Mawlana Bhashani Science and Technology University

Lab-Report

Report No: 03

Course code: ICT-3208

Course title: Network Planning and Designing Lab

Date of Performance:

Date of Submission:

Submitted by

Name: Md. Rafatul Haque

ID:IT-17037

3rd year 2nd semester

Session: 2016-2017

Dept. of ICT

MBSTU.

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Lab Report Name: Introduction to Python.

Objective: The objective of the lab 1 is to:

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

Theory:

Definition of Python:

The official definition of Python is:

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:

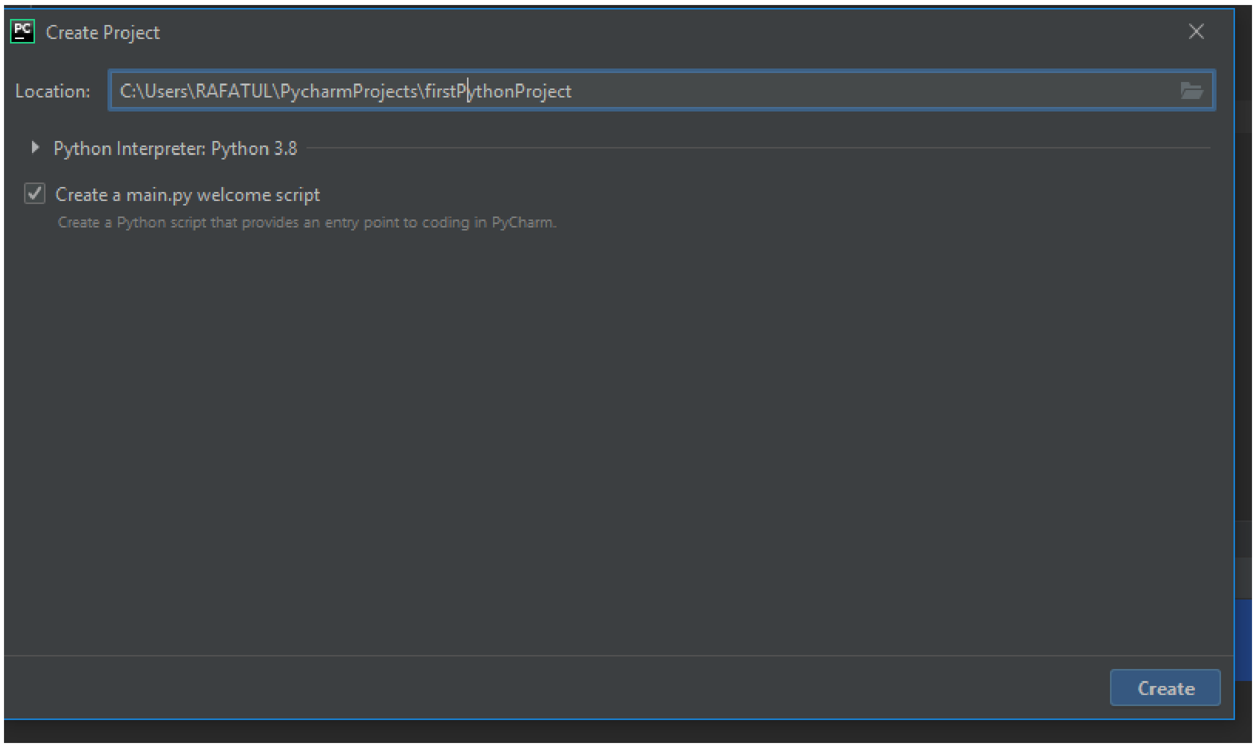
The main features of Python are:

- **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 with. 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, you 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 that you know you can do these things. FLOSS is based on the concept of a community which shares knowledge.
- **High-level Language**: When you write programs in Python, you 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 your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

- **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. You just run the program directly from the source code. Internally, Python converts the source code into an Page | 5 SDN-Labs intermediate form called byte codes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!
- **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.
- **Extensible**: If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your programing C or C++ and then use them from your Python program.
- **Embeddable**: You can embed Python within your C/C++ programs to give 'scripting' capabilities for your program's users.
- **Extensive Libraries**: The Python Standard Library is huge indeed. It can help you 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), Tk, and other system-dependent stuff. Remember, all this is always available wherever Python is installed.

Set up python:

1. Create a project



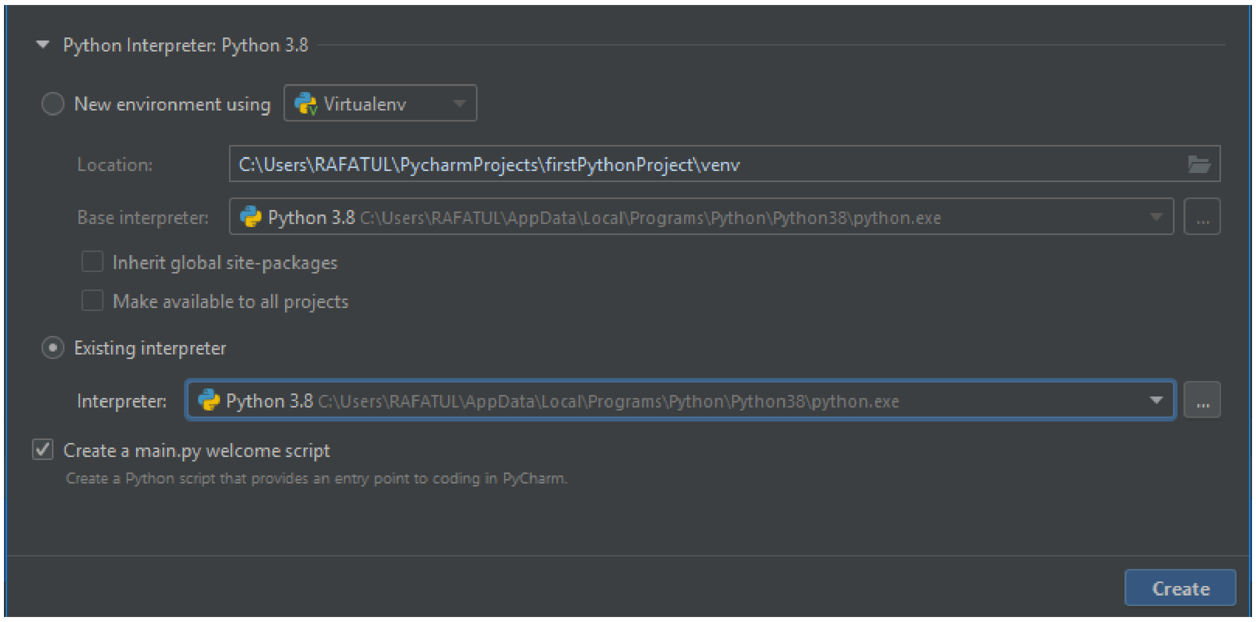
Create Project

Location: C:\Users\RAFATUL\PycharmProjects\firstPythonProject

Python Interpreter: Python 3.8

[✓] Create a main.py welcome script

Create a Python script that provides an entry point to coding in PyCharm.



Python Interpreter: Python 3.8

( ) New environment using Virtualenv

Location: C:\Users\RAFATUL\PycharmProjects\firstPythonProject\venv

Base interpreter: Python 3.8 C:\Users\RAFATUL\AppData\Local\Programs\Python\Python38\python.exe

[ ] Inherit global site-packages

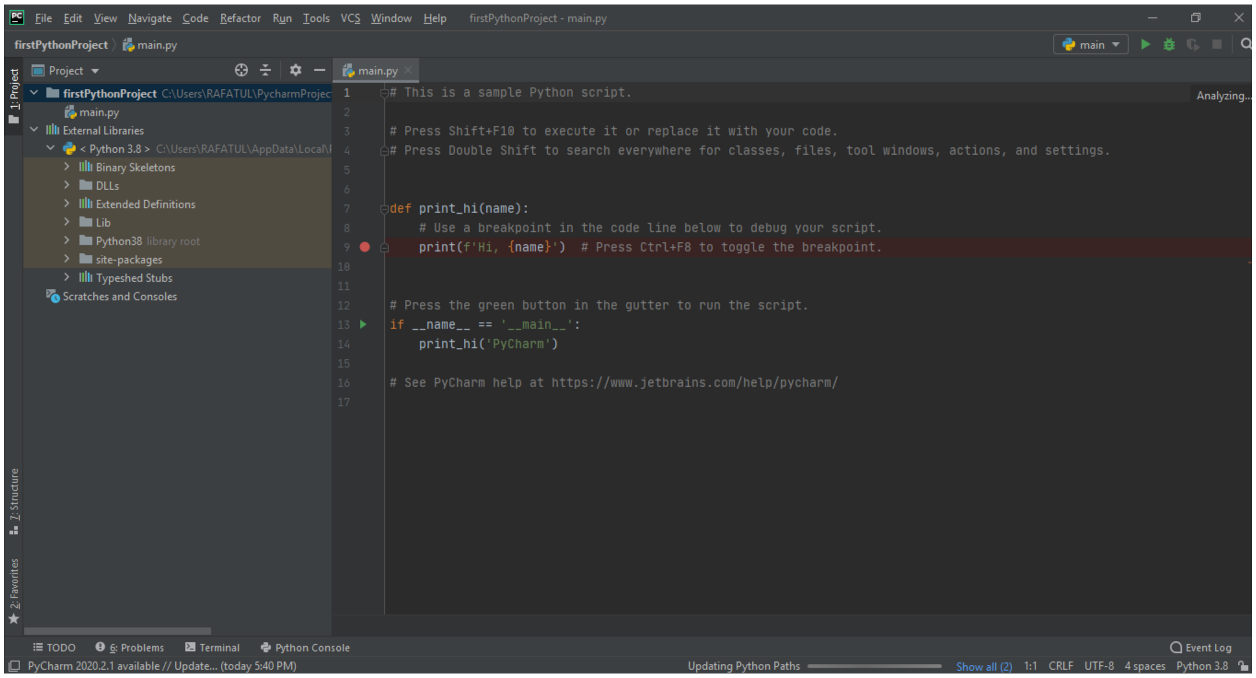
[ ] Make available to all projects

(●) Existing interpreter

Interpreter: Python 3.8 C:\Users\RAFATUL\AppData\Local\Programs\Python\Python38\python.exe

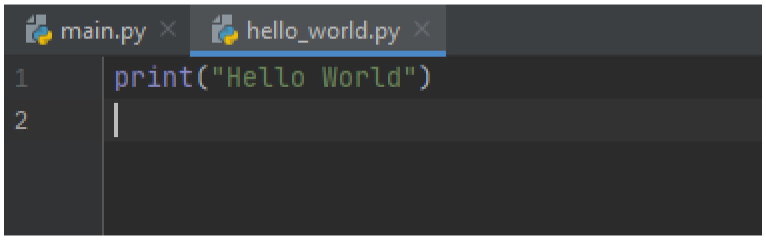
[✓] Create a main.py welcome script

Create a Python script that provides an entry point to coding in PyCharm.

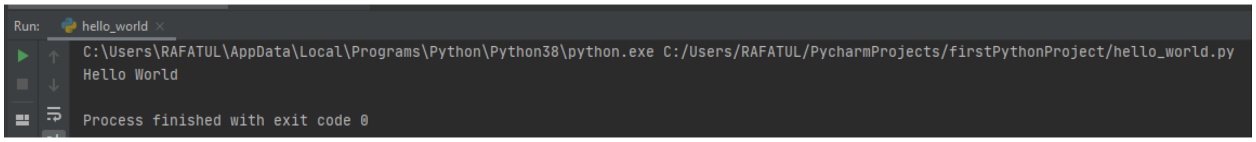

File Edit View Navigate Code Refactor Run Tools VCS Window Help firstPythonProject - main.py
firstPythonProject main.py
Project
firstPythonProject C:\Users\RAFATUL\PycharmProjec
main.py
External Libraries
< Python 3.8 > C:\Users\RAFATUL\AppData\Local\
Binary Skeletons
DLLs
Extended Definitions
Lib
Python38 library root
site-packages
Typeshed Stubs
Scratches and Consoles
TODO
6: Problems
Terminal
Python Console
Event Log
PyCharm 2020.2.1 available // Update... (today 5:40 PM)
Updating Python Paths
Show all (2) 1:1 CRLF UTF-8 4 spaces Python 3.8

```python
# This is a sample Python script.

# Press Shift+F10 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, files, tool windows, actions, and settings.


def print_hi(name):
    # Use a breakpoint in the code line below to debug your script.
    print(f'Hi, {name}')  # Press Ctrl+F8 to toggle the breakpoint.


# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    print_hi('PyCharm')

# See PyCharm help at https://www.jetbrains.com/help/pycharm/
```

Exercise 4.1.2: Write a Hello World program almost all books about programming languages start with a very simple program that prints the text Hello, World! to the screen. Make such a program in Python. (save as hello_world.py).

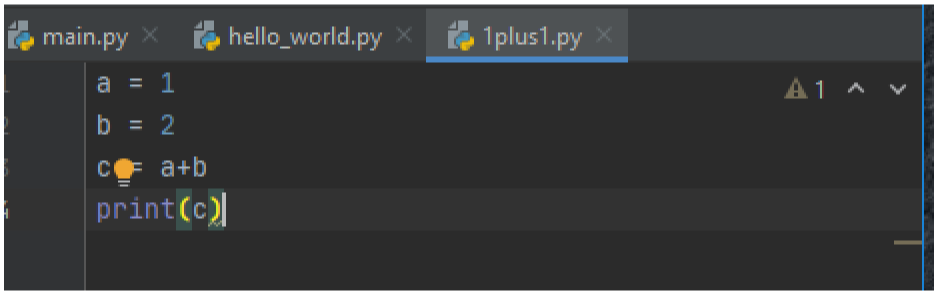


```python
print("Hello World")
```

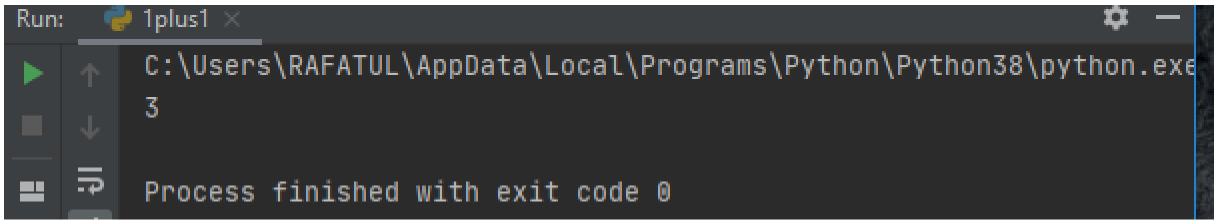


```
C:\Users\RAFATUL\AppData\Local\Programs\Python\Python38\python.exe C:/Users/RAFATUL/PycharmProjects/firstPythonProject/hello_world.py
Hello World

Process finished with exit code 0
```

Exercise 4.1.3: Compute 1+1 The first exercise concerns some very basic mathematics and programming: assign the result of 1+1 to a variable and print the value of that variable (save as 1plus1.py).

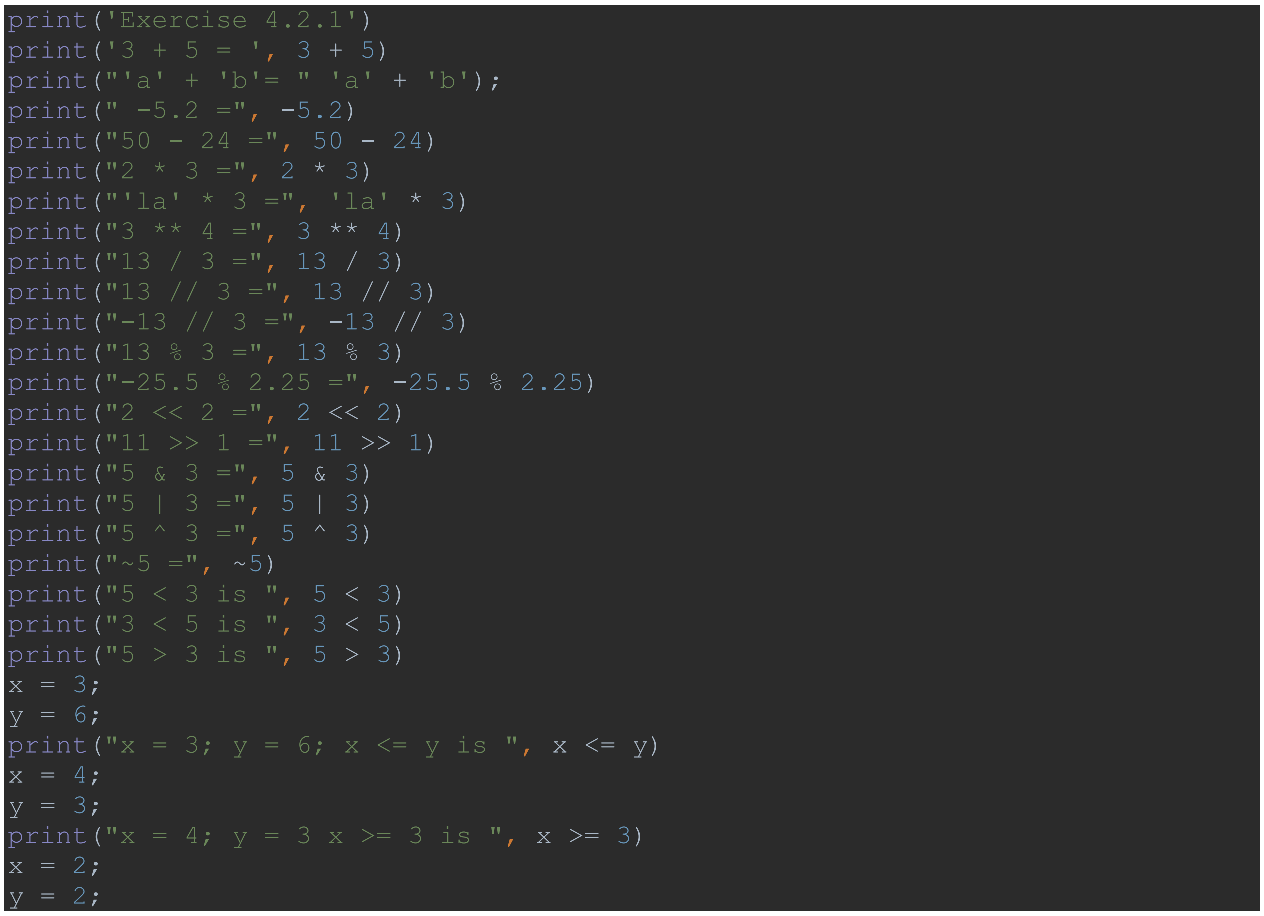


```
a = 1
b = 2
c = a+b
print(c)
```

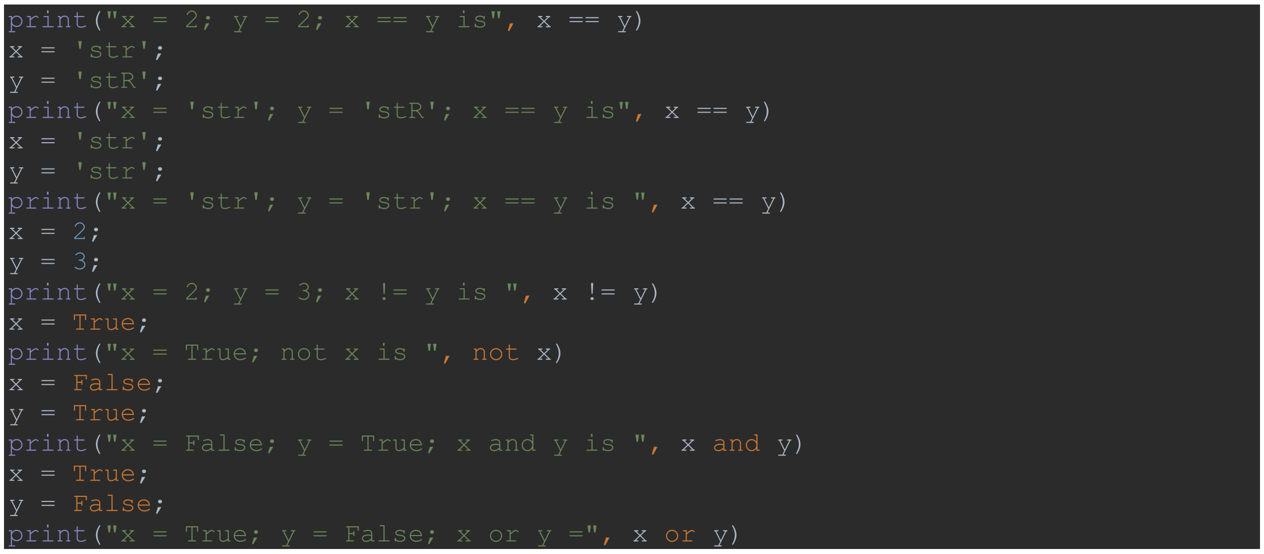


```
C:\Users\RAFATUL\AppData\Local\Programs\Python\Python38\python.exe
3

Process finished with exit code 0
```

Exercise 4.1.4: Type in program text Type the following program in your editor and execute it. If your program does not work, check that you have copied the code correctly and debug it (save as formulas_shapes.py).



```
print('Exercise 4.2.1')
print('3 + 5 = ', 3 + 5)
print("'a' + 'b'= " 'a' + 'b');
print(" -5.2 =", -5.2)
print("50 - 24 =", 50 - 24)
print("2 * 3 =", 2 * 3)
print("'la' * 3 =", 'la' * 3)
print("3 ** 4 =", 3 ** 4)
print("13 / 3 =", 13 / 3)
print("13 // 3 =", 13 // 3)
print("-13 // 3 =", -13 // 3)
print("13 % 3 =", 13 % 3)
print("-25.5 % 2.25 =", -25.5 % 2.25)
print("2 << 2 =", 2 << 2)
print("11 >> 1 =", 11 >> 1)
print("5 & 3 =", 5 & 3)
print("5 | 3 =", 5 | 3)
print("5 ^ 3 =", 5 ^ 3)
print("~5 =", ~5)
print("5 < 3 is ", 5 < 3)
print("3 < 5 is ", 3 < 5)
print("5 > 3 is ", 5 > 3)
x = 3;
y = 6;
print("x = 3; y = 6; x <= y is ", x <= y)
x = 4;
y = 3;
print("x = 4; y = 3 x >= 3 is ", x >= 3)
x = 2;
y = 2;
```



```
print("x = 2; y = 2; x == y is", x == y)
x = 'str';
y = 'stR';
print("x = 'str'; y = 'stR'; x == y is", x == y)
x = 'str';
y = 'str';
print("x = 'str'; y = 'str'; x == y is ", x == y)
x = 2;
y = 3;
print("x = 2; y = 3; x != y is ", x != y)
x = True;
print("x = True; not x is ", not x)
x = False;
y = True;
print("x = False; y = True; x and y is ", x and y)
x = True;
y = False;
print("x = True; y = False; x or y =", x or y)
```

Result:

Exercise 4.2.1

3 + 5 = 8

'a' + 'b'= ab

 -5.2 = -5.2

50 - 24 = 26

2 * 3 = 6

'la' * 3 = lalala

3 ** 4 = 81

13 / 3 = 4.333333333333333

13 // 3 = 4

-13 // 3 = -5

13 % 3 = 1

-25.5 % 2.25 = 1.5

2 << 2 = 8

11 >> 1 = 5

5 & 3 = 1

5 | 3 = 7

5 ^ 3 = 6

~5 = -6

5 < 3 is False

3 < 5 is True

5 > 3 is True

x = 3; y = 6; x <= y is True

x = 4; y = 3 x >= 3 is True

x = 2; y = 2; x == y is True

x = 'str'; y = 'stR'; x == y is False

x = 'str'; y = 'str'; x == y is True

x = 2; y = 3; x != y is True

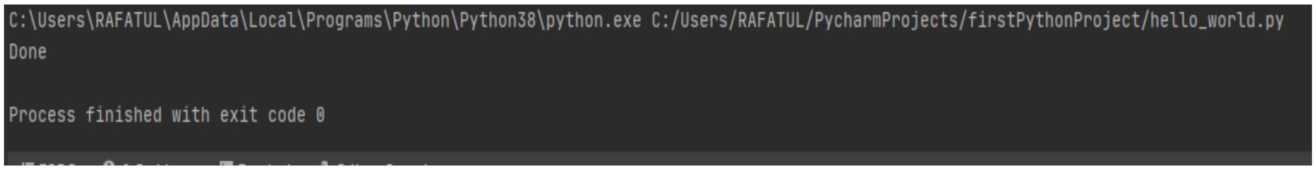
x = True; not x is False

x = False; y = True; x and y is False

x = True; y = False; x or y = True

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 (TIP: use input command). Save the file as if.py

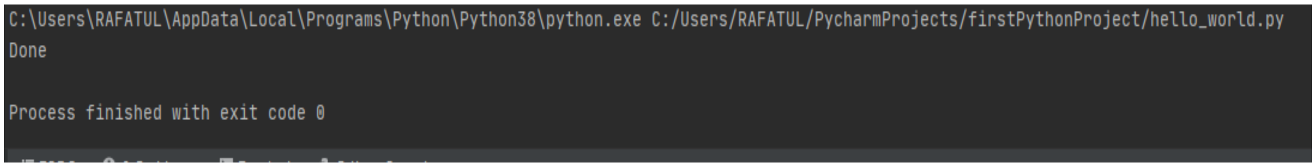
```
from numpy.core import number
number = 23
guess = int(input('Enter an integer : '))
if guess == number:

    print('Congratulations, you guessed it.')
    print('(but you do not win any prizes!)')
elif guess < number:
    print('No, it is a little higher than that')

else:
    print('No, it is a little lower than that')

print('Done')
```



```
C:\Users\RAFATUL\AppData\Local\Programs\Python\Python38\python.exe C:/Users/RAFATUL/PycharmProjects/firstPythonProject/hello_world.py
Done

Process finished with exit code 0
```

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. Save the file as while.py

```
number = 23
running = True
while running:
 guess = int(input('Enter an integer : '))
 if guess == number:
 print('Congratulations, you guessed it.')
 # this causes the while loop to stop
 running = False
 elif number > guess:
 print('No, it is a little higher than that.')
 else:
 print('No, it is a little lower than that.')
else:
 print('The while loop is over.')
 # Do anything else you want to do here
print('Done')
```

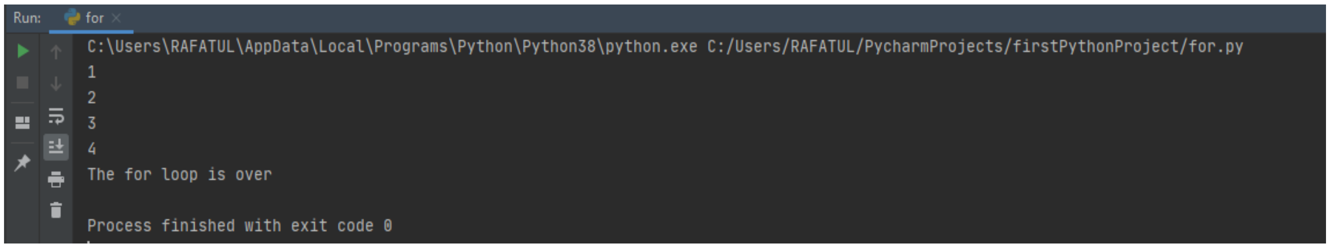


```
C:\Users\RAFATUL\AppData\Local\Programs\Python\Python38\python.exe C:/Users/RAFATUL/PycharmProjects/firstPythonProject/hello_world.py
Done

Process finished with exit code 0
```

Exercise 4.2.4: The for Statement Create a program for printing a sequence of numbers. Save the file as for.py

```
for i in range(1, 5):
 print(i)
```

```
else:
 print('The for loop is over')
```



```
C:\Users\RAFATUL\AppData\Local\Programs\Python\Python38\python.exe C:/Users/RAFATUL/PycharmProjects/firstPythonProject/for.py
1
2
3
4
The for loop is over

Process finished with exit code 0
```

Conclusion: In this lab, firstly set up Pycharm. Then, create a project. Some program implemented.