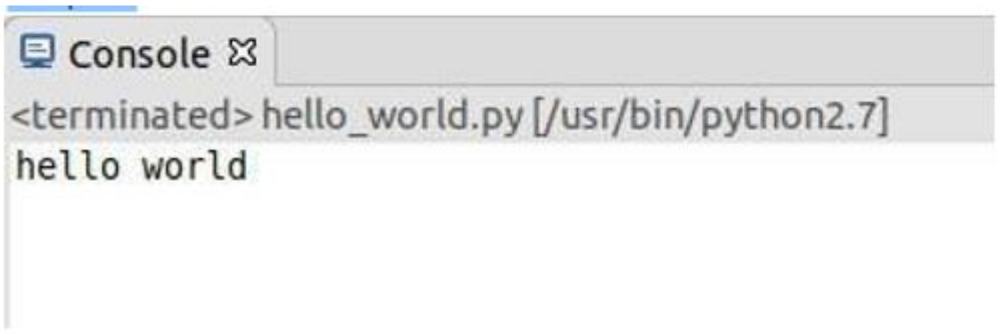
4. Exercises:

Exercise 4.1.2: Write a Hello World program

Ans:

```
print('hello world')
```

Output:



Console

```
<terminated> hello_world.py [/usr/bin/python2.7]
hello world
```

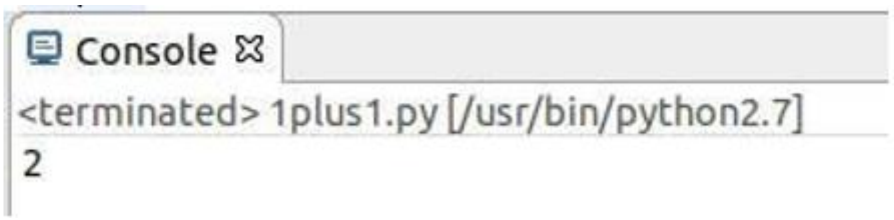
Exercise 4.1.3: Compute 1+1

Ans:

a=1+1

print(a)

output:



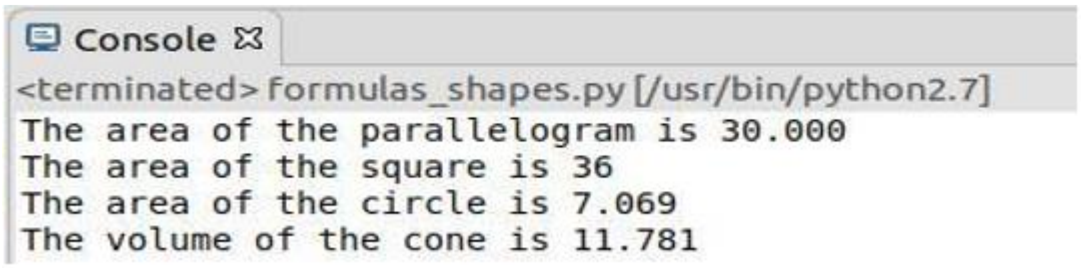
Console

```
<terminated> 1plus1.py [/usr/bin/python2.7]
2
```

Exercise 4.1.4: Type in program text

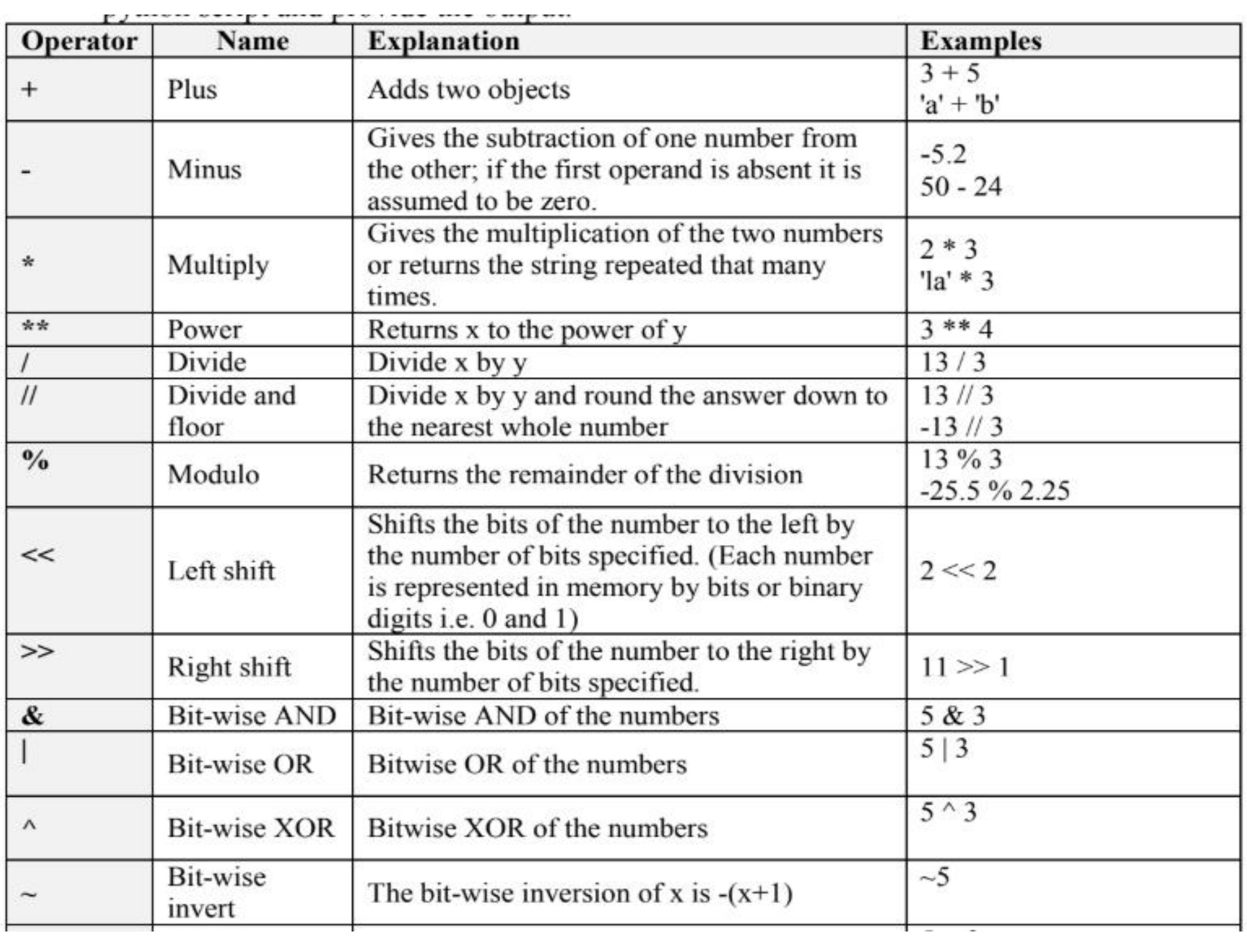
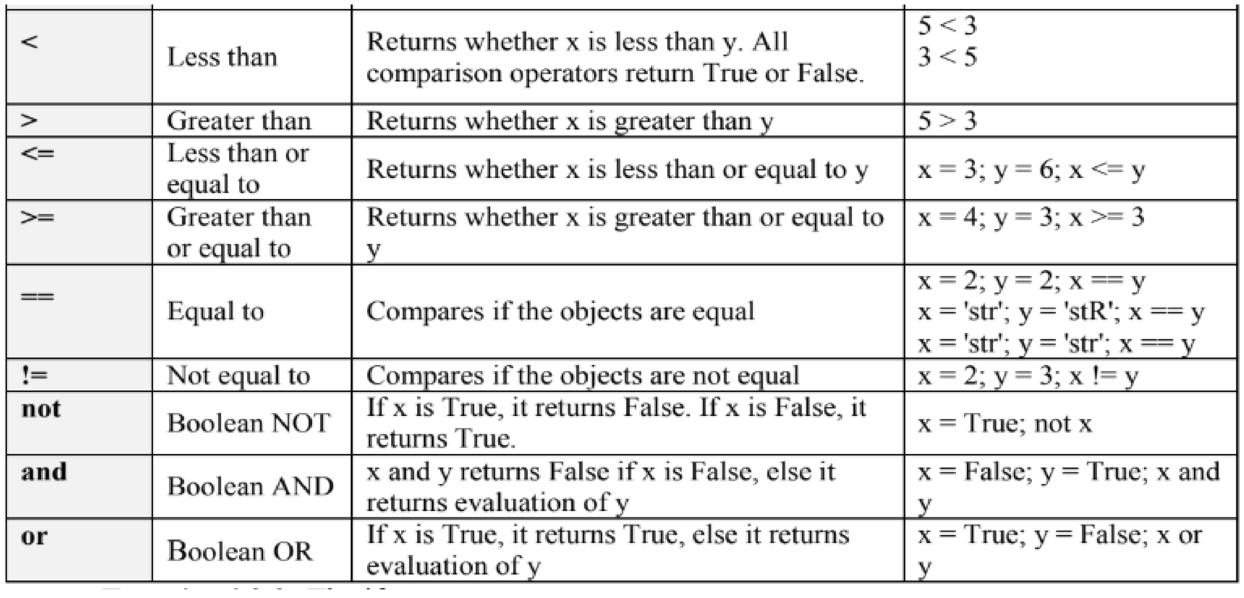
```
h = 5.0 # height

r = 1.5 # radius

 b = 6.0 #width

 area_parallelogram = h*b

 print ('The area of the parallelogram is %.3f' % area_parallelogram)

 area_square = b**2

print ('The area of the square is %g' % area_square)

 area_circle = 3.1416*r**2 print ('The area of the circle is %.3f' %
area_circle)

volume_cone = 1.0/3*3.1416*r**2*h

print ('The volume of the cone is %.3f' % volume_cone)
```

Output:



```
<terminated> formulas_shapes.py [/usr/bin/python2.7]
The area of the parallelogram is 30.000
The area of the square is 36
The area of the circle is 7.069
The volume of the cone is 11.781
```

Exercise 4.2.1: Verify the use of the following operator. Execute the example code in python script and provide the output.

Operator	Name	Explanation	Examples
+	Plus	Adds two objects	3 + 5 'a' + 'b'
-	Minus	Gives the subtraction of one number from the other; if the first operand is absent it is assumed to be zero.	-5.2 50 - 24
*	Multiply	Gives the multiplication of the two numbers or returns the string repeated that many times.	2 * 3 'la' * 3
**	Power	Returns x to the power of y	3 ** 4
/	Divide	Divide x by y	13 / 3
//	Divide and floor	Divide x by y and round the answer down to the nearest whole number	13 // 3 -13 // 3
%	Modulo	Returns the remainder of the division	13 % 3 -25.5 % 2.25
<<	Left shift	Shifts the bits of the number to the left by the number of bits specified. (Each number is represented in memory by bits or binary digits i.e. 0 and 1)	2 << 2
>>	Right shift	Shifts the bits of the number to the right by the number of bits specified.	11 >> 1
&	Bit-wise AND	Bit-wise AND of the numbers	5 & 3
|	Bit-wise OR	Bitwise OR of the numbers	5 | 3
^	Bit-wise XOR	Bitwise XOR of the numbers	5 ^ 3
~	Bit-wise invert	The bit-wise inversion of x is -(x+1)	~5
<	Less than	Returns whether x is less than y. All comparison operators return True or False.	5 < 3 3 < 5
>	Greater than	Returns whether x is greater than y	5 > 3
<=	Less than or equal to	Returns whether x is less than or equal to y	x = 3; y = 6; x <= y
>=	Greater than or equal to	Returns whether x is greater than or equal to y	x = 4; y = 3; x >= 3

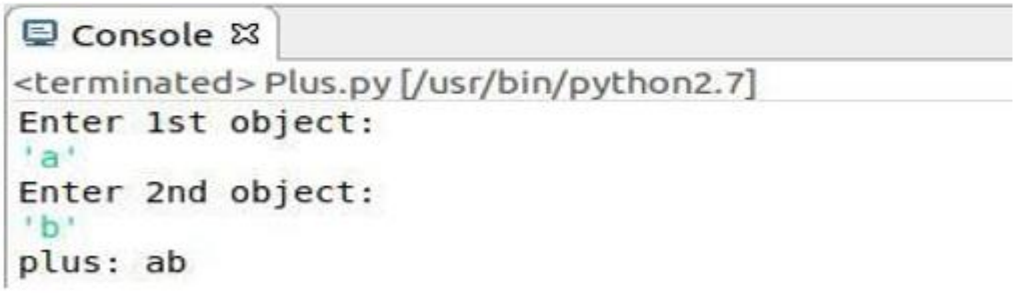
==	Equal to	Compares if the objects are equal	x = 2; y = 2; x == y
			x = 'str'; y = 'stR'; x == y
			x = 'str'; y = 'str'; x == y

!=	Not equal to	Compares if the objects are not equal	x = 2; y = 3; x != y
not	Boolean NOT	If x is True, it returns False. If x is False, it returns True.	x = True; not x
and	Boolean AND	x and y returns False if x is False, else it returns evaluation of y	x = False; y = True; x and y
or	Boolean OR	If x is True, it returns True, else it returns evaluation of y	x = True; y = False; x or y

Ans:

plus (+) operator:

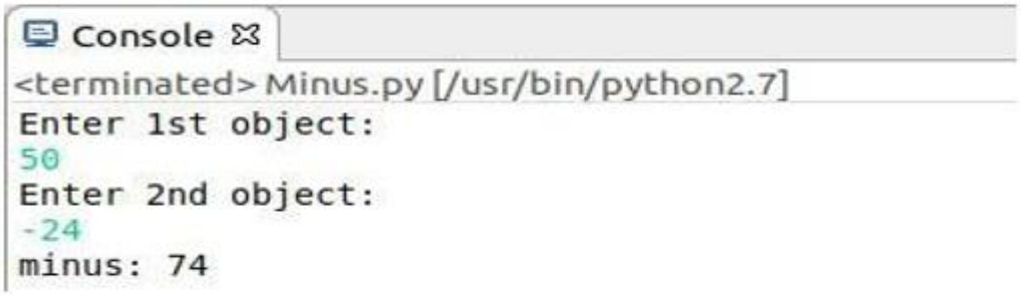
```
a= input('Enter 1st object:\n');
 b= input('Enter 2nd object:\n');
 plus = a+b
 print 'plus:',plus
```



```
Console
<terminated> Plus.py [/usr/bin/python2.7]
Enter 1st object:
'a'
Enter 2nd object:
'b'
plus: ab
```

Minus (-) operator:

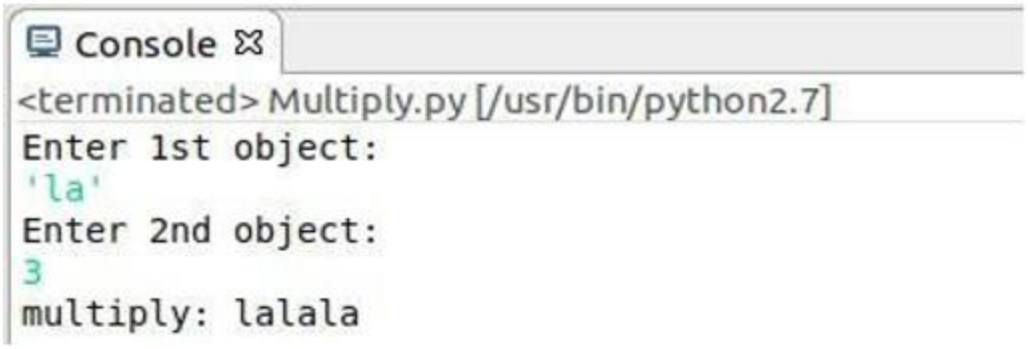
```
a=input('Enter 1st object:\n');
 b=input('Enter 2nd object:\n');
minus = a-b
print 'minus:’, minus
```



```
Console
<terminated> Minus.py [/usr/bin/python2.7]
Enter 1st object:
50
Enter 2nd object:
-24
minus: 74
```

Multiply (*) operator:

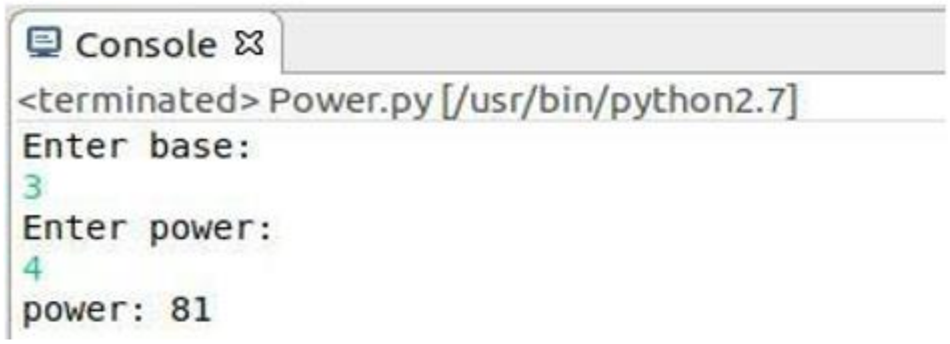
```
a=input('Enter 1st object:\n');
b=input('Enter 2nd object:\n');
multiply=a*b
print 'multiply:’, multiply
```



```
<terminated> Multiply.py [/usr/bin/python2.7]
Enter 1st object:
'la'
Enter 2nd object:
3
multiply: lalala
```

Power(**) operator:

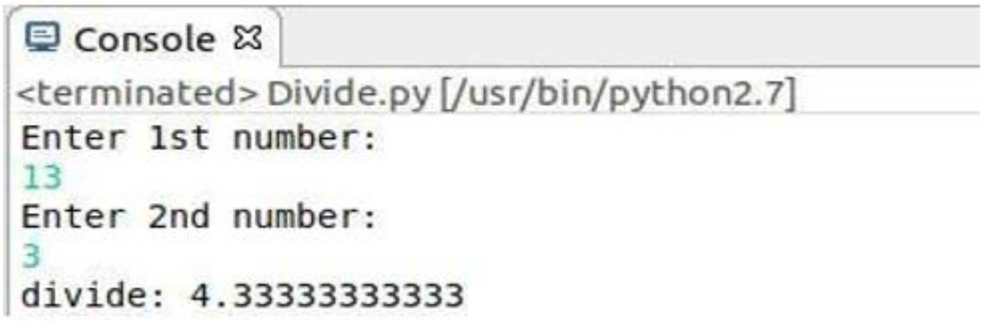
```
a=input('Enter base:\n');
b=input('Enter power:\n');
power=a**b
print 'power:',power
```



```
<terminated> Power.py [/usr/bin/python2.7]
Enter base:
3
Enter power:
4
power: 81
```

Divide (/) operator:

```
a=float(input('Enter 1st number:\n'))
b=float(input('Enter 2nd number:\n'))
divide=a/b
print 'divide:’, divide
```



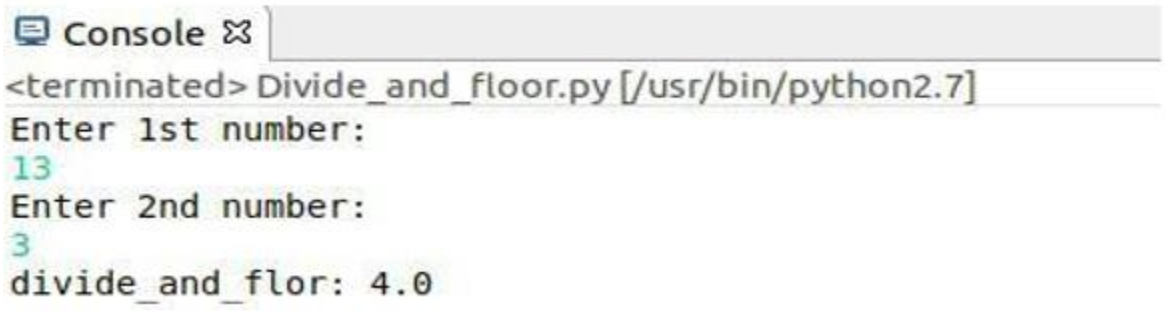
Console

<terminated> Divide.py [/usr/bin/python2.7]

```
Enter 1st number:
13
Enter 2nd number:
3
divide: 4.33333333333
```

Divide and floor (//)operator:

```
a=float(input('Enter 1st number:\n'))
b=float(input('Enter 2nd number:\n'))
divide_and_flor=a//b
print 'divide_and_flor:', divide_and_flor
```



Console

<terminated> Divide_and_floor.py [/usr/bin/python2.7]

```
Enter 1st number:
13
Enter 2nd number:
3
divide_and_flor: 4.0
```

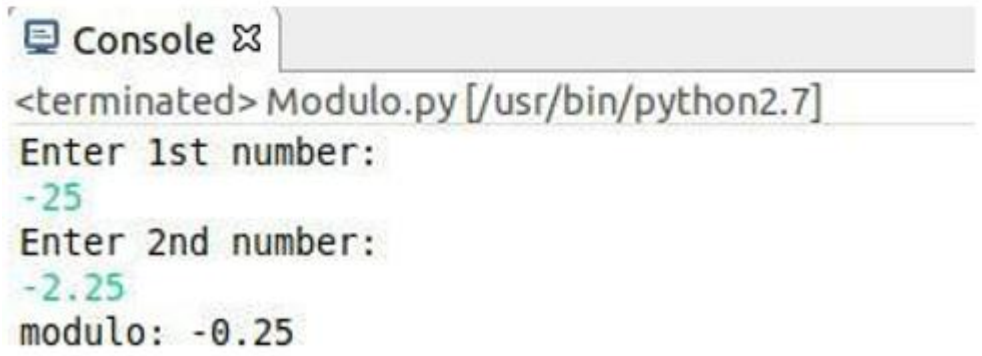
Modulo (%) operator:

a=input('Enter 1st number:\n')

b=input('Enter 2nd number:\n')

modulo=a%b

print 'modulo:', modul



```
<terminated> Modulo.py [/usr/bin/python2.7]
Enter 1st number:
-25
Enter 2nd number:
-2.25
modulo: -0.25
```

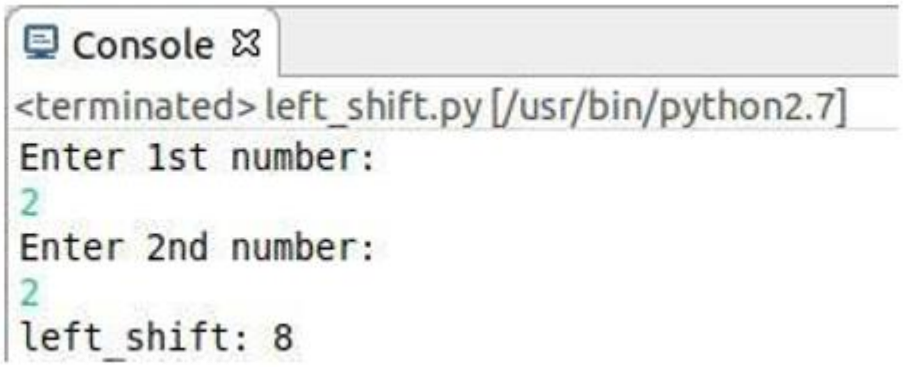
Left shift (<<) operator:

a=input('Enter 1st number:\n')

b=input('Enter 2nd number:\n')

left_shift=a<<b

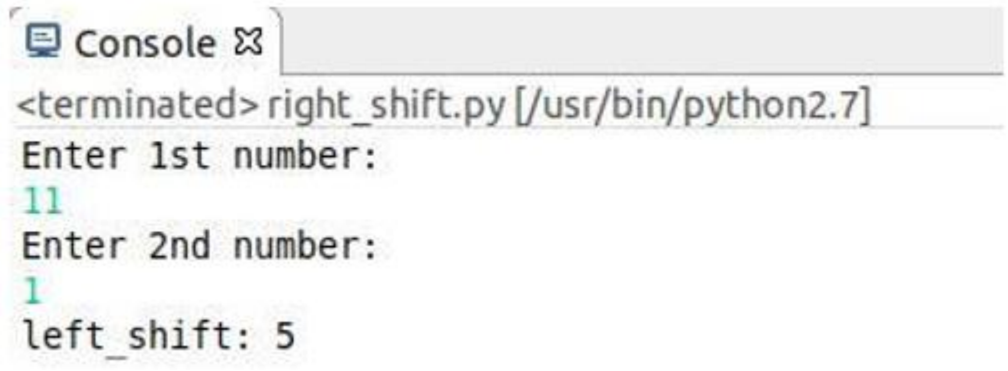
print 'left_shift:', left_shift



```
<terminated> left_shift.py [/usr/bin/python2.7]
Enter 1st number:
2
Enter 2nd number:
2
left_shift: 8
```

Right shift (>>) operator:

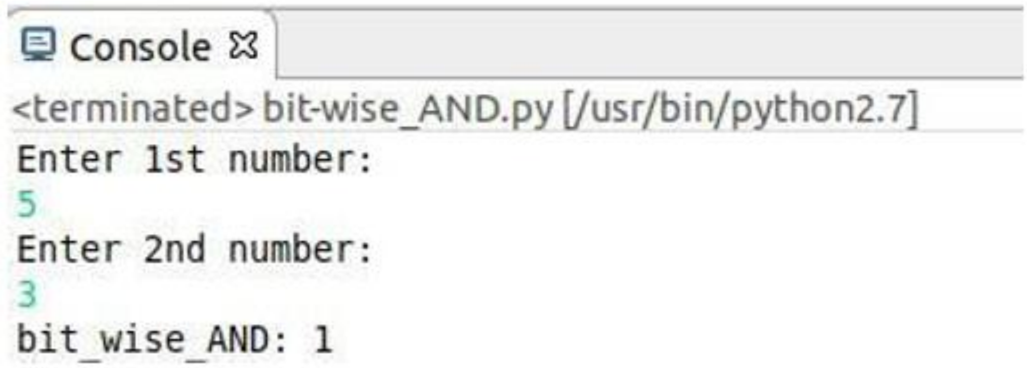
```
a=input('Enter 1st number:\n')

b=input('Enter 2nd number:\n')

left_shift=a>>b

print 'left_shift:',left_shift
```



```
<terminated> right_shift.py [/usr/bin/python2.7]
Enter 1st number:
11
Enter 2nd number:
1
left_shift: 5
```

Bit-wise AND (&) operator:

```
a=input('Enter 1st number:\n')

b=input('Enter 2nd number:\n')

bit_wise_AND=a&b

print 'bit_wise_AND:',bit_wise_AND
```



```
<terminated> bit-wise_AND.py [/usr/bin/python2.7]
Enter 1st number:
5
Enter 2nd number:
3
bit_wise_AND: 1
```

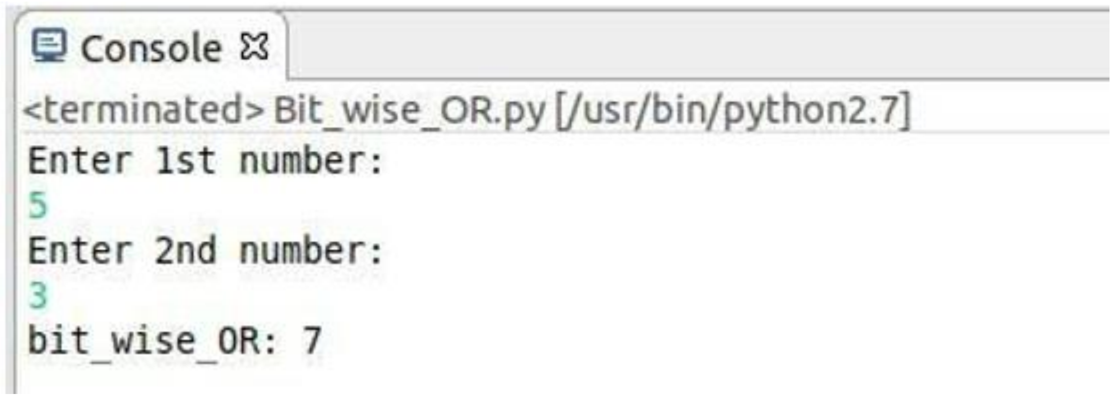
Bit-wise OR (|) operator:

a=input('Enter 1st number:\n')

b=input('Enter 2nd number:\n')

bit_wise_OR= a|b

print 'bit_wise_OR:',bit_wise_OR



Console

```
<terminated> Bit_wise_OR.py [/usr/bin/python2.7]
Enter 1st number:
5
Enter 2nd number:
3
bit_wise_OR: 7
```

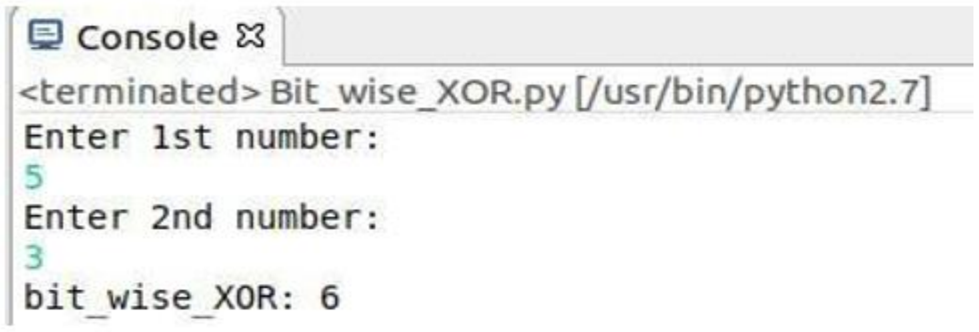
Bit-wise XOR (^) operator:

a=input('Enter 1st number:\n')

b=input('Enter 2nd number:\n')

bit_wise_XOR=a^b

print 'bit_wise_XOR:',bit_wise_XO

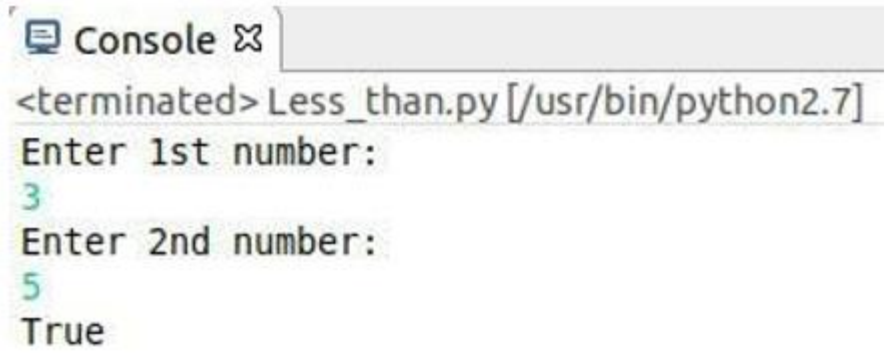


Console

```
<terminated> Bit_wise_XOR.py [/usr/bin/python2.7]
Enter 1st number:
5
Enter 2nd number:
3
bit_wise_XOR: 6
```

Less than (<)operator:

```
a=input('Enter 1st number:\n')

b=input('Enter 2nd number:\n')

if a<b:

        print True

else:

         print False
```

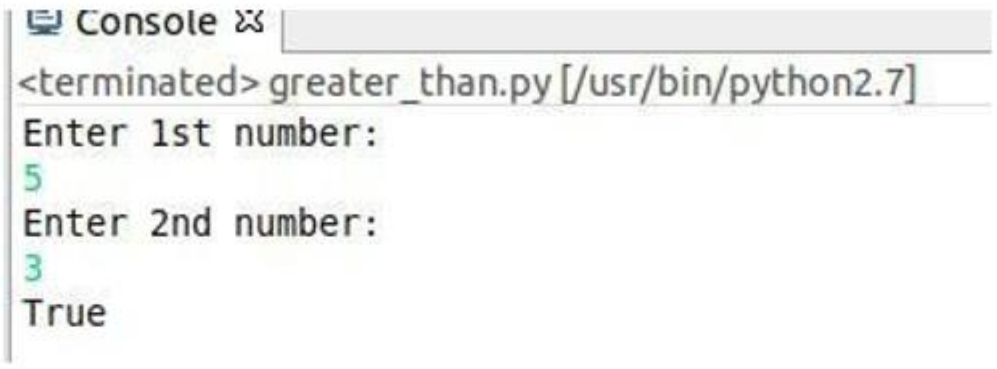


Console

```
<terminated> Less_than.py [/usr/bin/python2.7]
Enter 1st number:
3
Enter 2nd number:
5
True
```

Greater than(>) operator:

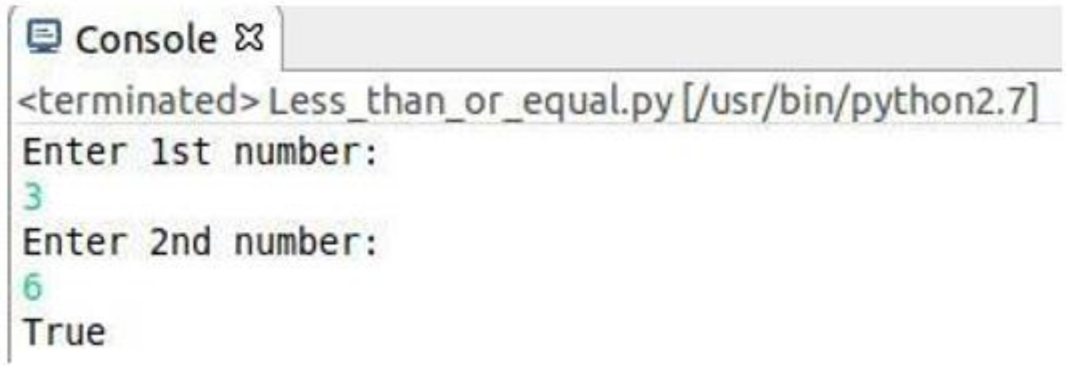
```
a=input('Enter 1st number:\n')

b=input('Enter 2nd number:\n')

if a>b:

        print True

else:

        print False
```



```
Console
<terminated> greater_than.py [/usr/bin/python2.7]
Enter 1st number:
5
Enter 2nd number:
3
True
```

Less than or equal to(<=) operator:

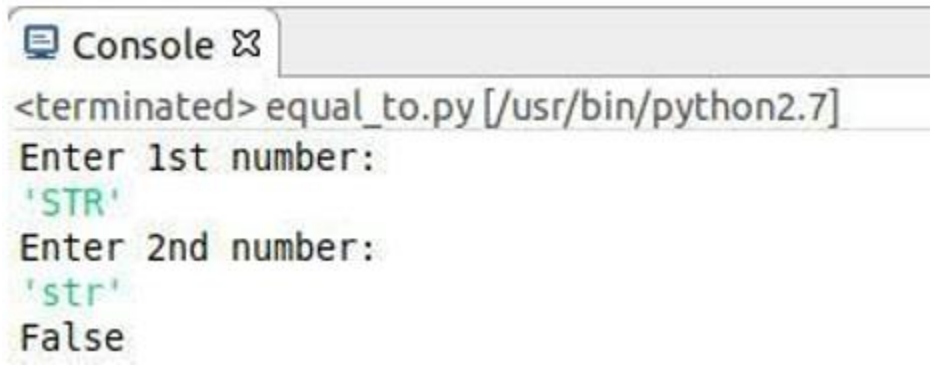
```
a=input('Enter 1st number:\n')
b=input('Enter 2nd number:\n')
if a<=b:
	print True
else:
	print False
```



```
Console
<terminated> Less_than_or_equal.py [/usr/bin/python2.7]
Enter 1st number:
3
Enter 2nd number:
6
True
```

Equal to (==) operator:

```
a=input('Enter 1st number:\n')

b=input('Enter 2nd number:\n')

if a==b:

        print True

else:

         print False
```

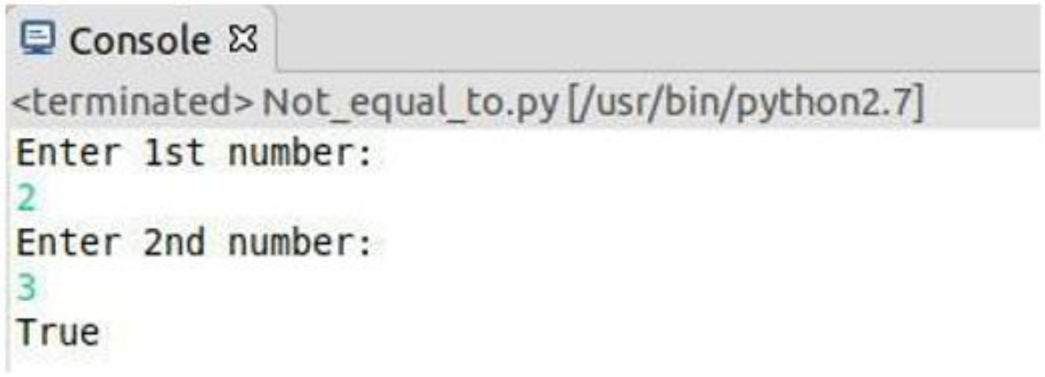


Console

```
<terminated> equal_to.py [/usr/bin/python2.7]
Enter 1st number:
'STR'
Enter 2nd number:
'str'
False
```

Not equal to(!=) operator:

```
a=input('Enter 1st number:\n')

b=input('Enter 2nd number:\n')

if a!=b:

        print True

else:

        print False
```


Console

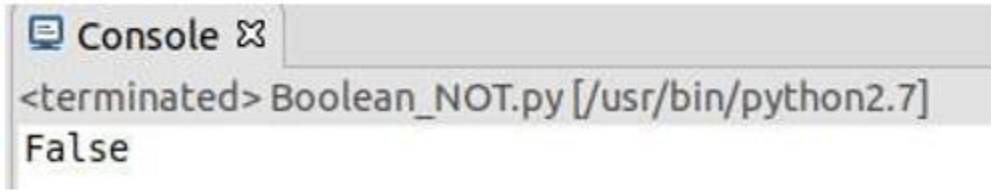
```
<terminated> Not_equal_to.py [/usr/bin/python2.7]
Enter 1st number:
2
Enter 2nd number:
3
True
```

Boolean NOT(not) operator:

from operator import not_

 a=True

print not True


Console

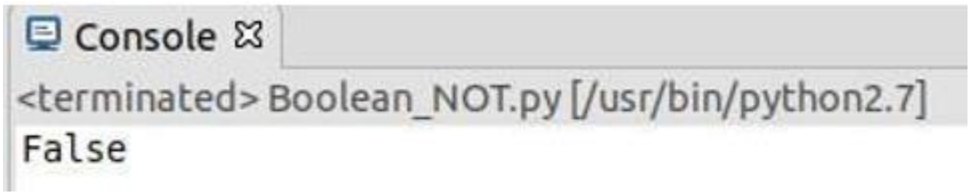
```
<terminated> Boolean_NOT.py [/usr/bin/python2.7]
False
```

Boolean AND(and) operator:

a=True

 b=False

print a and b


Console

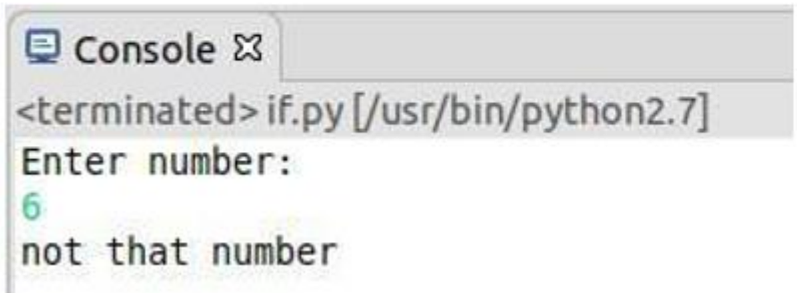
```
<terminated> Boolean_NOT.py [/usr/bin/python2.7]
False
```

Exercise 4.2.2: The if statement

Create a program for taking a number from the user and check if it is the number that you have saved in the code.

Ans:

```
a=input('Enter number:\n')

b=5

if a==b:

        print a

else:

        print "not that number"
```



```
Console
<terminated> if.py [/usr/bin/python2.7]
Enter number:
6
not that number
```

Exercise 4.2.3: The while Statement

Create a program for taking a number from the user and check if it is the number that you have saved in the code. The program run until the user will guess the number.

program:

saved_namber=5

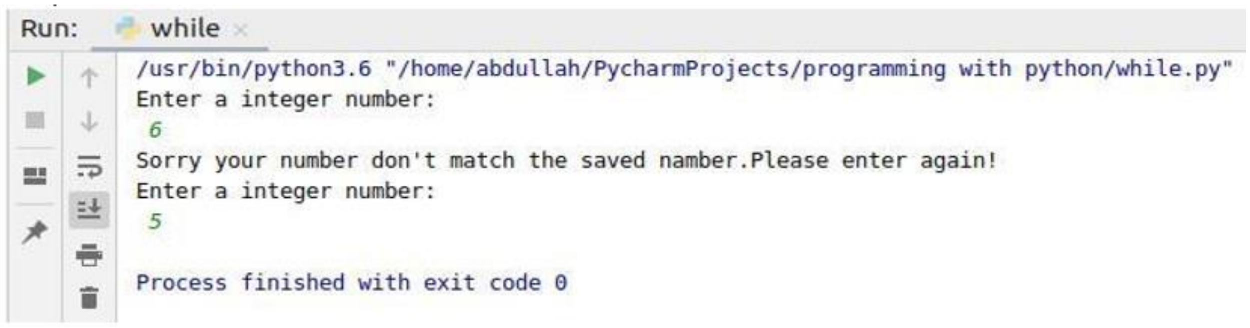
number=int(input('Enter a integer number:\n '))

 while number !=saved_namber:

	print("Sorry your number don't match the saved namber. Please enter again!")

	number= int(input('Enter a integer number:\n '))

Output:



```
/usr/bin/python3.6 "/home/abdullah/PycharmProjects/programming with python/while.py"
Enter a integer number:
 6
Sorry your number don't match the saved namber.Please enter again!
Enter a integer number:
 5

Process finished with exit code 0
```

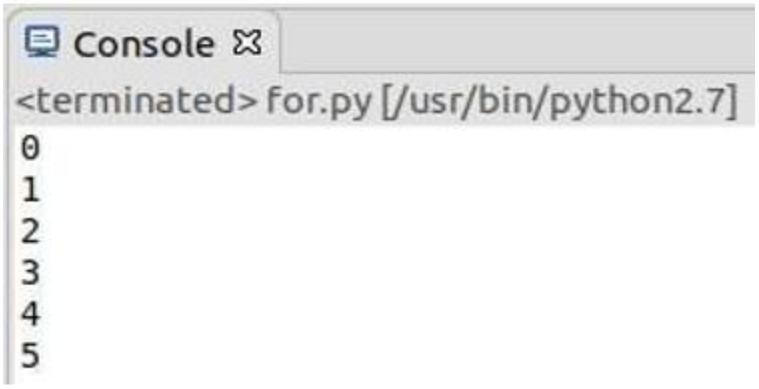
Exercise 4.2.4: The for Statement

Create a program for printing a sequence of numbers.

Ans:

for x in range(6):

print(x)


Console

```
<terminated> for.py [/usr/bin/python2.7]
0
1
2
3
4
5
```

Question 5.1: Explain what is eclipse? And why we use it for programing on python?

Ans:

Eclipse is an integrated development environment (IDE) for developing applications using the Java programming language and other programming languages such as C/C++, Python, PERL, Ruby etc.

We use eclipse for developing python modules.

Question 5.2: Explain three main characteristics of python that you test in the lab?

Ans:

Simple

Easy to Learn

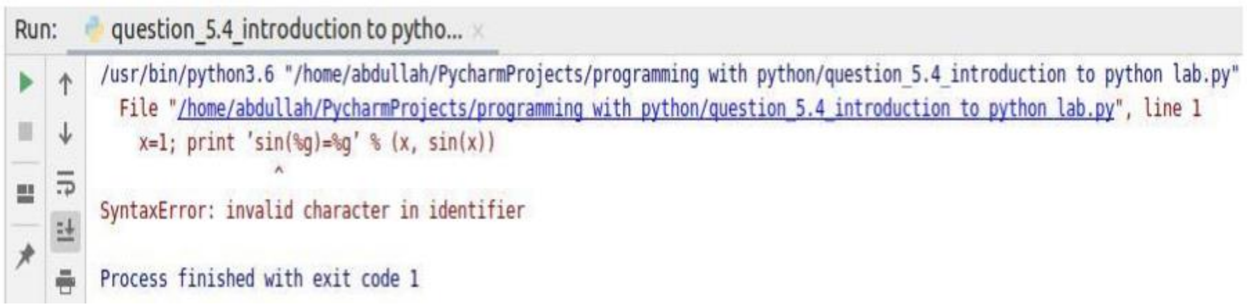
Free and Open Source

Question 5.4: Find error(s) in a program

Suppose somebody has written a simple one-line program for computing sin(1): x=1; print 'sin(%g)=%g' % (x, sin(x)) Create this program and try to run it. What is the problem? Which is the correct code?

Ans:

Program: x=1; print 'sin(%g)=%g' % (x, sin(x)



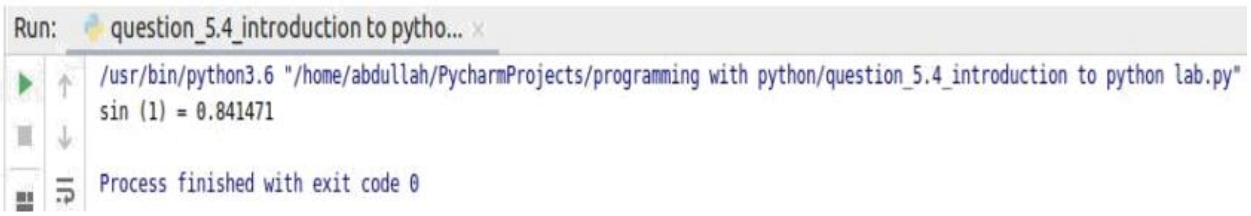
```
Run: question_5.4_introduction to pytho...
/usr/bin/python3.6 "/home/abdullah/PycharmProjects/programming with python/question_5.4_introduction to python lab.py"
  File "/home/abdullah/PycharmProjects/programming with python/question_5.4_introduction to python lab.py", line 1
    x=1; print 'sin(%g)=%g' % (x, sin(x))
                  ^
SyntaxError: invalid character in identifier

Process finished with exit code 1
```

Correct code:

import math as m

x=1

print("sin (%g) = %g"%(x, m.sin(x)))



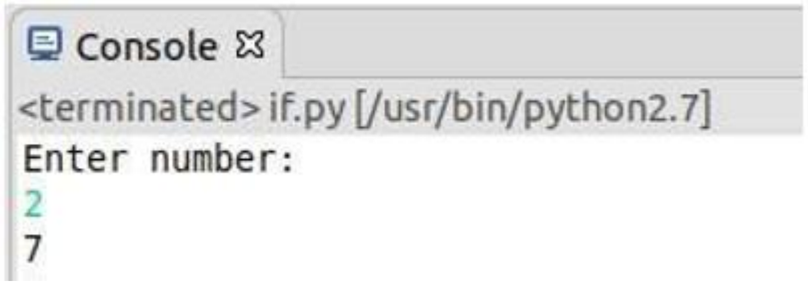
```
Run: question_5.4_introduction to pytho...
/usr/bin/python3.6 "/home/abdullah/PycharmProjects/programming with python/question_5.4_introduction to python lab.py"
sin (1) = 0.841471

Process finished with exit code 0
```

Question 5.5: Create a python program that combines at least 4 operators and one statement (if, while or for)

Ans:

```
a=input('Enter number:\n')

b=5;

if a>b:

        print a-b

 else:

        print a+b
```

Output:



```
Console
<terminated> if.py [/usr/bin/python2.7]
Enter number:
2
7
```

Discussion:

I learned many things from this lab. This lab helps me to understand the basic of python programming. I also know how to download the python and python IDE(pyCharm) and how to setup the python environment. I learn how python programming works, structure and many things. I also learn how to run a python program. Also, learn about variables, operators, keywords in python programming. This was an interesting lab, I can be able to run successfully all the above program as screenshot given above