



Mawlana Bhashani Science And Technology University

Lab-Report

Lab Report No: 01

Lab Report Name: Mininet lab 01

Group member ID: IT-18013 and IT-18028

Date of Performance: 08-01-2021

Date of Submission: 08-01-2021

Submitted by

Name: Anjom Nour Anika

ID: IT-18013.

3rd Year 2nd Semester

Session: 2017-2018

Dept. of ICT, MBSTU

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Lab report name: Introduction to Python

Objectives:

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

Theory:

Definition of python: Python is a multiparadigm, general-purpose, interpreted, high-level programming language. Python allows programmers to use different programming styles to create simple or complex programs, get quicker results and write code almost as if speaking in a human language. Some of the popular systems and applications that have employed Python during development include Google Search, YouTube, BitTorrent, Google App Engine, Eve Online, Maya and iRobot machines

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

1. Easy

When we say the word 'easy', we mean it in different contexts.

a. Easy to Code

Python is very **easy to code** as compared to other popular languages like Java and C++.

Anyone can learn **Basic Python syntax** in just a few hours. Thus, it is programmer-friendly.

b. Easy to Read

Being a high-level language, Python code is quite like English. Looking at it, you can tell what the code is supposed to do.

Also, since it is **dynamically-typed**, it mandates indentation. This aids readability.

2. Expressive

First, let's learn what is expressiveness. Suppose we have two languages A and B, and all programs that can be made in A can be made in B using local transformations.

However, there are some programs that can be made in B, but not in A, using local transformations. Then, B is said to be more expressive than A.

Python provides us with a myriad of constructs that help us focus on the solution rather than on the syntax.

This is one of the outstanding python features that tell you why you should learn Python.

3. Free and Open-Source

Firstly, Python is **freely available**.

Secondly, it is **open-source**. This means that its source code is available to the public. You can download it, change it, use it, and distribute it.

This is called **FLOSS(Free/Libre and Open Source Software)**.

As the Python community, we're all headed toward one goal- an ever-bettering Python.

4. High-Level

Python is a high-level language. This means that as programmers, we don't need to remember the system architecture.

Also, we need not manage memory. This makes it more **programmer-friendly** and is one of the key python features.

5. Portable

Let's assume you've written a Python code for your Windows machine. Now, if you want to run it on a Mac, you don't need to make changes to it for the same.

In other words, you can take one code and run it on any machine. This makes Python a **portable language**.

However, you must avoid any system-dependent features in this case.

6. Interpreted

If you're familiar with any languages like C++ or Java, you must first compile it, and then run it. But in Python, there is no need to compile it.

Internally, its source code is converted into an immediate form called **bytecode**.

So, all you need to do is to run your Python code without worrying about linking to libraries, and a few other things.

By interpreted, we mean the source code is executed line by line, and not all at once. Because of this, it is **easier to debug your code**.

Also, interpreting makes it just slightly slower than Java, but that does not matter compared to the benefits it offers.

7. Object-Oriented

A programming language that can model the real world is said to be object-oriented. It focuses on objects and combines data and functions.

Contrarily, a procedure-oriented language revolves around functions, which are code that can be reused.

Python supports both **procedure-oriented** and **object-oriented programming** which is one of the key python features.

It also supports multiple inheritance, unlike Java.

A class is a blueprint for such an object. It is an abstract data type and holds no values.

8. Extensible

If needed, you can write some of your Python code in other languages like C++. This makes Python an extensible language, meaning that it can be extended to other languages.

9. Embeddable

We just saw that we can put code in other languages in our Python source code.

However, it is also possible to put our Python code in a source code in a different language like C++.

This allows us to integrate scripting capabilities into our program of the other language.

10. Large Standard Library

Python downloads with a large library that you can use so you don't have to write your own code for every single thing.

There are libraries for regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and a lot of other functionality.

11. GUI Programming

Software is not user-friendly until its GUI is made. A user can easily interact with the software with a GUI.

Python offers various libraries for making Graphical user interface for your applications.

For this, you can use Tkinter, wxPython or JPython. These toolkits allow you for easy and fast development of GUI.

12. Dynamically Typed

Python is dynamically-typed. This means that the type for a value is decided at runtime, not in advance.

This is why we don't need to specify the type of data while declaring it.

This is all about the features of the python programming language.

Methodology:

There are basically three types of methods in Python:

Instance Method

Class Method

Static Method

Let's talk about each method in detail.

Instance Methods

The purpose of instance methods is to set or get details about instances (objects), and that is why they're known as instance methods. They are the most common type of methods used in a Python class.

Any method you create inside a class is an instance method unless you specially specify Python otherwise. Let's see how to create an instance method:

```
class My_class:

    def instance_method(self):

        return "This is an instance method."
```

Class Methods

The purpose of the class methods is to set or get the details (status) of the class. That is why they are known as class methods. They can't access or modify specific instance data. They are bound to the class instead of their objects. Two important things about class methods:

In order to define a class method, you have to specify that it is a class method with the help of the @classmethod decorator

Class methods also take one default parameter- cls, which points to the class. Again, this not mandatory to name the default parameter “cls”. But it is always better to go with the conventions

Static Methods

Static methods cannot access the class data. In other words, they do not need to access the class data. They are self-sufficient and can work on their own. Since they are not attached to any class attribute, they cannot get or set the instance state or class state.

In order to define a static method, we can use the @staticmethod decorator (in a similar way we used @classmethod decorator). Unlike instance methods and class methods, we do not need to pass any special or default parameters. Let's look at the implementation:

```
class My_class:

    @staticmethod

    def static_method():

        return "This is a static method."
```

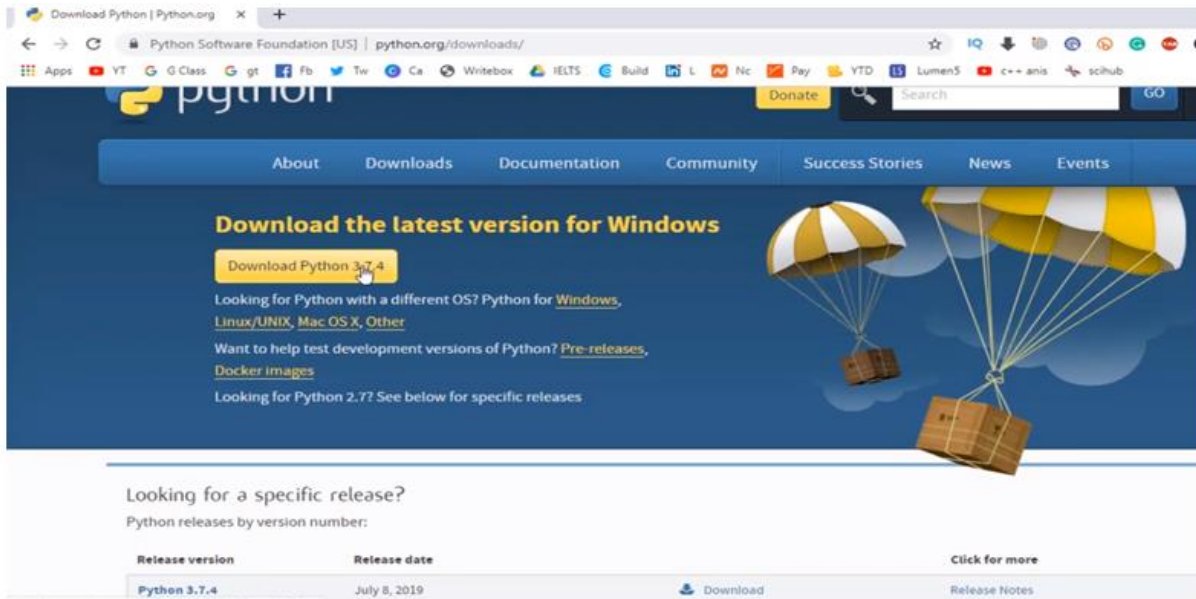
Then we need next setup.

Setup of Python Environment

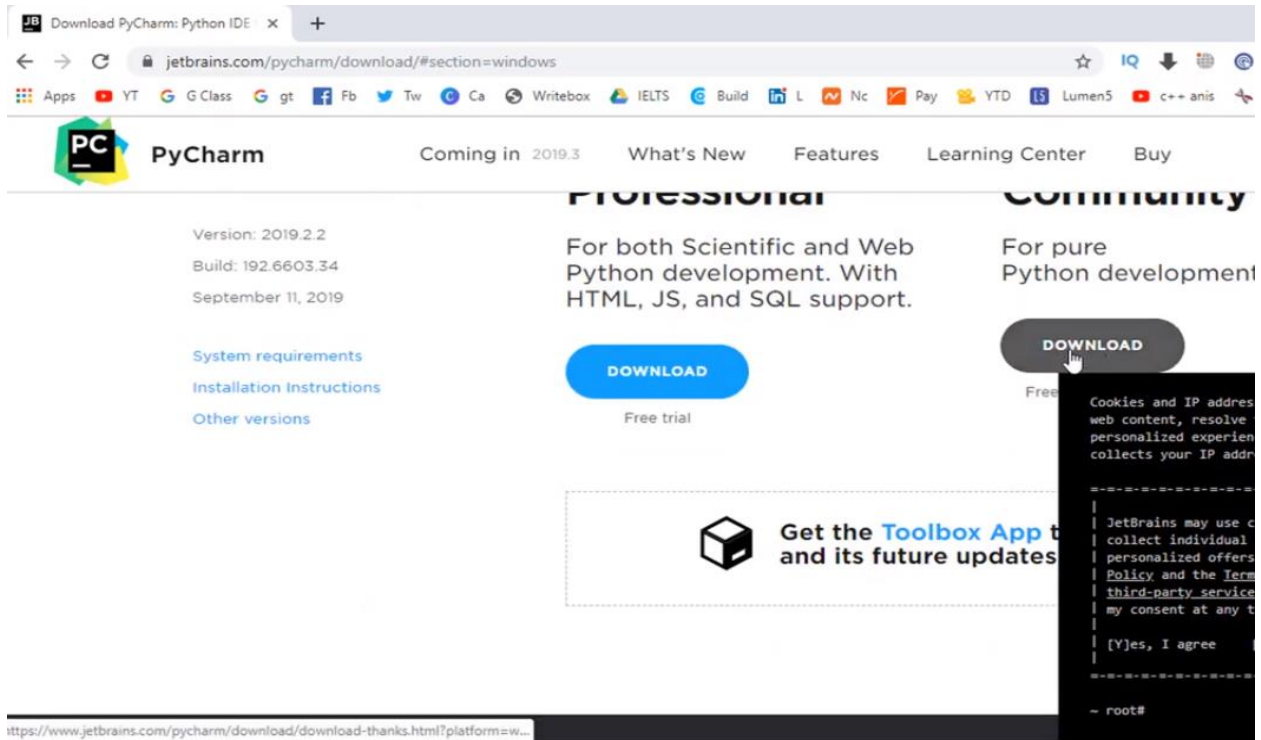
First install: 1. Python

2. pyCharm

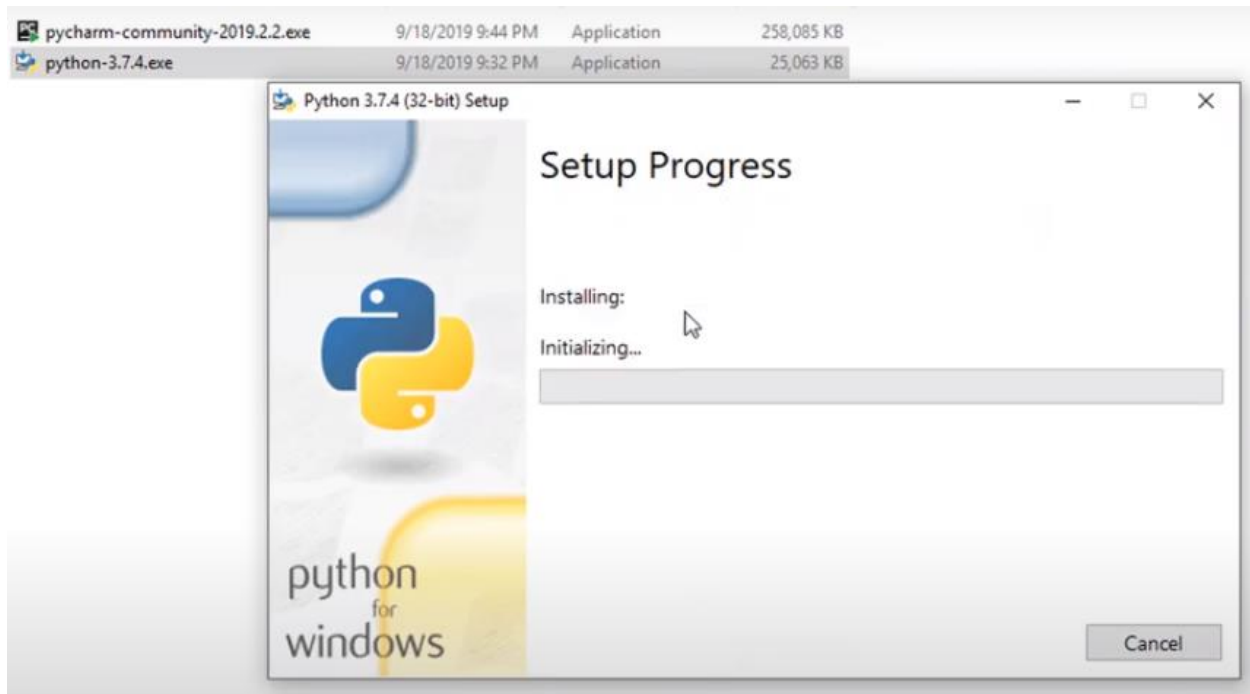
Step-1: Download the latest version python 3.7.4



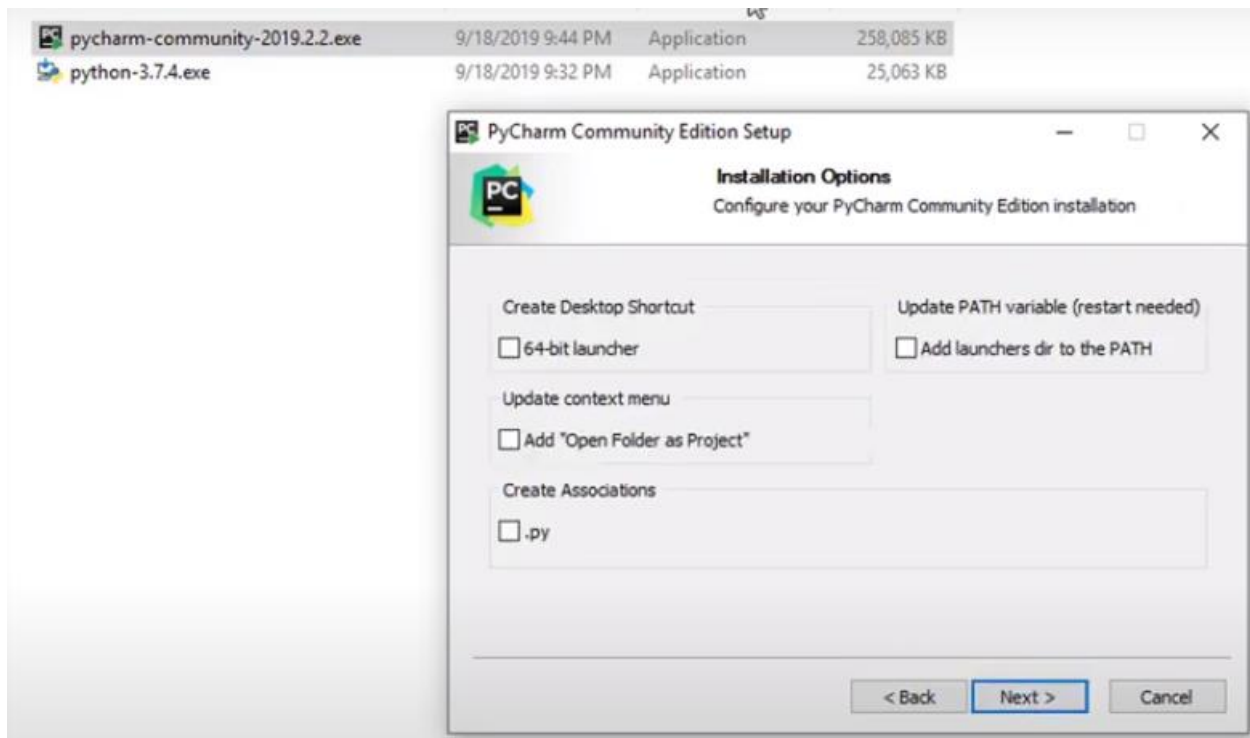
Step-2: Download pyCharm community version.



Step-3: python 3.7.4 setup process



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



4. Exercises:

Exercise 4.1.2: Write a Hello World program

Ans:

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

Output:



Exercise 4.1.3: Compute 1+1

Ans:

```
a=1+1
```

```
print(a)
```

output:

A screenshot of a terminal window. The title bar says "Console" with a close button. The terminal text shows "<terminated> 1plus1.py [/usr/bin/python2.7]" followed by the output "2".

```
<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
```

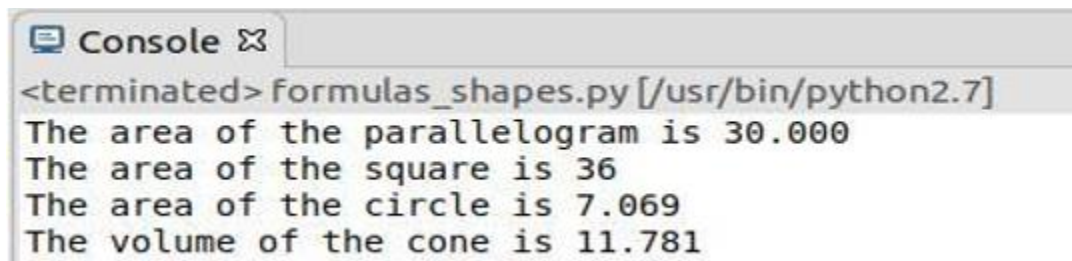
```
print ('The area of the square is %g' % area_square)
```

```
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:

A screenshot of a console window titled "Console" with a close button. The window shows the output of a Python script named "formulas_shapes.py" executed from the directory "/usr/bin/python2.7". The output consists of four lines: "The area of the parallelogram is 30.000", "The area of the square is 36", "The area of the circle is 7.069", and "The volume of the cone is 11.781".

```
<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

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:

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

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

Power () operator:**

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

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

Divide (/) operator:

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

```
Console ✕
<terminated> Divide.py [/usr/bin/python2.7]
Enter 1st number:
13
Enter 2nd number:
3
divide: 4.333333333333
```

Divide and floor (//)operator:

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

```
Q =float (input ('Enter 2nd number:\n'))
```

```
divide_and_flor=P//Q
```

```
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
```



```
Console 
<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

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

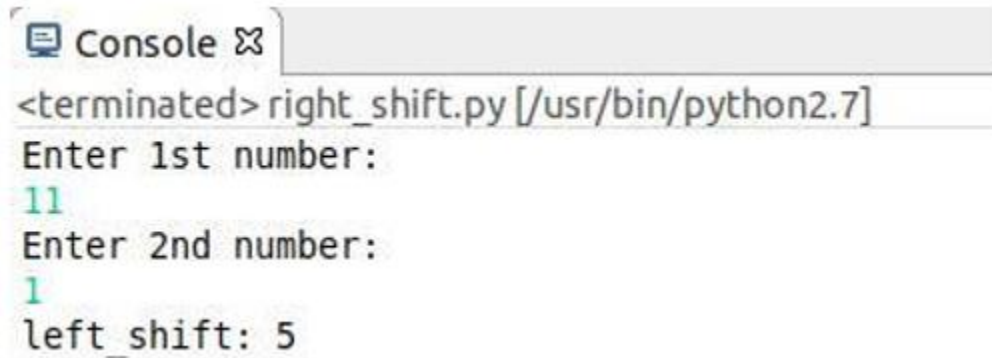
Right shift (>>) operator:

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

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

left_shift=a>>b

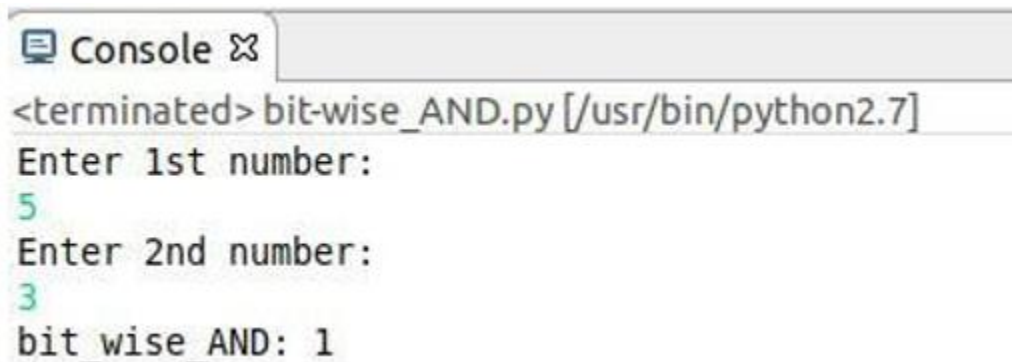
```
print 'left_shift:', left_shift
```



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

Bit-wise AND (&) operator:

```
p=input ('Enter 1st number:\n')  
q =input ('Enter 2nd number:\n')  
bit_wise_AND=p& q  
print 'bit_wise_AND:', bit_wise_AND
```



```
Console ✖  
<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
```



A screenshot of a console window titled "Console" with a close button. The window shows the execution of a Python script named "Bit_wise_OR.py" using Python 2.7. The prompt "<terminated>" is at the top. The script prompts the user to "Enter 1st number:" and "Enter 2nd number:". The user enters "5" and "3" respectively. The final output is "bit_wise_OR: 7".

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

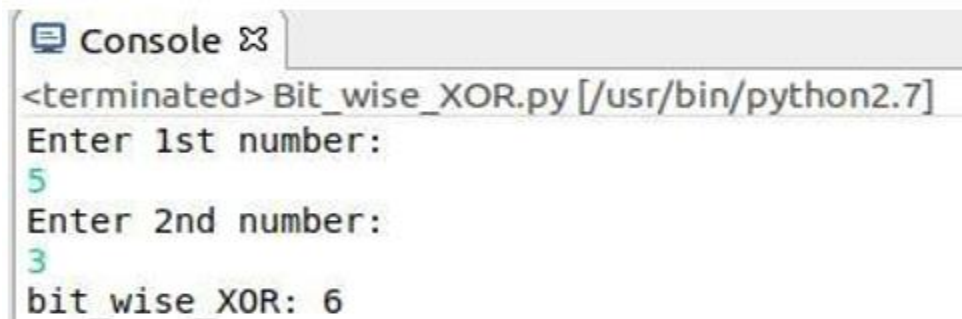
Bit-wise XOR (^) operator:

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

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

```
bit_wise_XOR=a^b
```

```
print 'bit_wise_XOR:', bit_wise_XO
```



A screenshot of a console window titled "Console" with a close button. The window shows the execution of a Python script named "Bit_wise_XOR.py" using Python 2.7. The prompt "<terminated>" is at the top. The script prompts the user to "Enter 1st number:" and "Enter 2nd number:". The user enters "5" and "3" respectively. The final output is "bit_wise_XOR: 6".

```
<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
```

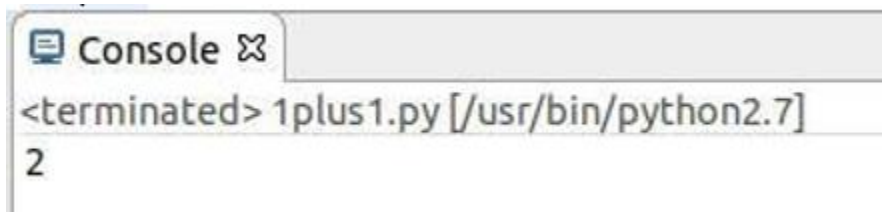
Exercise 4.1.3: Compute 1+1

Ans:

```
a=1+1
```

```
print(a)
```

output:

A screenshot of a terminal window. The title bar says "Console" with a close button. The terminal text shows "<terminated> 1plus1.py [/usr/bin/python2.7]" on the first line and "2" on the second line, indicating the output of the script.

```
<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
```

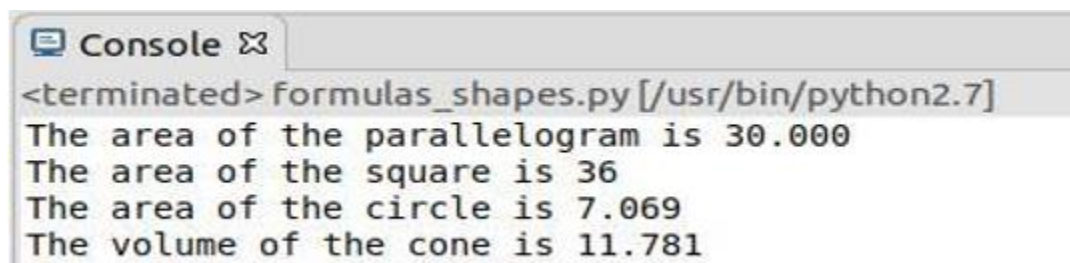
```
print ('The area of the square is %g' % area_square)
```

```
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:



The screenshot shows a terminal window titled "Console" with a close button. The prompt is "<terminated> formulas_shapes.py [/usr/bin/python2.7]". The output consists of four lines: "The area of the parallelogram is 30.000", "The area of the square is 36", "The area of the circle is 7.069", and "The volume of the cone is 11.781".

```
<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
```

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

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:

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

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

Power () operator:**


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

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

Divide (/) operator:

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



```
Console 
<terminated> Divide.py [/usr/bin/python2.7]
Enter 1st number:
13
Enter 2nd number:
3
divide: 4.333333333333
```

Divide and floor (//)operator:

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

```
Q =float (input ('Enter 2nd number:\n'))
```

```
divide_and_flor=P//Q
```

```
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
```

```
Console 
<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

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

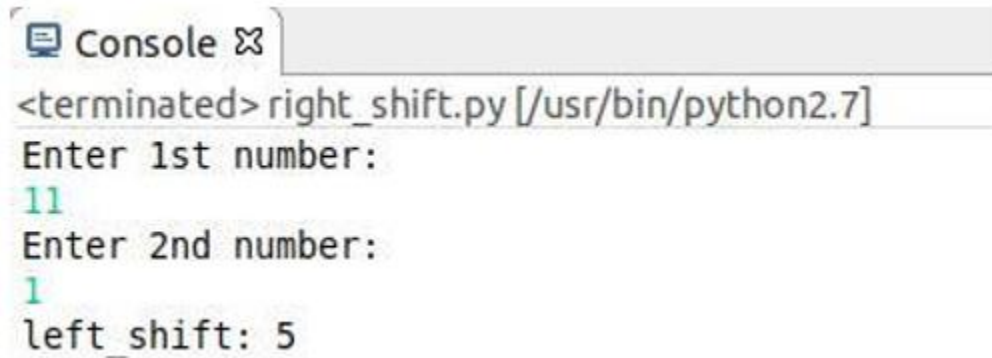
Right shift (>>) operator:

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

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

left_shift=a>>b

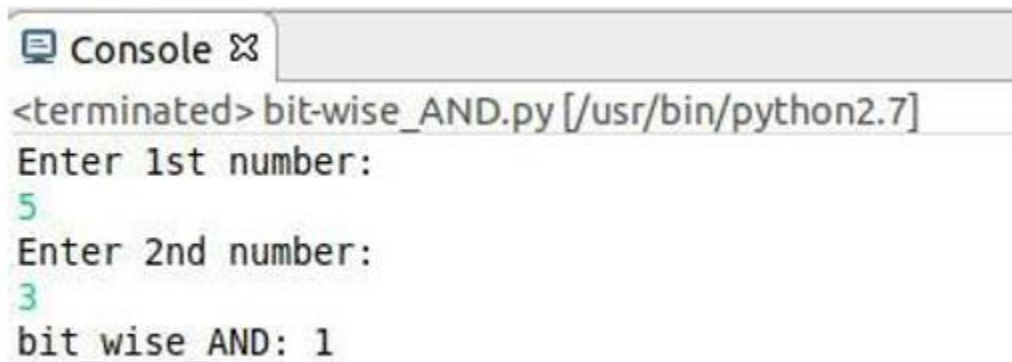
```
print 'left_shift:', left_shift
```



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

Bit-wise AND (&) operator:

```
p=input ('Enter 1st number:\n')
q =input ('Enter 2nd number:\n')
bit_wise_AND=p& q
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
```



A screenshot of a console window titled "Console" with a close button. The window shows the execution of a Python script named "Bit_wise_OR.py" located at "/usr/bin/python2.7". The script prompts the user to "Enter 1st number:" and "Enter 2nd number:". The user enters "5" and "3" respectively. The script then outputs "bit_wise_OR: 7".

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

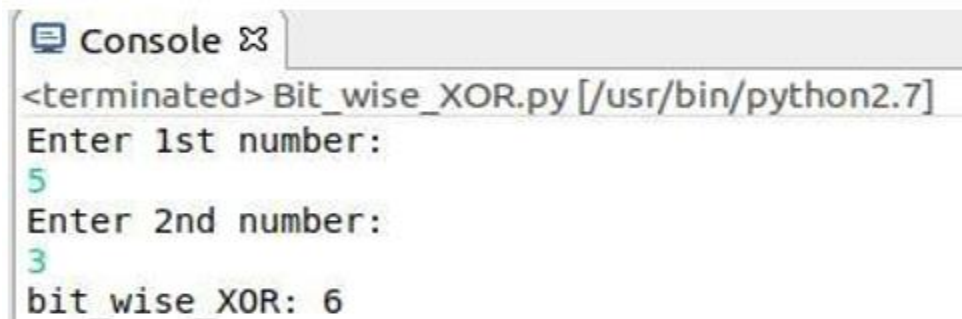
Bit-wise XOR (^) operator:

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

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

```
bit_wise_XOR=a^b
```

```
print 'bit_wise_XOR:', bit_wise_XO
```



A screenshot of a console window titled "Console" with a close button. The window shows the execution of a Python script named "Bit_wise_XOR.py" located at "/usr/bin/python2.7". The script prompts the user to "Enter 1st number:" and "Enter 2nd number:". The user enters "5" and "3" respectively. The script then outputs "bit_wise_XOR: 6".

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

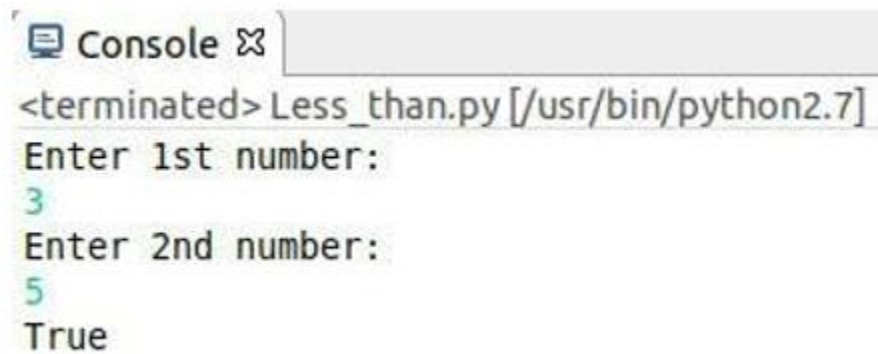
Less than (<) operator:

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

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

```
if a<b:
```

```
        print True
else:
    print False
```



The screenshot shows a terminal window titled 'Console'. The prompt is '<terminated> Less_than.py [/usr/bin/python2.7]'. The program prompts the user to 'Enter 1st number:', and the user enters '3'. Then it prompts 'Enter 2nd number:', and the user enters '5'. Finally, the program outputs 'True'.

```
<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 (\leq) 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:

```
e=input ('Enter 1st number:\n')
f=input ('Enter 2nd number:\n')
if e==f:
    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
```

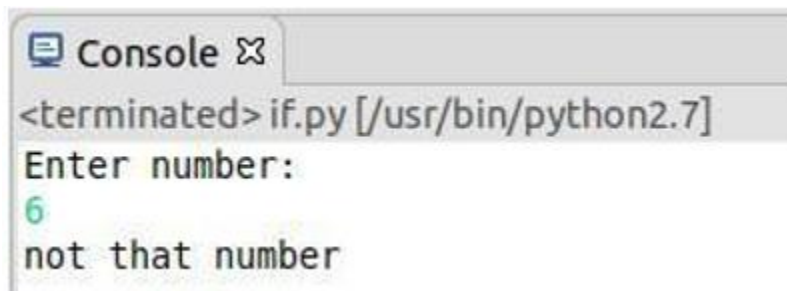
```
Console ✕  
<terminated> Boolean_NOT.py [/usr/bin/python2.7]  
False
```


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"
```

A screenshot of a terminal window titled "Console". The prompt is "<terminated> if.py [/usr/bin/python2.7]". The user has entered "Enter number:" and the program has responded with "6" and "not that number".

```
<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_number=5
```

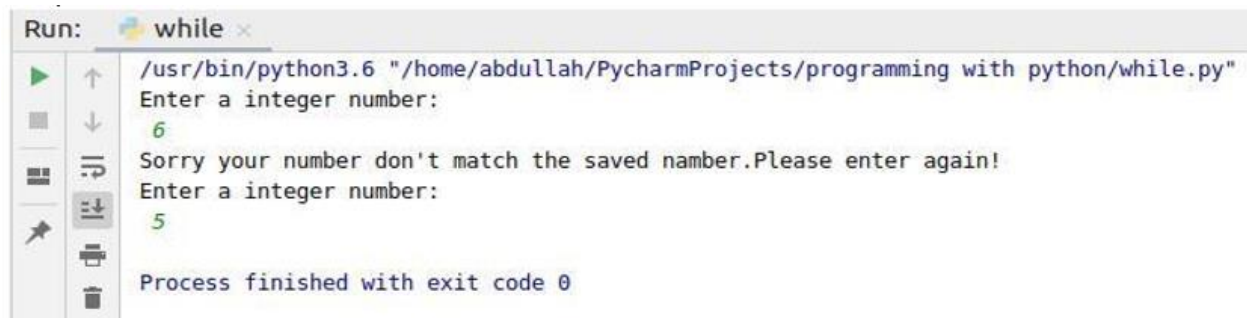
```
number=int(input('Enter a integer number:\n '))
```

```
while number !=saved_number:
```

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

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

Output:



```
Run: while x
/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 number.Please enter again!
Enter a integer number:
5
Process finished with exit code 0
```

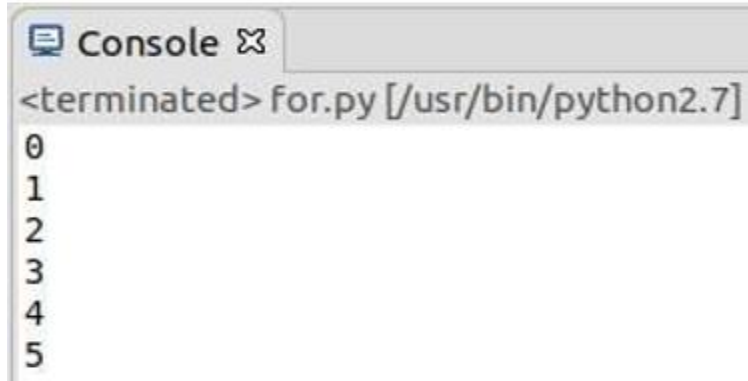
Exercise 4.2.4: The for Statement

Create a program for printing a sequence of numbers.

Ans:

```
for x in range(6):
```

```
    print(x)
```

A screenshot of a console window titled "Console" with a close button. The prompt is "<terminated> for.py [/usr/bin/python2.7]". Below the prompt, the numbers 0 through 5 are listed vertically, each on a new line.

```
<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))`



The screenshot shows a Python IDE window titled "question_5.4_introduction to pytho...". The code being executed is `/usr/bin/python3.6 "/home/abdullah/PycharmProjects/programming with python/question_5.4_introduction to python lab.py"`. The output shows the file path and line number, followed by the code `x=1; print 'sin(%g)=%g' % (x, sin(x))`. A red arrow points to the closing quote of the string, indicating a syntax error. The error message is "SyntaxError: invalid character in identifier". The process finished with exit code 1.

Correct code:

`import math as m`

`x=1`

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



The screenshot shows the same Python IDE window. The code being executed is `/usr/bin/python3.6 "/home/abdullah/PycharmProjects/programming with python/question_5.4_introduction to python lab.py"`. The output shows the file path and line number, followed by the code `sin (1) = 0.841471`. The 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:



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

Discussion:

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of

variable. We learn from this report about python, main feature of python, type of variable and their methodology.