MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY

Santosh,Tangail-1902

LAB REPORT

Lab No	: 01
Lab name	: Introduction to Python
Course Title	: Network Planning and Designing Lab
Course Code	: ICT-3208

Submitted by,

Tanvir Ahmed
IT-18043
Farhana Afrin Shikha
IT-18038
3rd year 2nd semester
Dept. of ICT

Submitted to,

Nazrul Islam
Assistant professor
Dept. of ICT,
MBSTU

1. Objectives

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.

2. Theory

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:

Simple: Python is a simple and minimalistic language. This pseudocode 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-Platform: 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 PocketPC.

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 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 programming 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.

Other Libraries: Besides, the standard library, there are various other high-quality libraries such as:

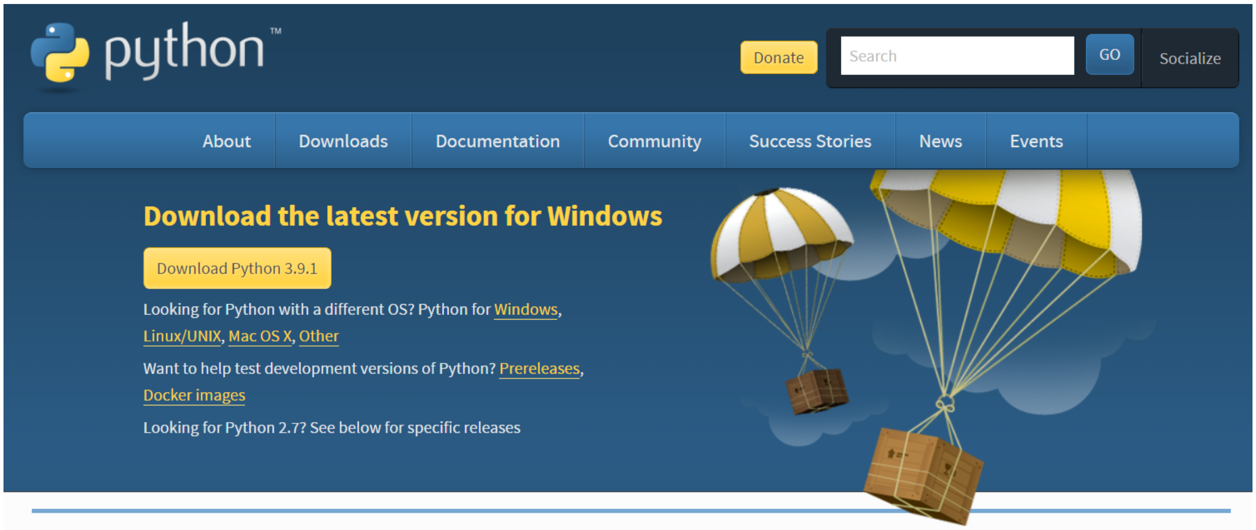
o wxPython [http://www.wxpython.org],
o Twisted [http://www.twistedmatrix.com/products/twisted],
o Python Imaging Library [http://www.pythonware.com/products/pil/index.htm]

3. Methodology

Section 3.1: *Download python3 and Pycharm IDE and install them.*

STEP 1: In order to set up follow the instructions :

a. Go to this website **https://www.python.org/downloads/**



Download the latest version for Windows

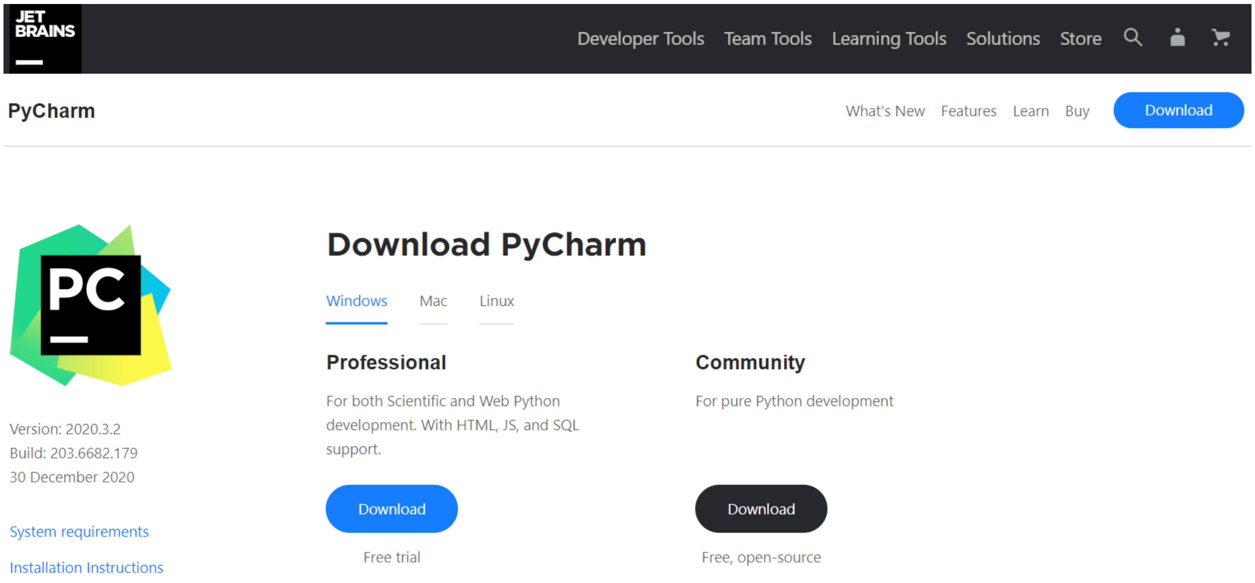
Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Prereleases, Docker images

Looking for Python 2.7? See below for specific releases

b. Download python3

c. Go to this website https://www.jetbrains.com/pycharm/download/



Download PyCharm

Version: 2020.3.2
Build: 203.6682.179
30 December 2020

System requirements

Installation Instructions

Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

Free trial

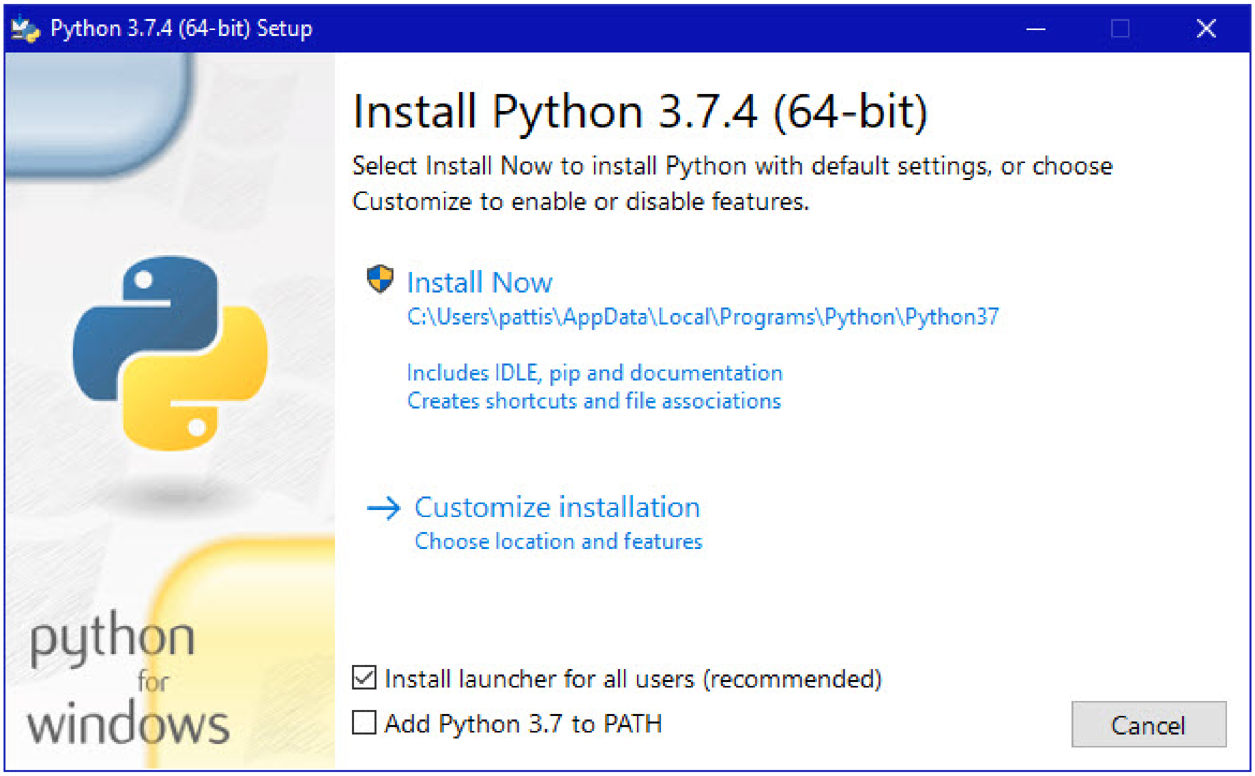
Community

For pure Python development

Free, open-source

d. Download Pycharm IDE

e. Install python3



Python 3.7.4 (64-bit) Setup

Install Python 3.7.4 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\pattis\AppData\Local\Programs\Python\Python37

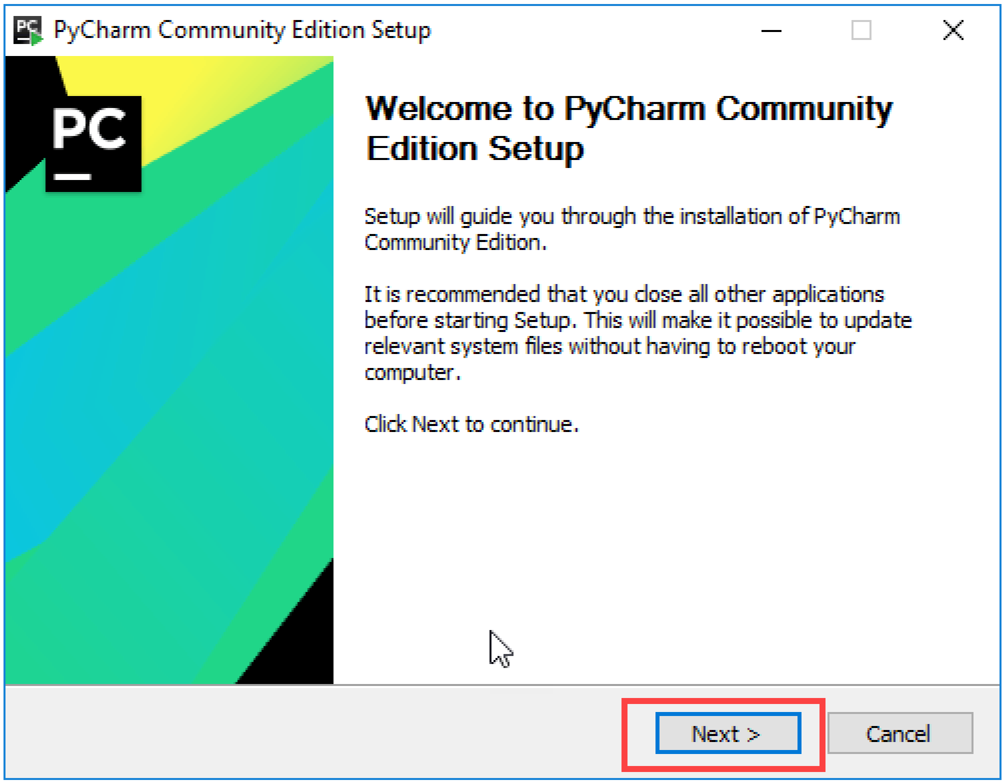
Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)

□ Add Python 3.7 to PATH

f. Install pycharm IDE



PyCharm Community Edition Setup

Welcome to PyCharm Community Edition Setup

Setup will guide you through the installation of PyCharm Community Edition.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

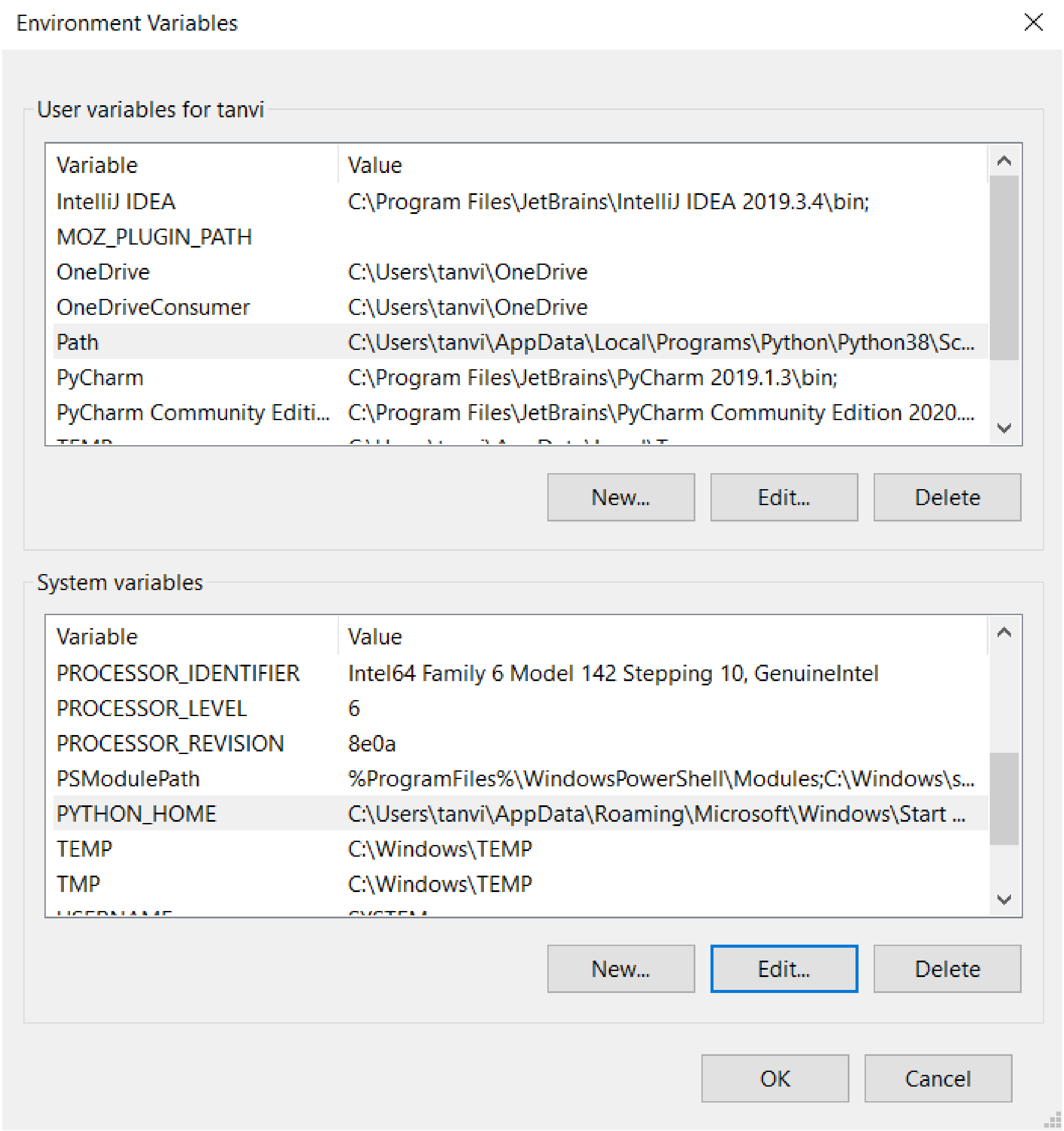
Section 3.2: *Setup of Python Environment*

STEP 1: Open windows environment variables settings and set the path variable of python3.

In order to set up follow the instructions :

a. Go to **Control Panel >System and Security >System >Advanced System Settings**

b. Open **Environment variables settings**

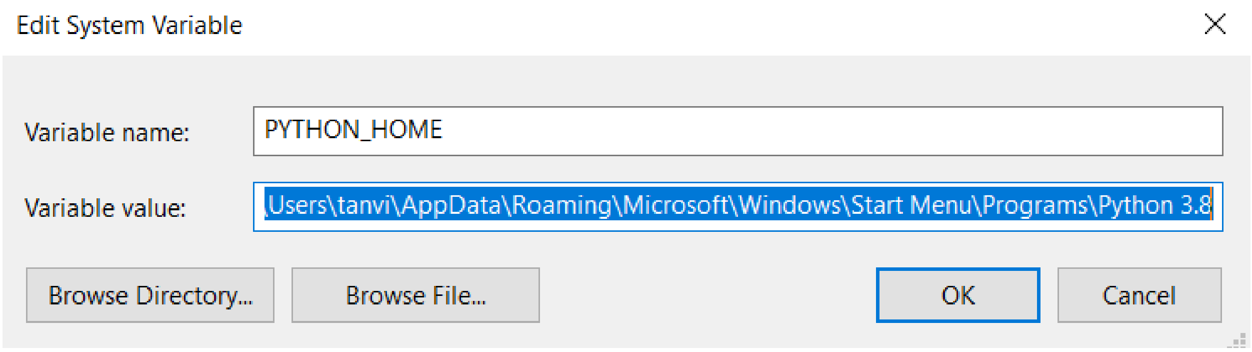


Variable	Value
IntelliJ IDEA	C:\Program Files\JetBrains\IntelliJ IDEA 2019.3.4\bin;
MOZ_PLUGIN_PATH	
OneDrive	C:\Users\tanvi\OneDrive
OneDriveConsumer	C:\Users\tanvi\OneDrive
Path	C:\Users\tanvi\AppData\Local\Programs\Python\Python38\Sc...
PyCharm	C:\Program Files\JetBrains\PyCharm 2019.1.3\bin;
PyCharm Community Editi...	C:\Program Files\JetBrains\PyCharm Community Edition 2020....
TEMP	C:\Users\tanvi\AppData\Local\T

Variable	Value
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 142 Stepping 10, GenuineIntel
PROCESSOR_LEVEL	6
PROCESSOR_REVISION	8e0a
PSModulePath	%ProgramFiles%\WindowsPowerShell\Modules;C:\Windows\s...
PYTHON_HOME	C:\Users\tanvi\AppData\Roaming\Microsoft\Windows\Start ...
TEMP	C:\Windows\TEMP
TMP	C:\Windows\TEMP
USERNAME	SYSTEM

c. In the System Variables Section create a new variable named **PYTHON_HOME**

d. Set the path for PYTHON_HOME



Edit System Variable

Variable name: PYTHON_HOME

Variable value: Users\tanvi\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Python 3.8

e. Save that path.

Section 3.3: *Start pycharm IDE.*

In order to start pycharm IDE follow these instructions :

a. Go to **Start**

b. Search Pycharm IDE

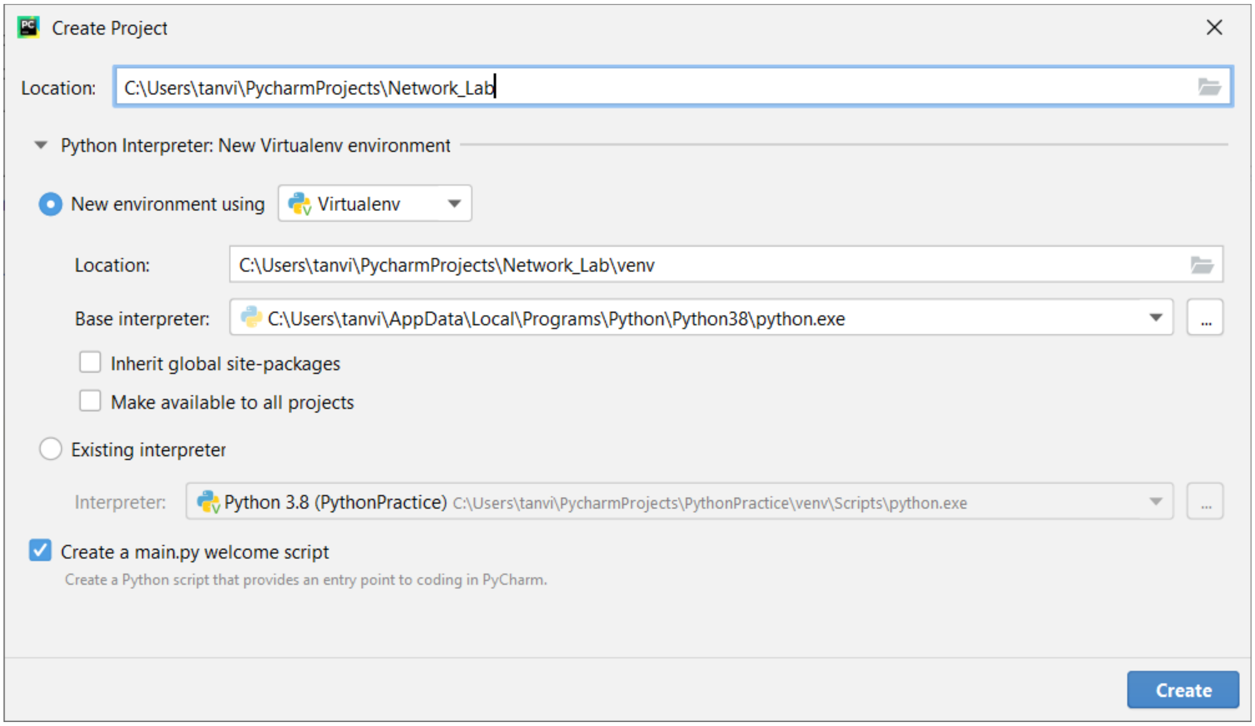
c. Open Pycharm IDE

4. Exercises

Section 4.1: Basics of python and programing

Exercise 4.1.1: Create a python project, click in File > New > PyDev Project. Provide a name for the project (Network_Lab for the fits lab), then select the version of python to be used and select to add the project to working set as shown below:

Answer:



Create Project

Location: C:\Users\tanvi\PycharmProjects\Network_Lab

Python Interpreter: New Virtualenv environment

● New environment using Virtualenv

Location: C:\Users\tanvi\PycharmProjects\Network_Lab\venv

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

□ Inherit global site-packages

□ Make available to all projects

○ Existing interpreter

Interpreter: Python 3.8 (PythonPractice) C:\Users\tanvi\PycharmProjects\PythonPractice\venv\Scripts\python.exe

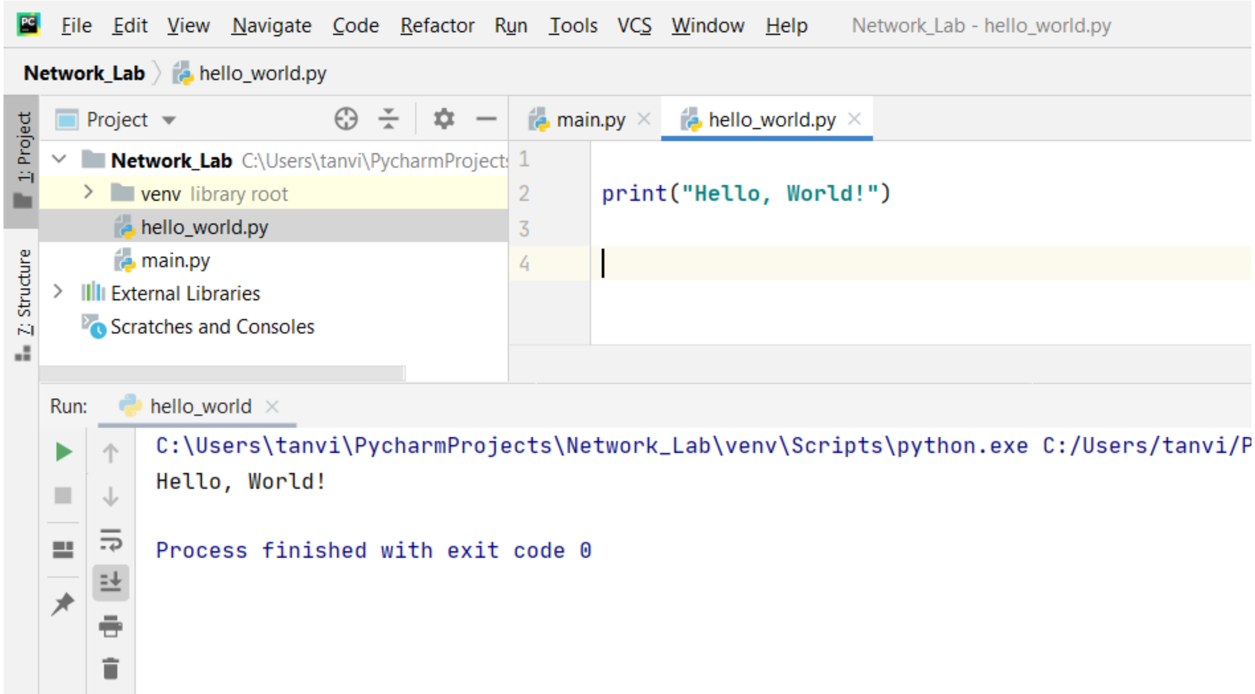
☑ Create a main.py welcome script

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

Answer:

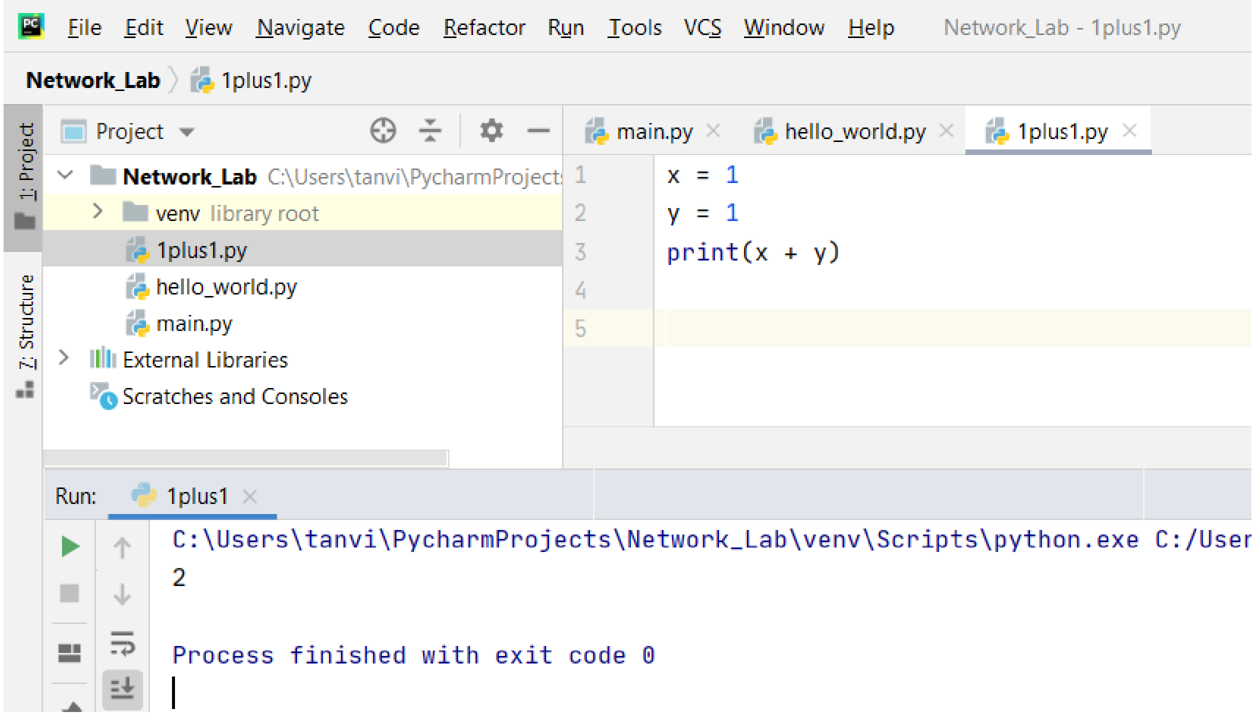


```
print("Hello, World!")
```

```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe C:/Users/tanvi/P
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).

Answer:

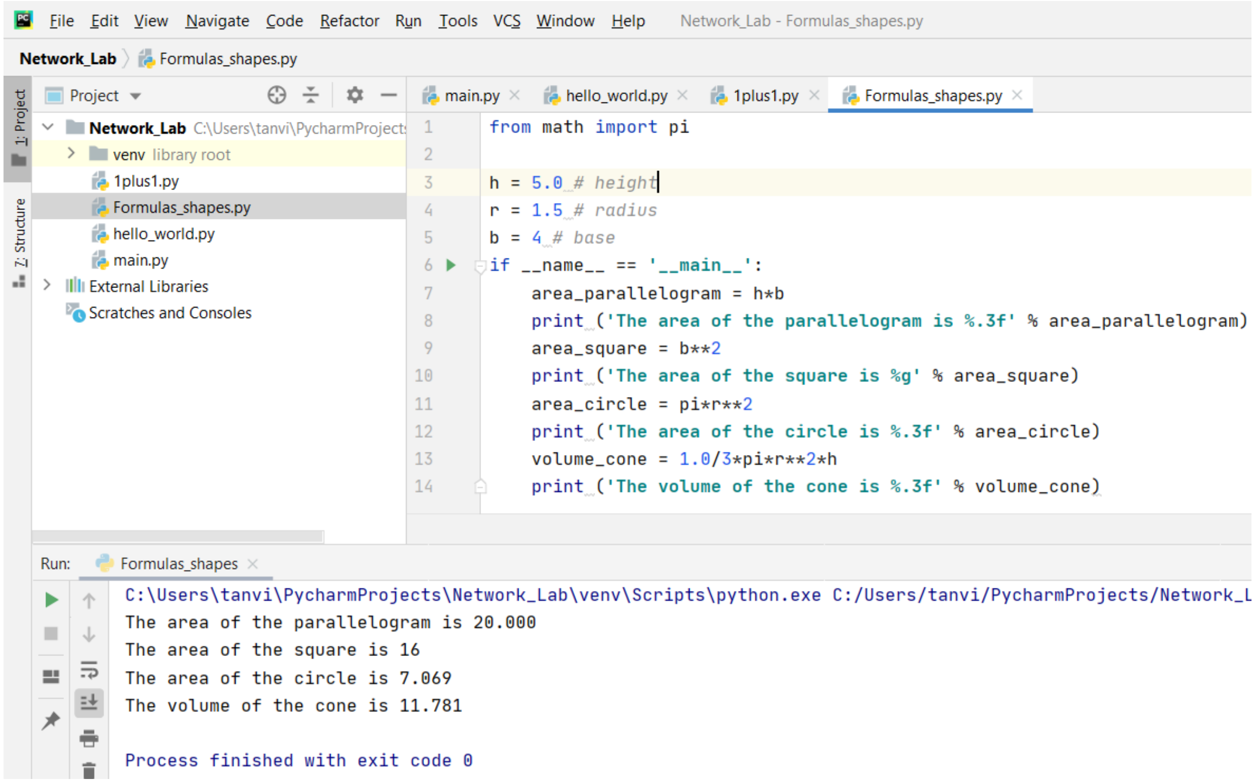


```
x = 1
y = 1
print(x + y)
```

```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe C:/User
2

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).

Answer:



```
from math import pi

h = 5.0 # height
r = 1.5 # radius
b = 4 # base
if __name__ == '__main__':
    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 = pi*r**2
    print ('The area of the circle is %.3f' % area_circle)
    volume_cone = 1.0/3*pi*r**2*h
    print ('The volume of the cone is %.3f' % volume_cone)
```

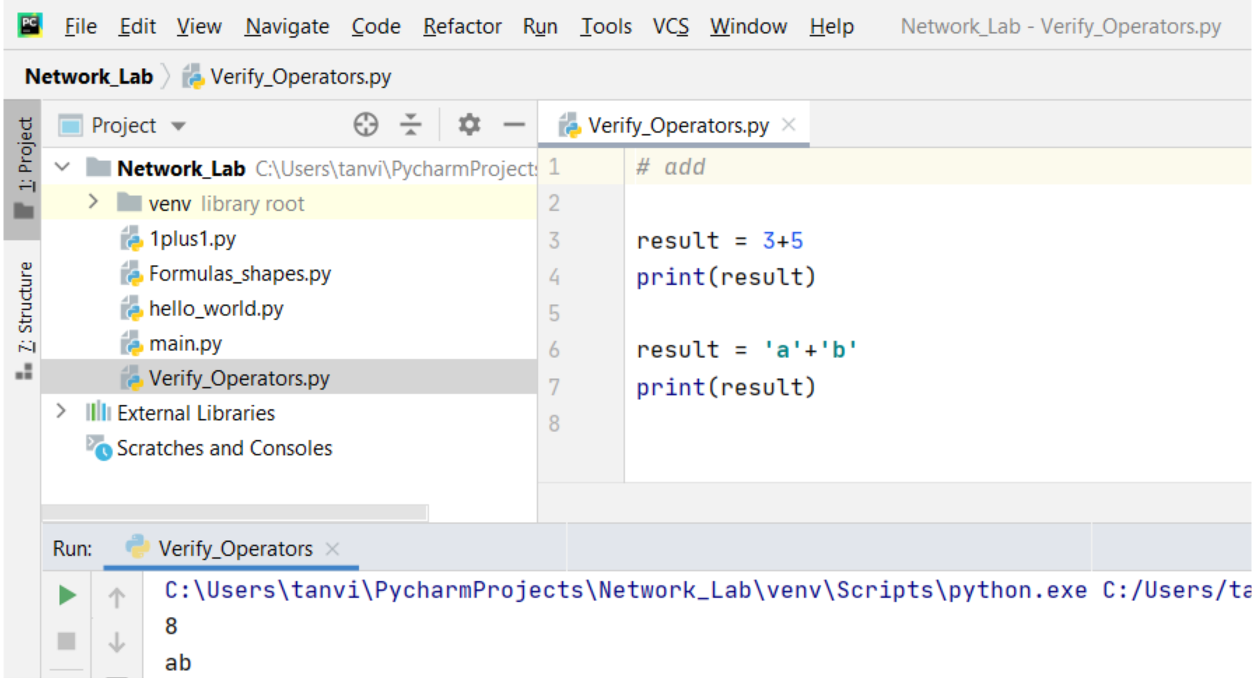
```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe C:/Users/tanvi/PycharmProjects/Network_L
The area of the parallelogram is 20.000
The area of the square is 16
The area of the circle is 7.069
The volume of the cone is 11.781

Process finished with exit code 0
```

Section 4.2: Create and run basic examples.

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

Answer:

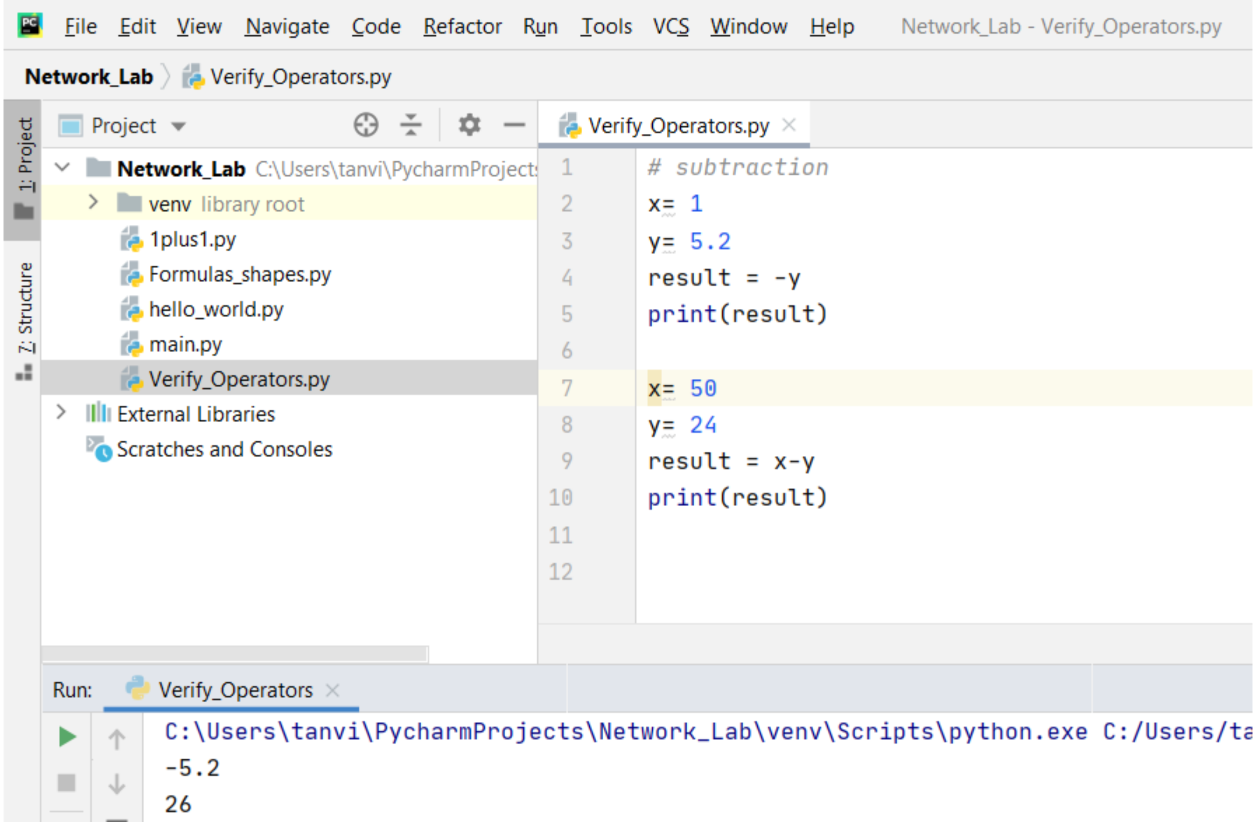
#add



```python
# add

result = 3+5
print(result)

result = 'a'+'b'
print(result)
```

```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe C:/Users/ta
8
ab
```

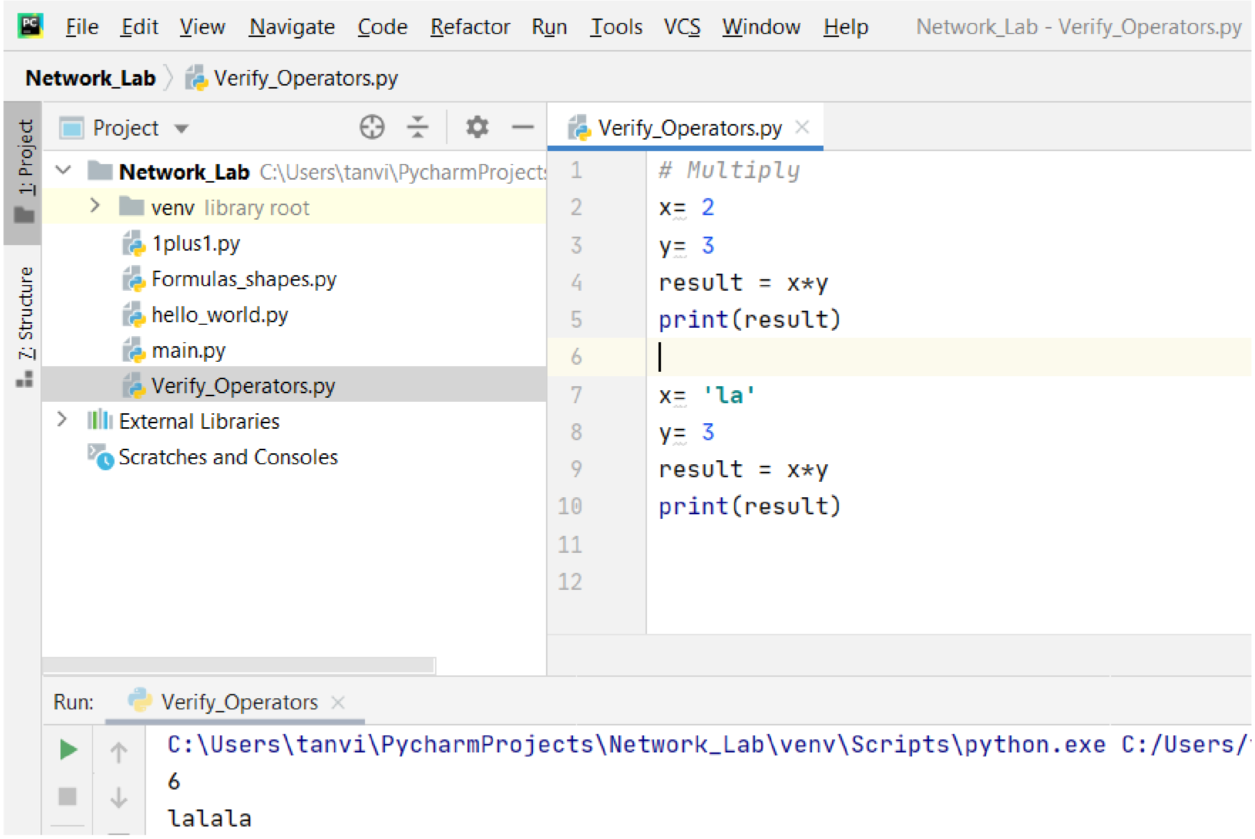
subtraction



```python
# subtraction
x= 1
y= 5.2
result = -y
print(result)

x= 50
y= 24
result = x-y
print(result)


```

```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe C:/Users/ta
-5.2
26
```

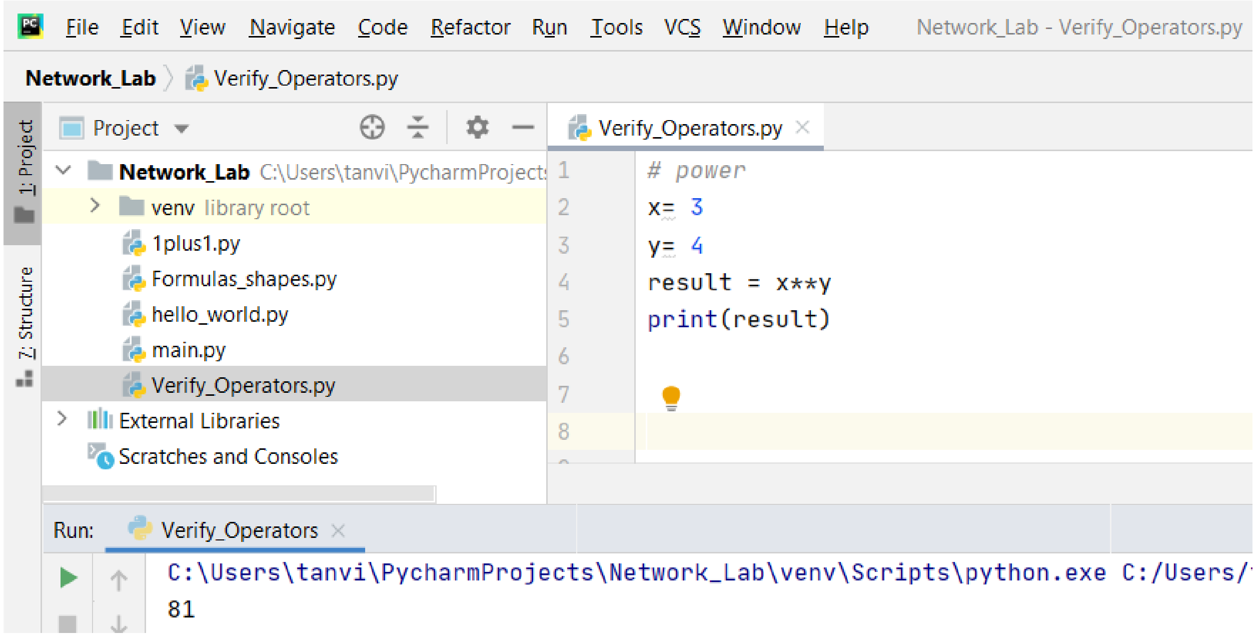
#Multiply



```
# Multiply
x= 2
y= 3
result = x*y
print(result)

x= 'la'
y= 3
result = x*y
print(result)
```

```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe C:/Users/
6
lalala
```

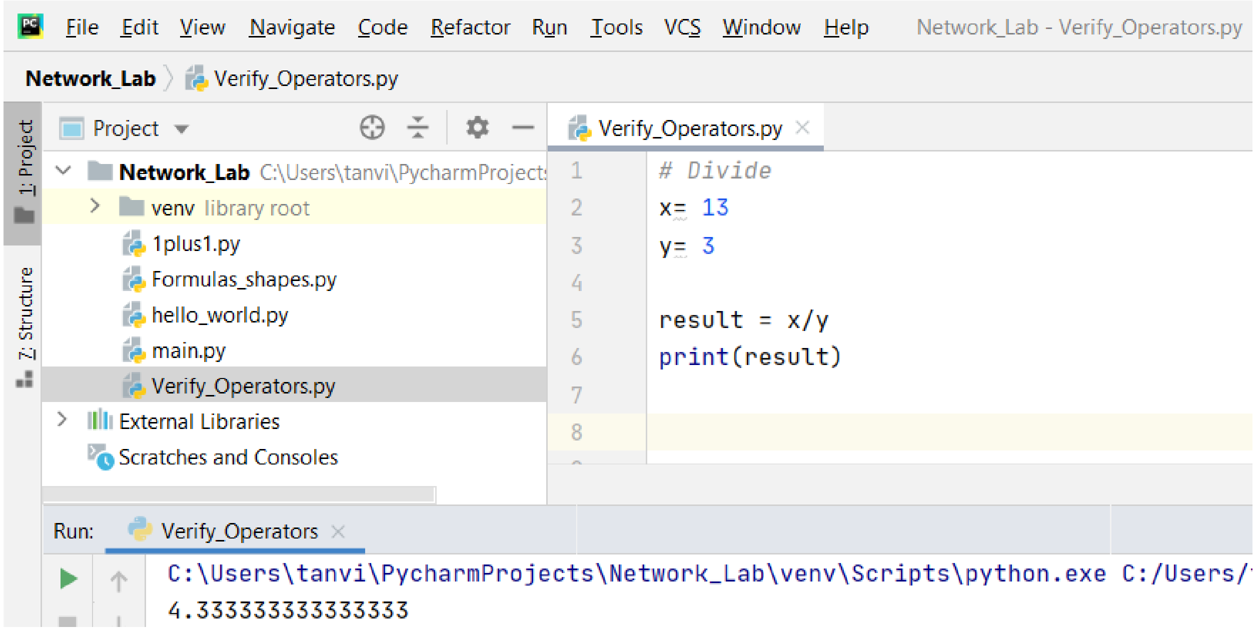
#power



```
# power
x= 3
y= 4
result = x**y
print(result)
```

```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe C:/Users/
81
```

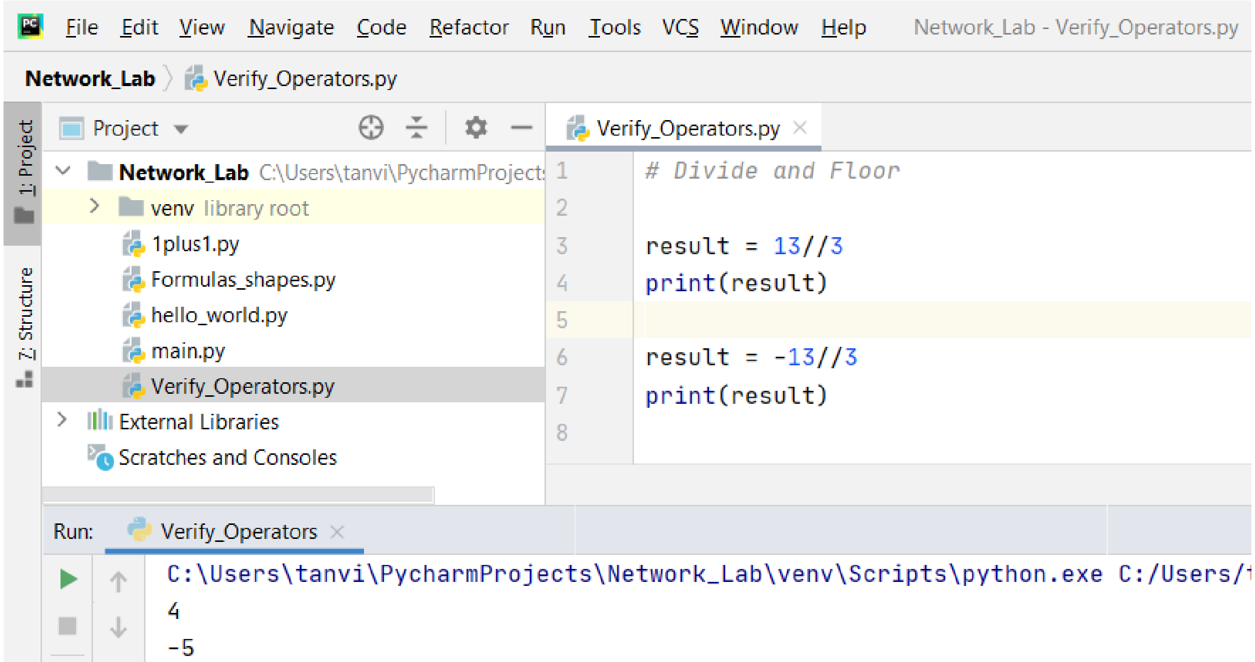
Divide



```
# Divide
x= 13
y= 3

result = x/y
print(result)
```

```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe C:/Users/
4.333333333333333
```

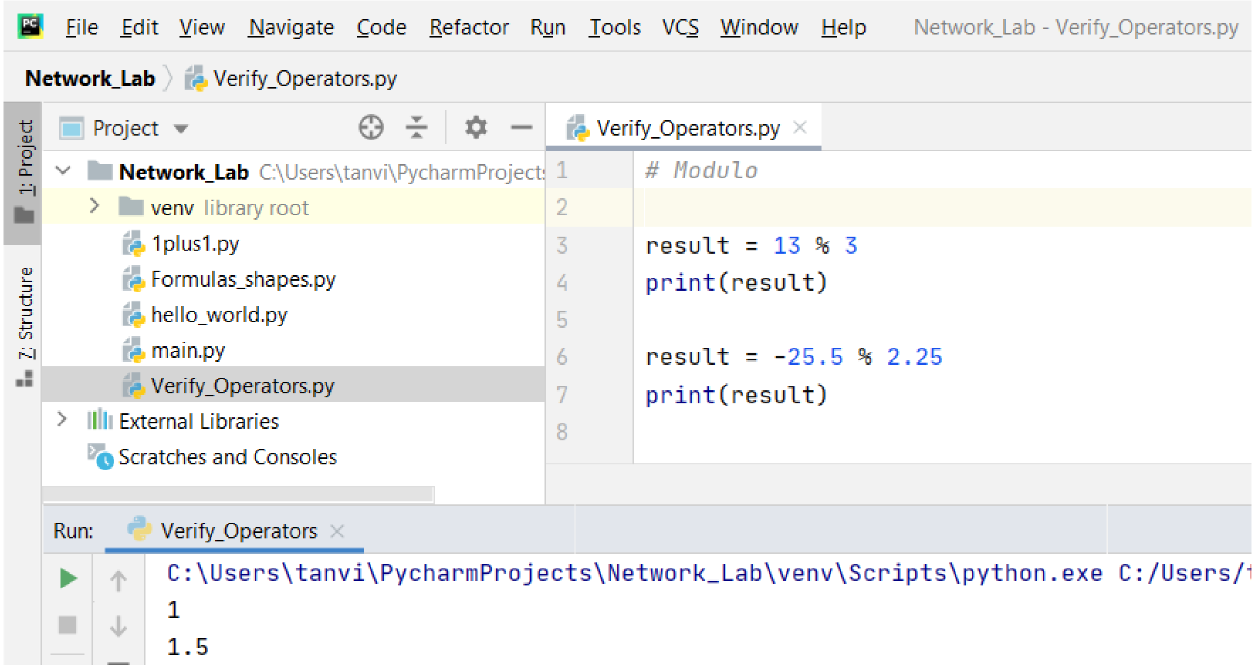
# Divide and floor



```
# Divide and Floor

result = 13//3
print(result)

result = -13//3
print(result)
```

```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe C:/Users/
4
-5
```

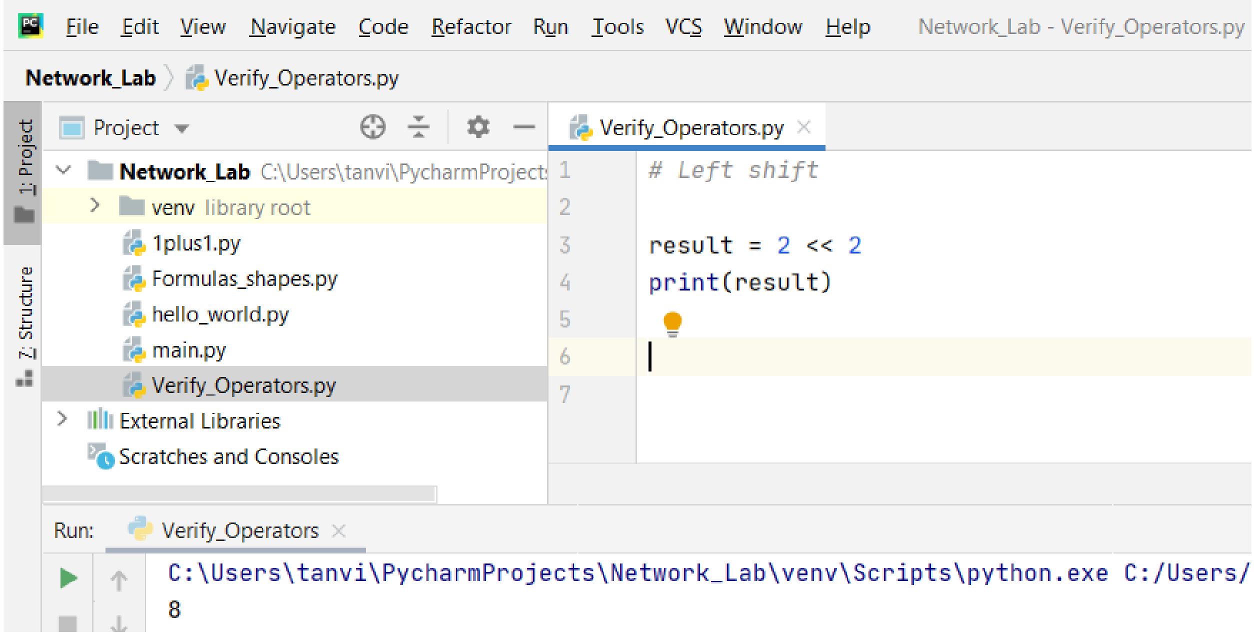
Modulo



```
# Modulo

result = 13 % 3
print(result)

result = -25.5 % 2.25
print(result)
```

```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe C:/Users/t
1
1.5
```

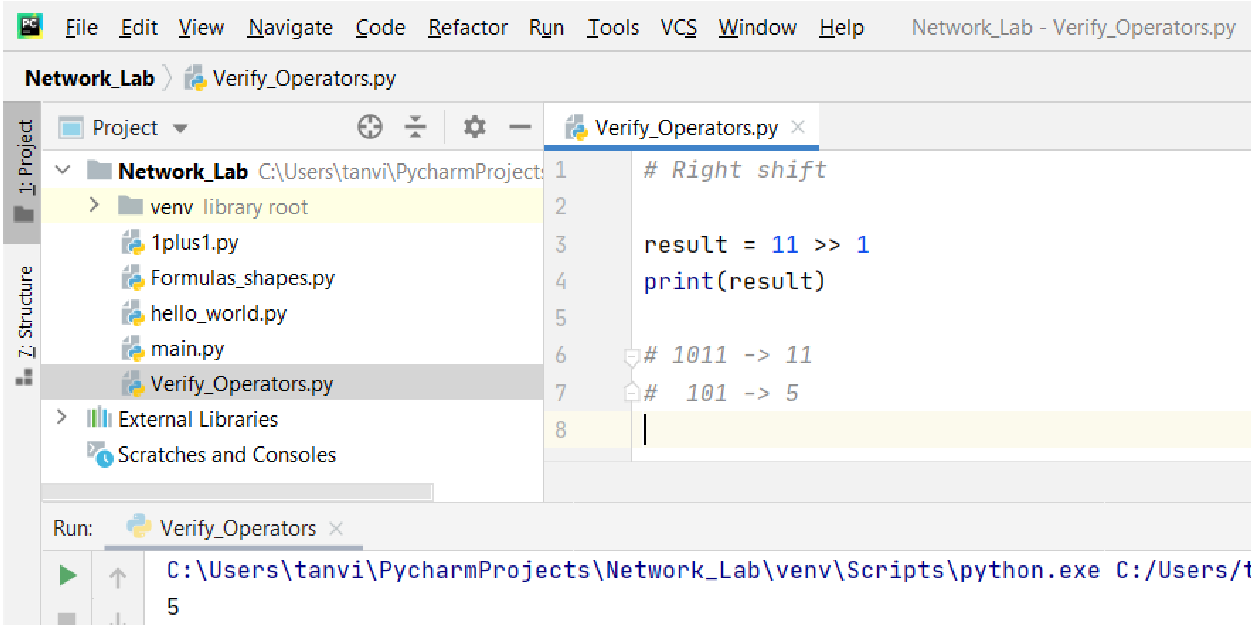
Left shift



```
# Left shift

result = 2 << 2
print(result)
```

```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe C:/Users/
8
```

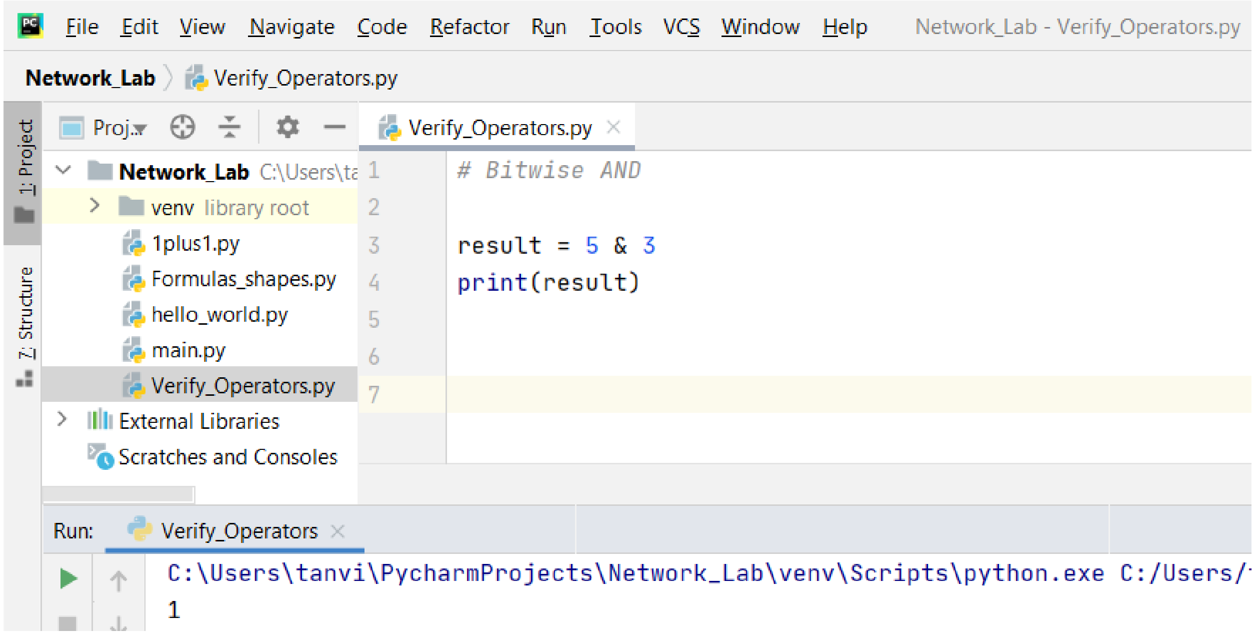
Right shift



```
# Right shift

result = 11 >> 1
print(result)

# 1011 -> 11
#  101 -> 5
```

```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe C:/Users/t
5
```

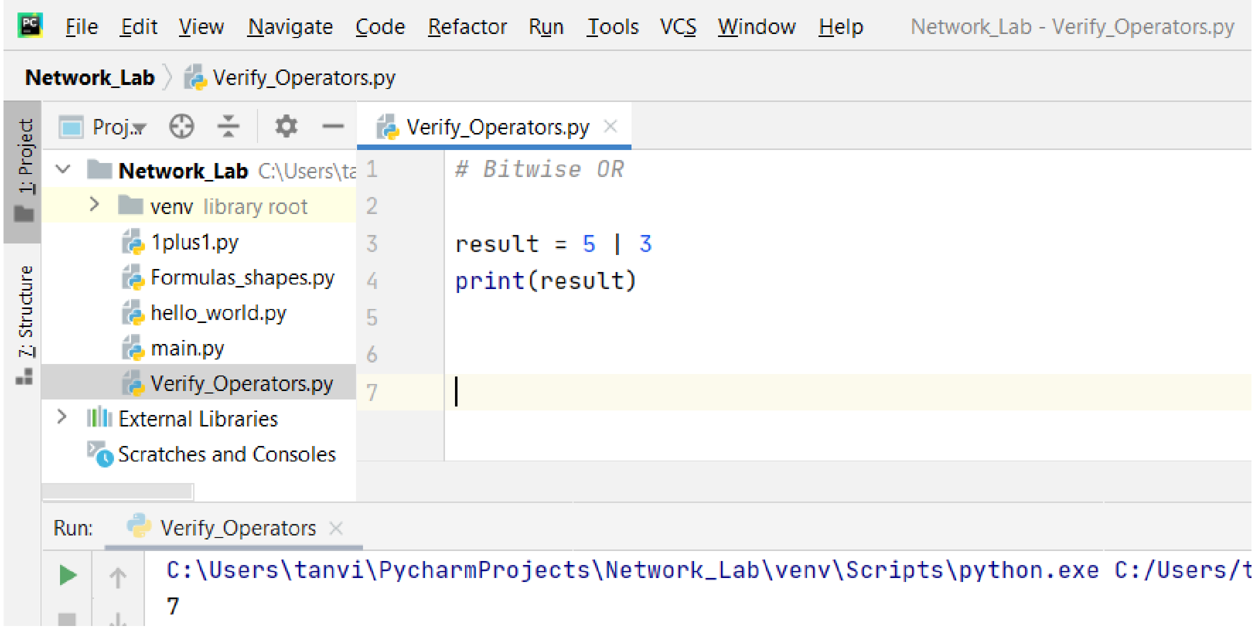
Bitwise AND



```
# Bitwise AND

result = 5 & 3
print(result)
```

```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe C:/Users/
1
```

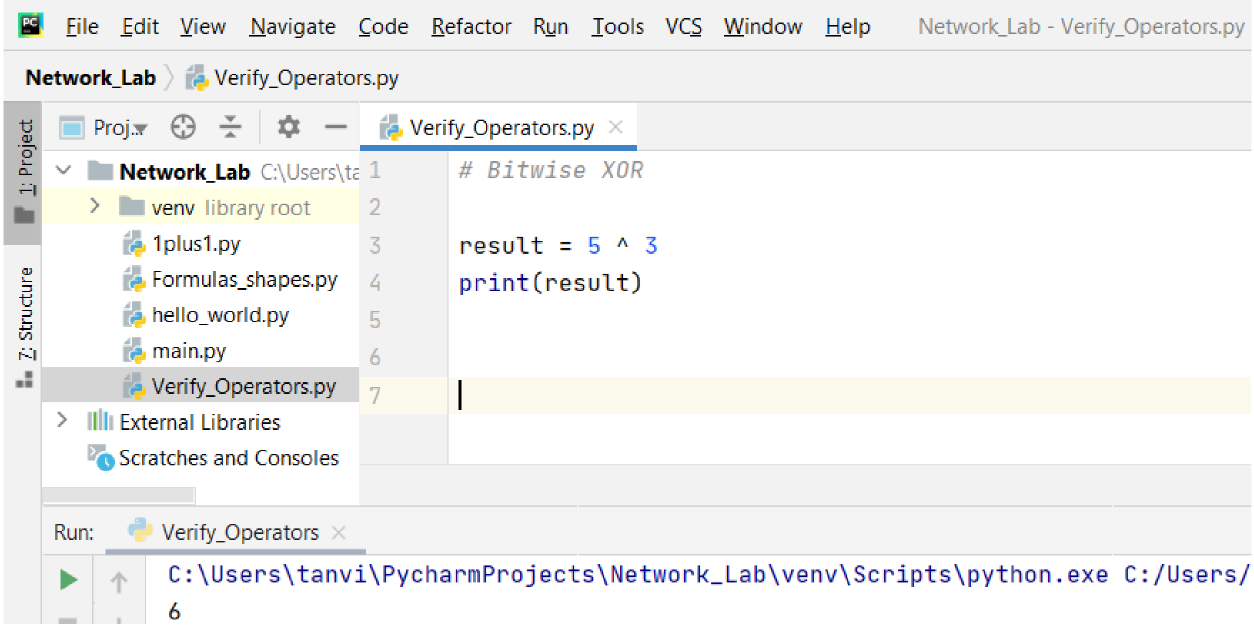
Bitwise OR



```
# Bitwise OR

result = 5 | 3
print(result)
```

```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe C:/Users/t
7
```

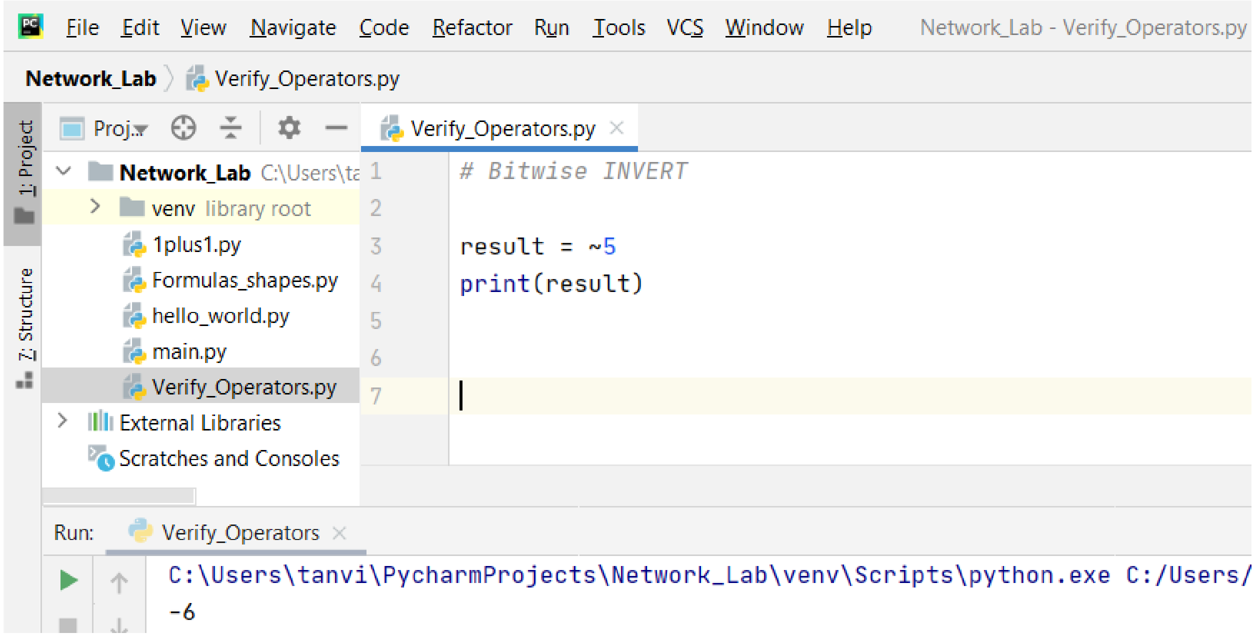
Bitwise XOR



```
# Bitwise XOR

result = 5 ^ 3
print(result)
```

```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe C:/Users/
6
```

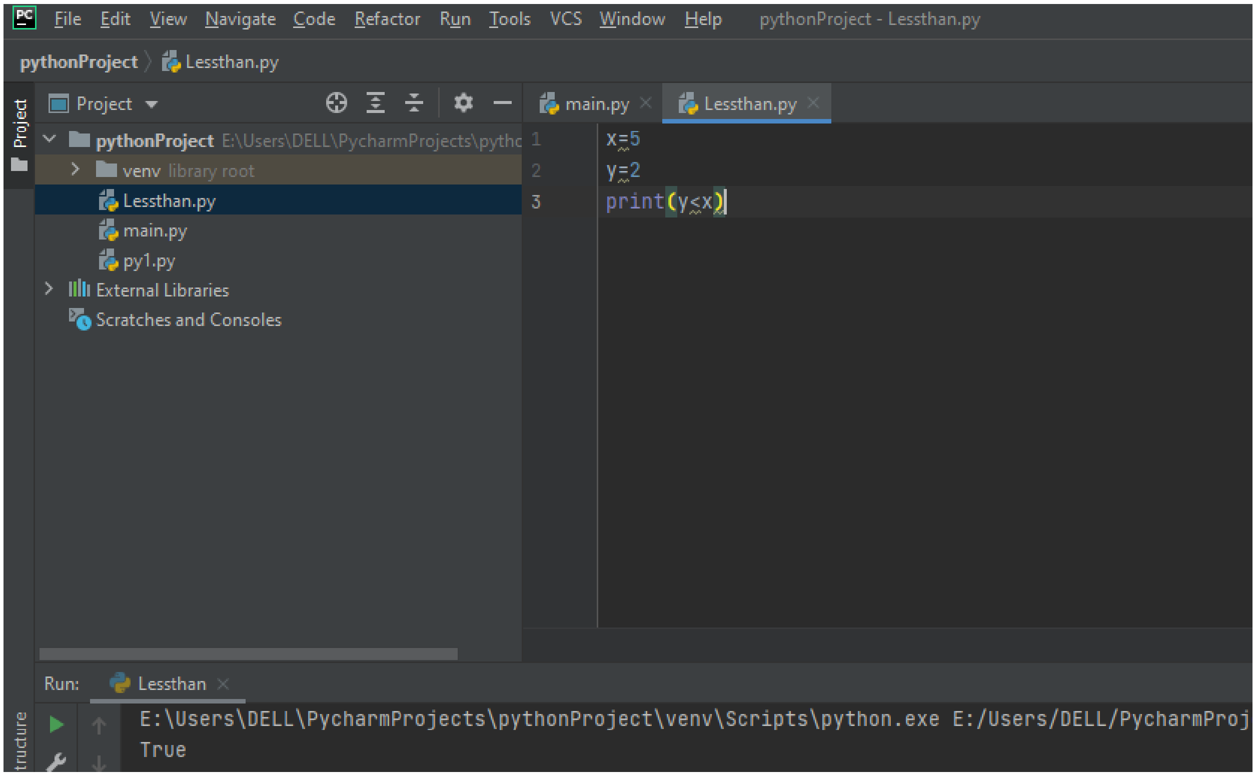
# Bitwise Invert



```
# Bitwise INVERT

result = ~5
print(result)
```

```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe C:/Users/
-6
```

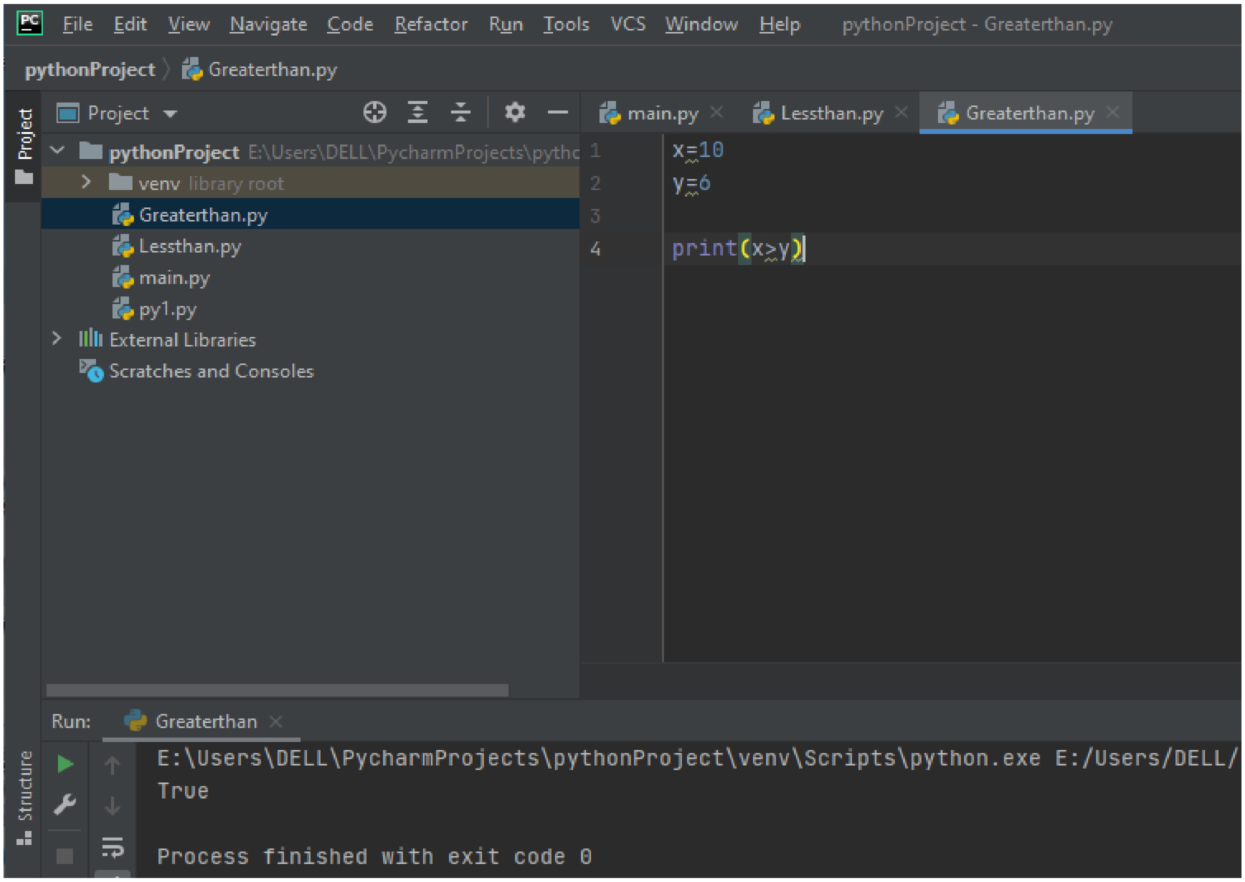
# Less than


File Edit View Navigate Code Refactor Run Tools VCS Window Help pythonProject - Lessthan.py
pythonProject 〉 Lessthan.py
Project
main.py
Lessthan.py
pythonProject E:\Users\DELL\PycharmProjects\pytho
venv library root
Lessthan.py
main.py
py1.py
External Libraries
Scratches and Consoles
Run: Lessthan
tructure

```
x=5
y=2
print(y<x)
```

```
E:\Users\DELL\PycharmProjects\pythonProject\venv\Scripts\python.exe E:/Users/DELL/PycharmProj
True
```

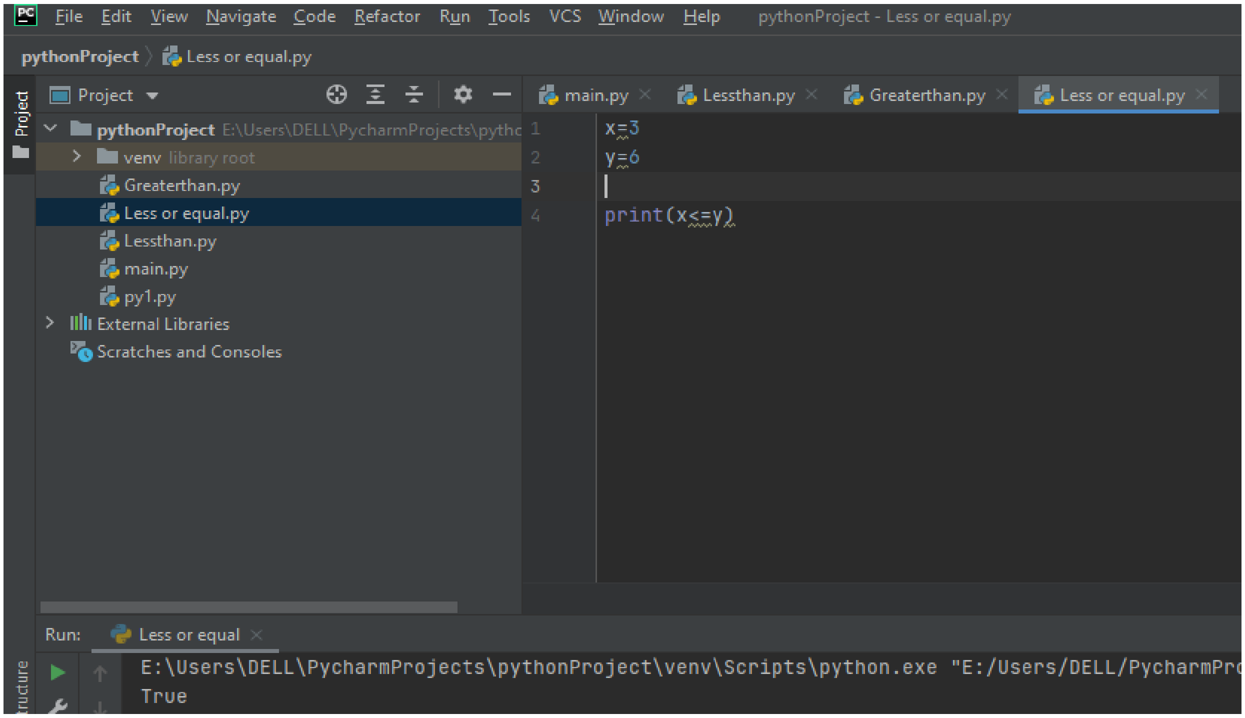
# Greater than


File Edit View Navigate Code Refactor Run Tools VCS Window Help
pythonProject - Greaterthan.py
pythonProject 〉 Greaterthan.py
Project
main.py
Lessthan.py
Greaterthan.py
pythonProject E:\Users\DELL\PycharmProjects\pytho
venv library root
Greaterthan.py
Lessthan.py
main.py
py1.py
External Libraries
Scratches and Consoles
Run:
Greaterthan
Structure

```
x=10
y=6

print(x>y)
```

```
E:\Users\DELL\PycharmProjects\pythonProject\venv\Scripts\python.exe E:/Users/DELL/
True

Process finished with exit code 0
```

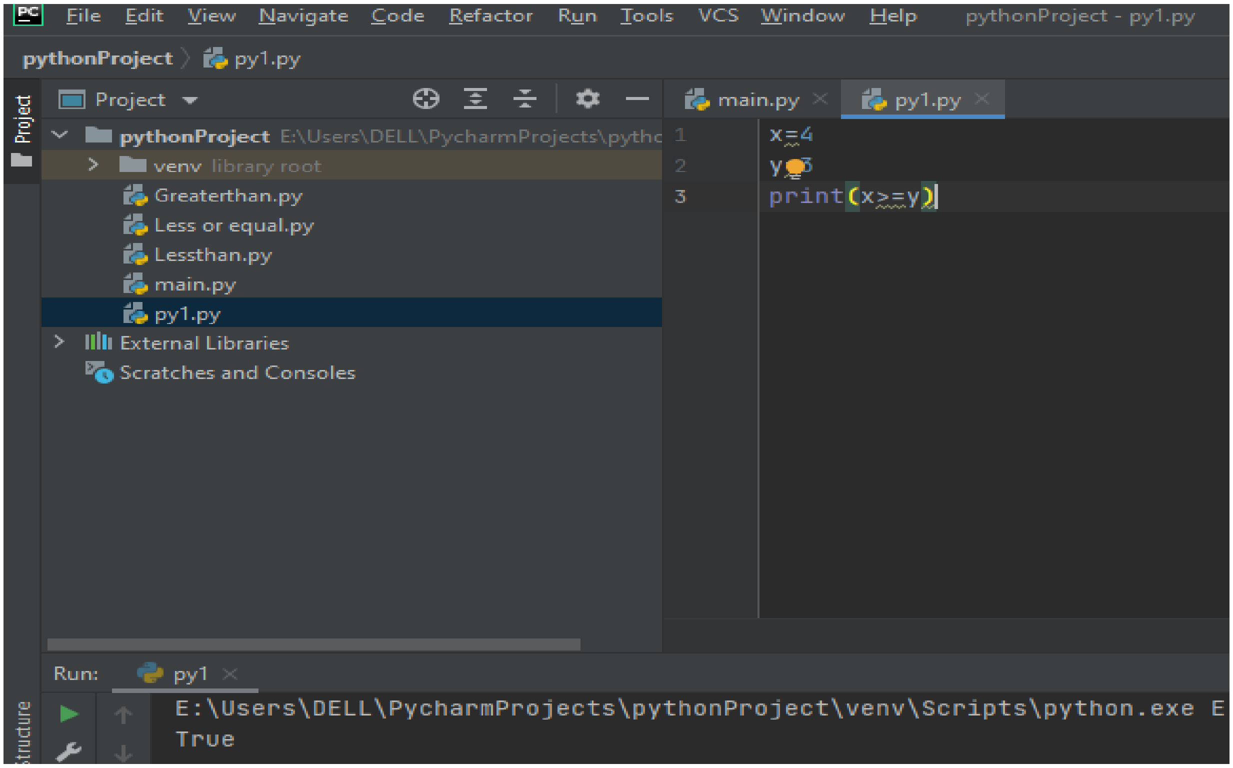
# Less than or equal to



```
x=3
y=6

print(x<=y)
```

```
E:\Users\DELL\PycharmProjects\pythonProject\venv\Scripts\python.exe "E:/Users/DELL/PycharmPro
True
```

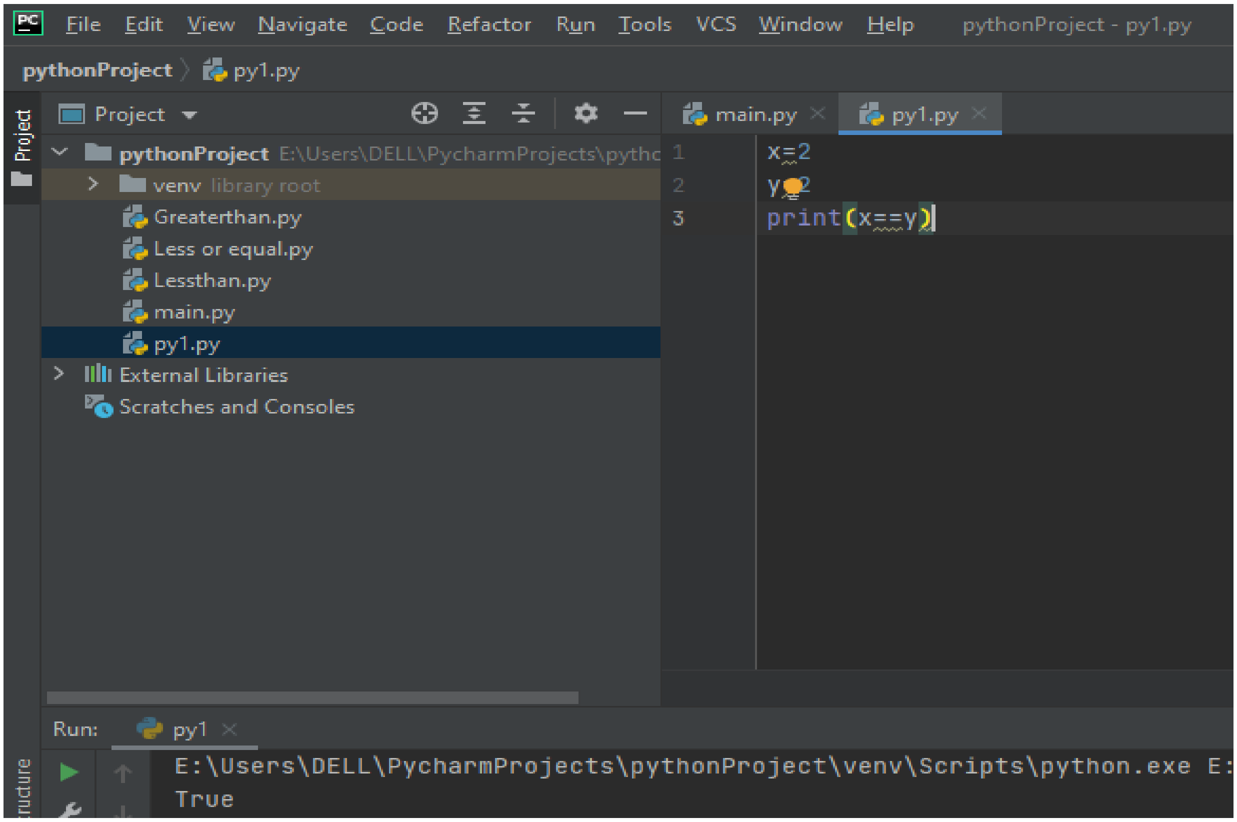
# Greater than or equal to



```
x=4
y=3
print(x>=y)
```

```
E:\Users\DELL\PycharmProjects\pythonProject\venv\Scripts\python.exe E
True
```

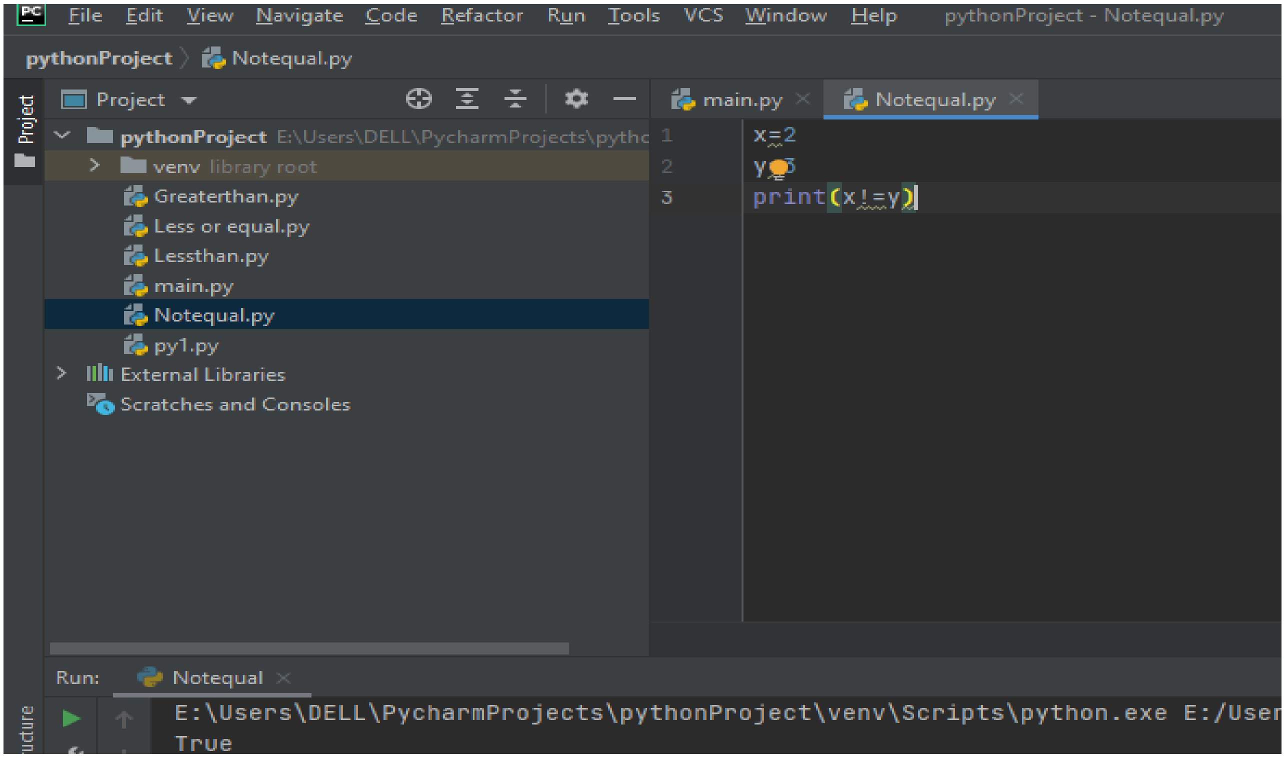
# Equal to



```
x=2
y=2
print(x==y)
```

```
E:\Users\DELL\PycharmProjects\pythonProject\venv\Scripts\python.exe E:
True
```

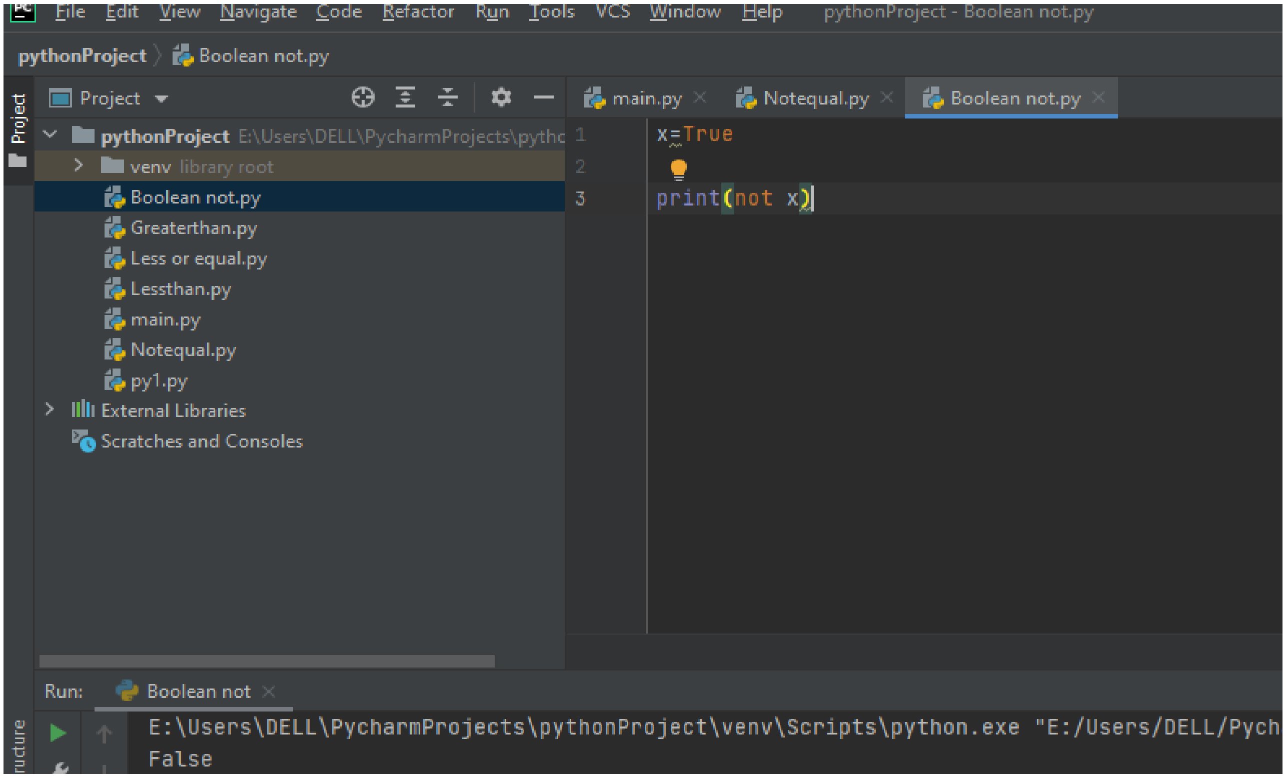
# Not equal to



```
x=2
y=3
print(x!=y)
```

```
E:\Users\DELL\PycharmProjects\pythonProject\venv\Scripts\python.exe E:/User
True
```

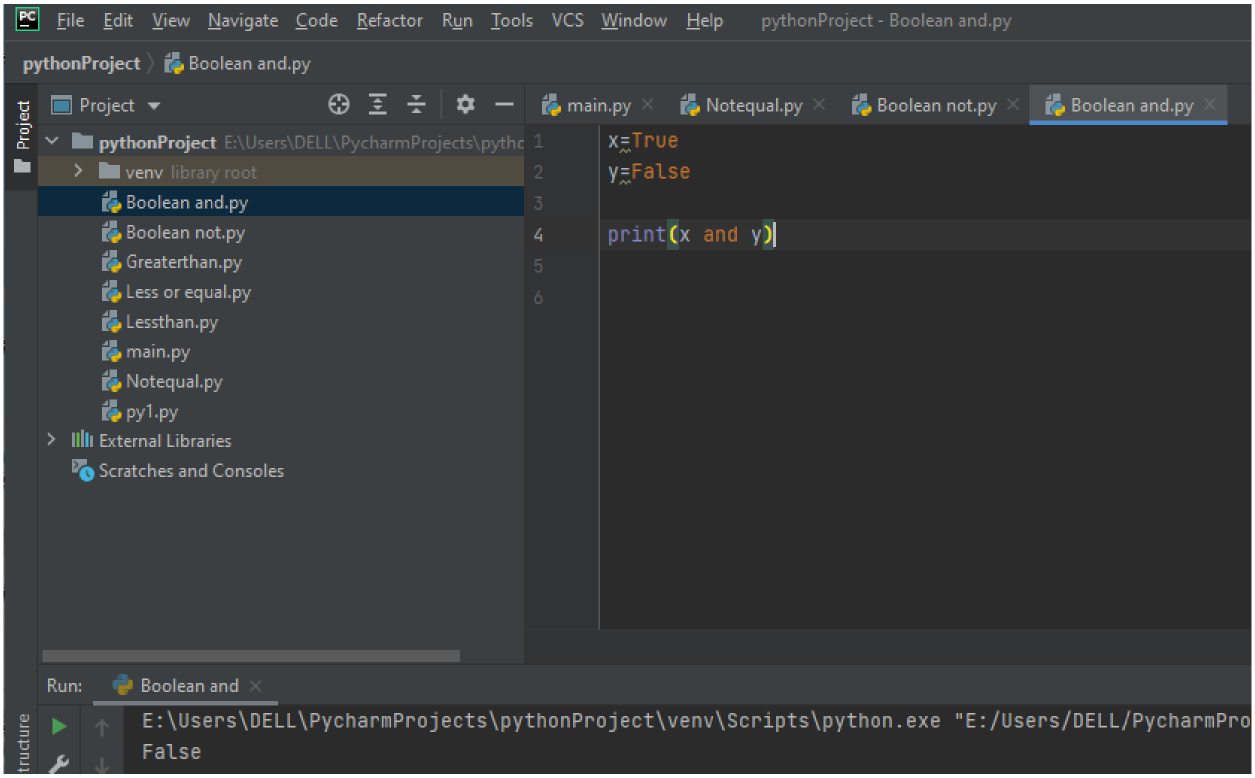
# Boolean NOT



```
x=True

print(not x)
```

```
E:\Users\DELL\PycharmProjects\pythonProject\venv\Scripts\python.exe "E:/Users/DELL/Pych
False
```

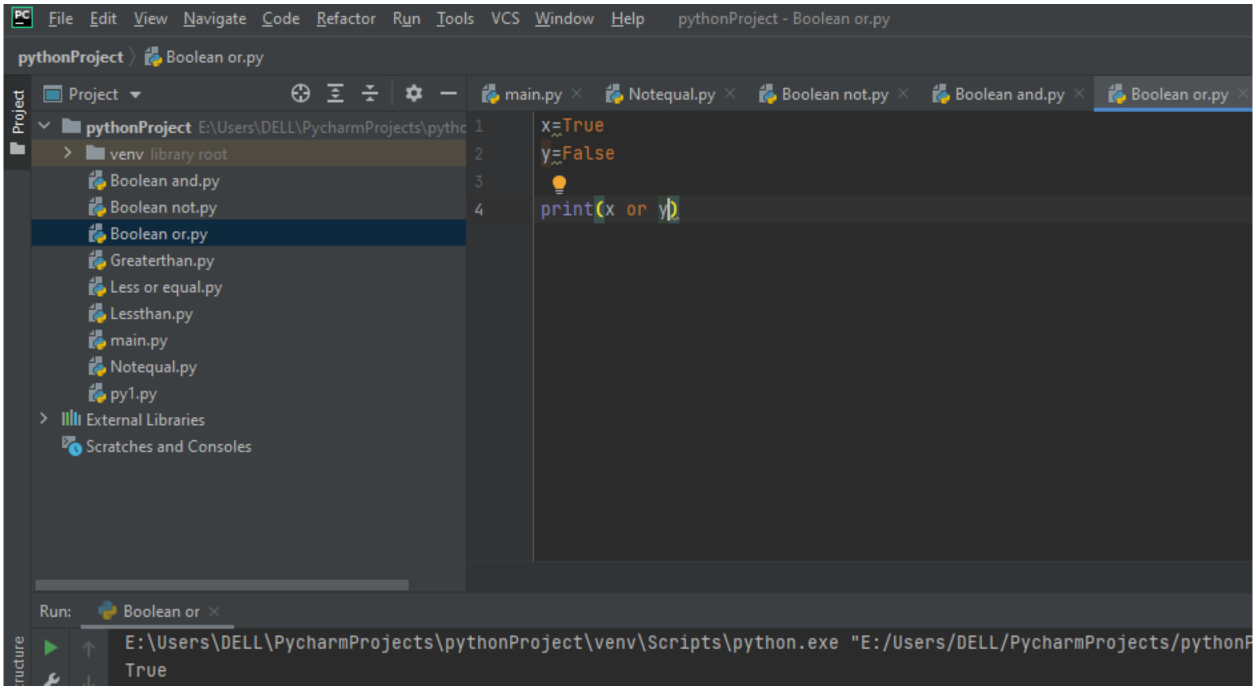
# Boolean AND



```
x=True
y=False

print(x and y)
```

```
E:\Users\DELL\PycharmProjects\pythonProject\venv\Scripts\python.exe "E:/Users/DELL/PycharmPro
False
```

# Boolean OR



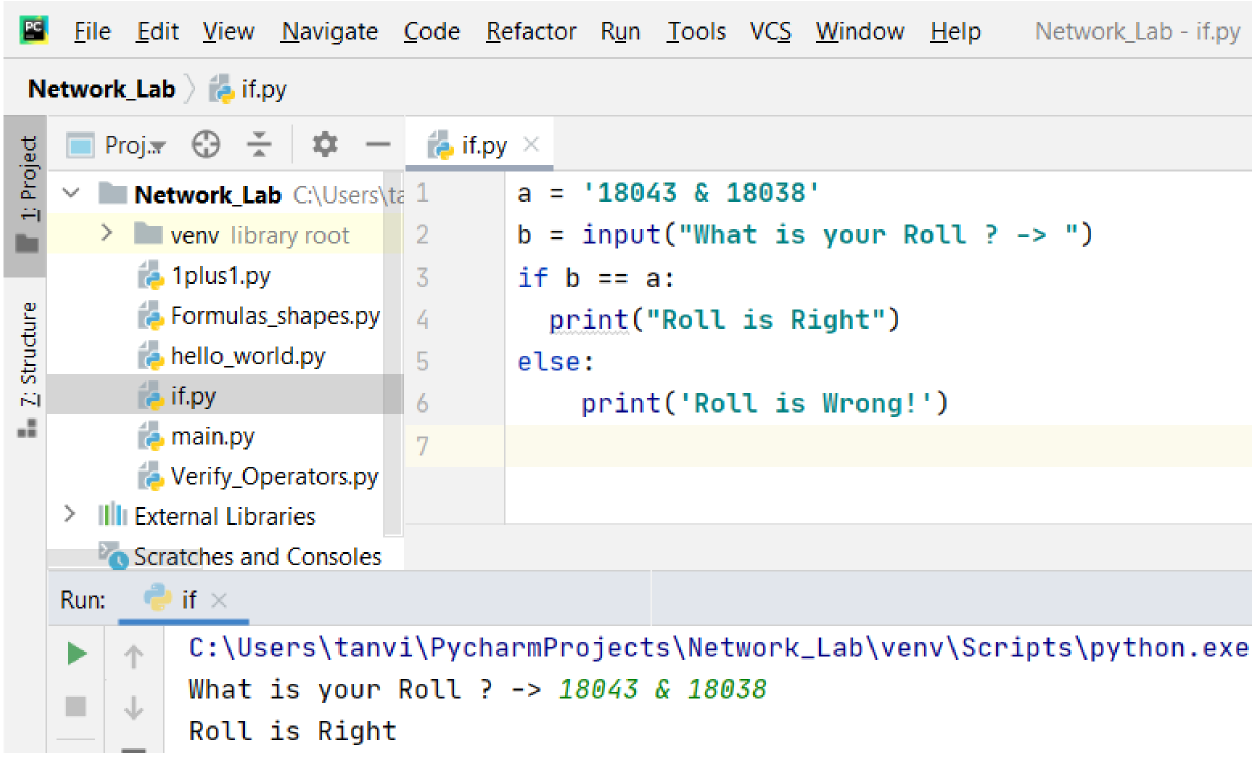
```
x=True
y=False

print(x or y)
```

```
E:\Users\DELL\PycharmProjects\pythonProject\venv\Scripts\python.exe "E:/Users/DELL/PycharmProjects/pythonP
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

The if statement is used to check a condition: if the condition is true, we run a block of statements (called the if-block), else we process another block of statements (called the else-block). The else clause is optional.

Answer:



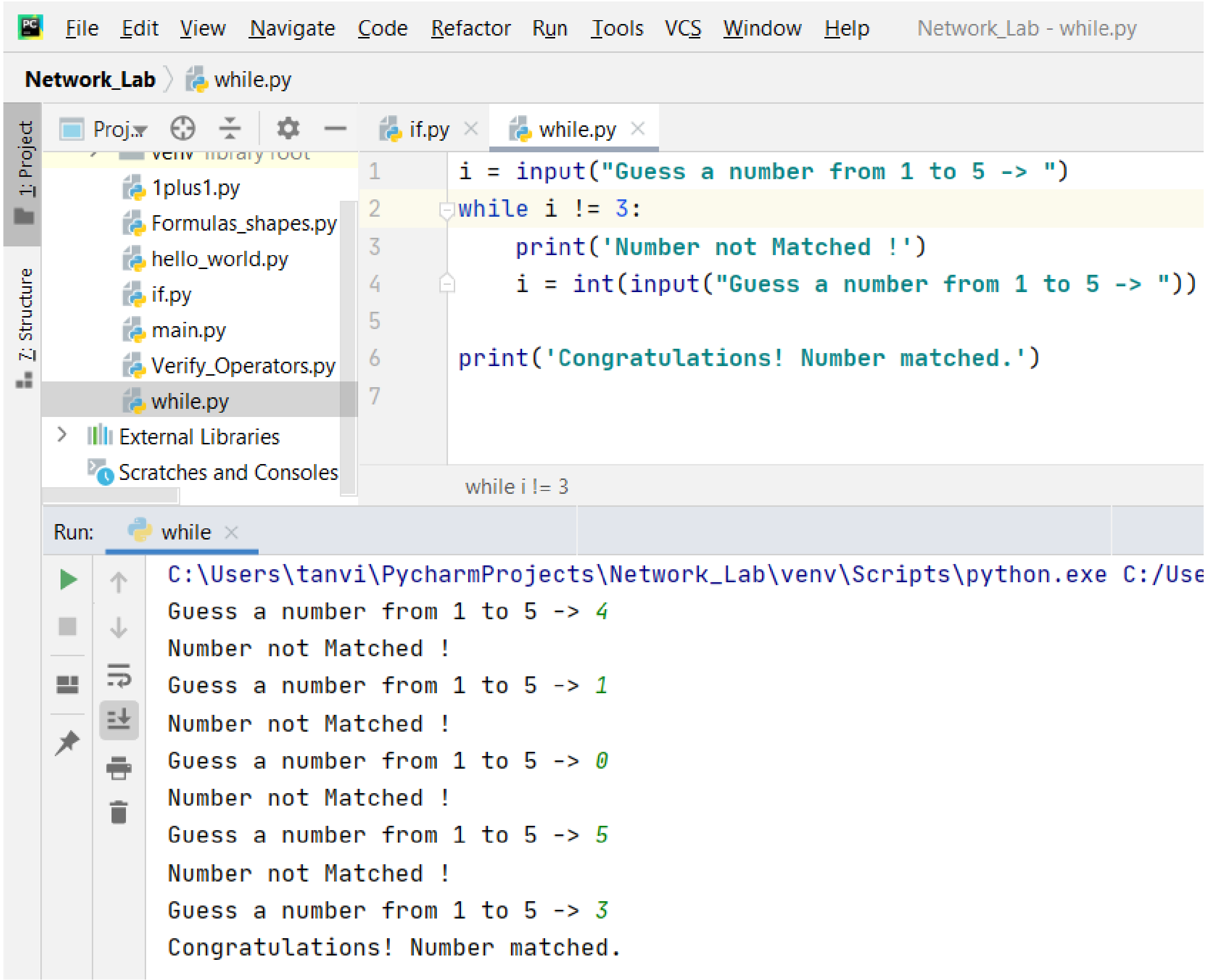
```
a = '18043 & 18038'
b = input("What is your Roll ? -> ")
if b == a:
  print("Roll is Right")
else:
    print('Roll is Wrong!')
```

```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe
What is your Roll ? -> 18043 & 18038
Roll is Right
```

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 runs until the user will guess the number. Save the file as while.py

The while statement allows you to repeatedly execute a block of statements as long as a condition is true. A while statement is an example of what is called a looping statement. A while statement can have an optional else clause.

Answer:



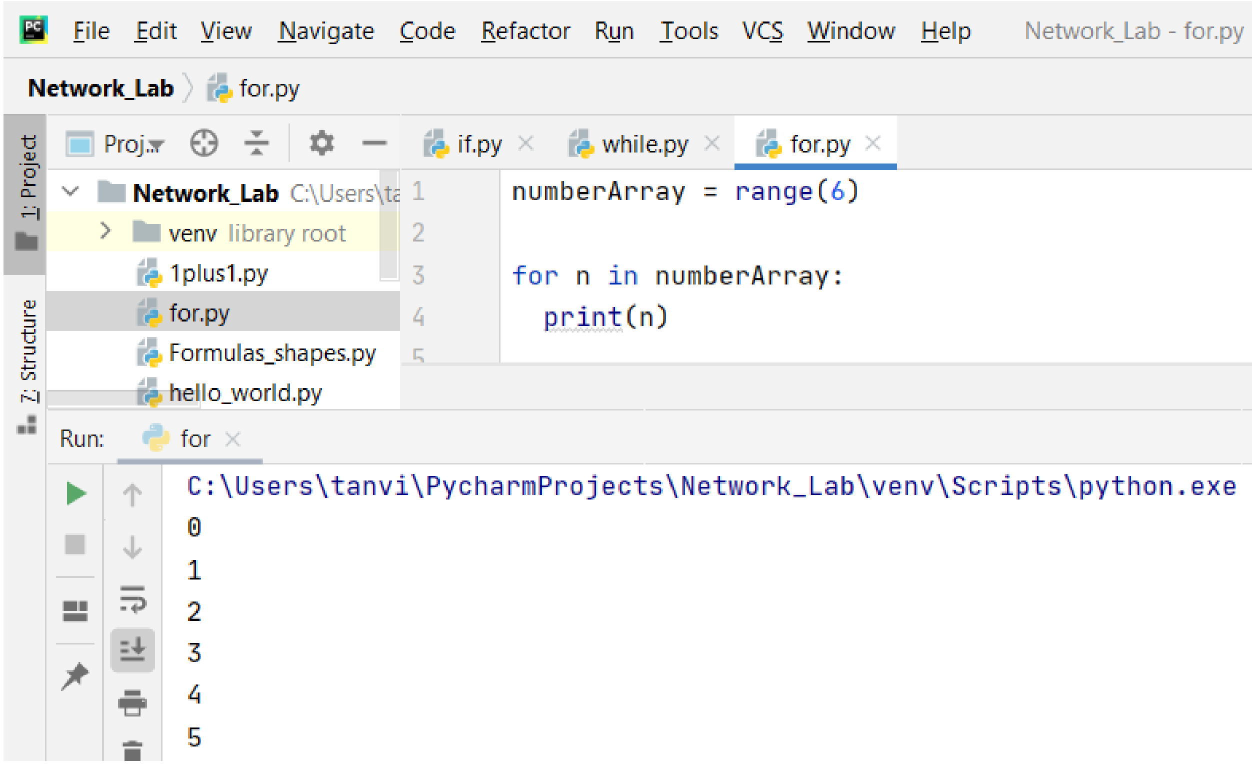
```
i = input("Guess a number from 1 to 5 -> ")
while i != 3:
    print('Number not Matched !')
    i = int(input("Guess a number from 1 to 5 -> "))

print('Congratulations! Number matched.')
```

```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe C:/Use
Guess a number from 1 to 5 -> 4
Number not Matched !
Guess a number from 1 to 5 -> 1
Number not Matched !
Guess a number from 1 to 5 -> 0
Number not Matched !
Guess a number from 1 to 5 -> 5
Number not Matched !
Guess a number from 1 to 5 -> 3
Congratulations! Number matched.
```

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

The for..in statement is another looping statement which iterates over a sequence of objects i.e. go through each item in a sequence.

Answer:



```
numberArray = range(6)

for n in numberArray:
  print(n)
```

```
C:\Users\tanvi\PycharmProjects\Network_Lab\venv\Scripts\python.exe
0
1
2
3
4
5
```