
# Mawlana Bhashani Science & Technology University

Lab Report No : 02

Course Code : ICT3208

Course Title : Computer Network Lab

Report Name : Programming with python

**Submitted by**

Name : Maskur Al Shal Sabil

Id: IT18021

3rd year 2nd Semester

Session 2017-2018

Dept of ICT

MBSTU

**Submitted To**

Nazrul Islam

Assistant Professor

Dept of ICT

MBSTU

## Theory

**Python functions:** Functions are reusable pieces of programs. They allow you to give a name to a block of statements, allowing you to run that block using the specified name anywhere in the program and any number of times. This is known as calling the function.

**Local Variables:** Variables declared inside a function definition are not related in any way to other variables with the same names used outside the function (variable names are local to the function). This is called the scope of the variable. All variables have the scope of the block they are declared in starting from the point of definition of the name.

**The global statement:** Variables defined at the top level of the program are intended global. Global variables are intended to be used in any functions or classes). Global statement allows defining global variables inside functions as well.

**Modules:** Modules allow reusing a number of functions in other programs.

**Networking background for sockets**

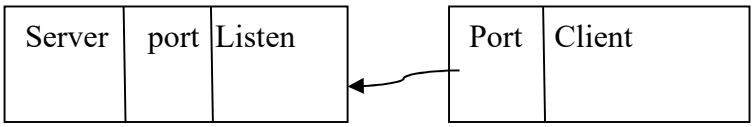
What is a socket and how use it? A socket is one endpoint of a two-way communication link between two programs running on the network or PC. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.

Endpoint: An endpoint is a combination of an IP address and a port number.

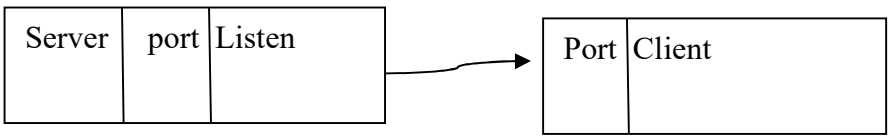
Server and Client: Normally, a server runs on a specific computer and has a socket that is bound to a specific port number.

 **On the server-side:** The server just waits, listening to the socket for a client to make a connection request.

 **On the client-side:** The client knows the hostname of the machine on which the server is running and the port number on which the server is listening. To make a connection request, the client tries to rendezvous with the server on the server's machine and port. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection. This is usually assigned by the system.




If everything goes well, the server accepts the connection. Upon acceptance, the server gets a new socket bound to the same local port and also has its remote endpoint set to the address and port of the client. It needs a new socket so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client.




On the client side, if the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server. The client and server can now communicate by writing to or reading from their sockets

• **TCP**: TCP stands for transmission control protocol. It is implemented in the transport layer of the IP/TCP model and is used to establish reliable connections. TCP is one of the protocols that encapsulate data into packets. It then transfers these to the remote end of the connection using the methods available on the lower layers. On the other end, it can check for errors, request certain pieces to be resent, and reassemble the information into one logical piece to send to the application layer.
The protocol builds up a connection prior to data transfer using a system called a three-way handshake. This is a way for the two ends of the communication to acknowledge the request and agree upon a method of ensuring data reliability. After the data has been sent, the connection is torn down using a similar four-way handshake.
TCP is the protocol of choice for many of the most popular uses for the internet, including WWW, FTP, SSH, and email. It is safe to say that the internet we know today would not be here without TCP.
 **UDP:** UDP stands for user datagram protocol. It is a popular companion protocol to TCP and is also implemented in the transport layer.

The fundamental difference between UDP and TCP is that UDP offers unreliable data transfer. It does not verify that data has been received on the other end of the connection. This might sound like a bad thing, and for many purposes, it is. However, it is also extremely important for some functions. Because it is not required to wait for confirmation that the data was received and forced to resend data, UDP is much faster than TCP. It does not establish a connection with the remote host, it simply fires off the data to that host and doesn't care if it is accepted or not. Because it is a simple transaction, it is useful for simple communications like querying for network resources. It also doesn't maintain a state, which makes it great for transmitting data from one machine to many real-time clients. This makes it ideal for VOIP, games, and other applications that cannot afford delays.

## *3.* Methodology

**Defining functions:** Functions are defined using the ***def*** keyword. After this keyword comes an identifier name for the function, followed by a pair of parentheses which may enclose some names of variables, and by the final colon that ends the line.

```
def XX_YY(variable1, varible2):
# block belonging to the function
# End of function
```

**Defining local and global variables:** Local and global variables can be defined using:

```
 x = 50 #Local
global x
```

**Defining modules:** There are various methods of writing modules, but the simplest way is to create a file with a .py extension that contains functions and variables.

```
def xx_yy():
  aa
```

**Using modules:** A module can be imported by another program to make use of its functionality. This is how we can use the Python standard library as well.

```
import xx_yy
```

## *4.* Exercises

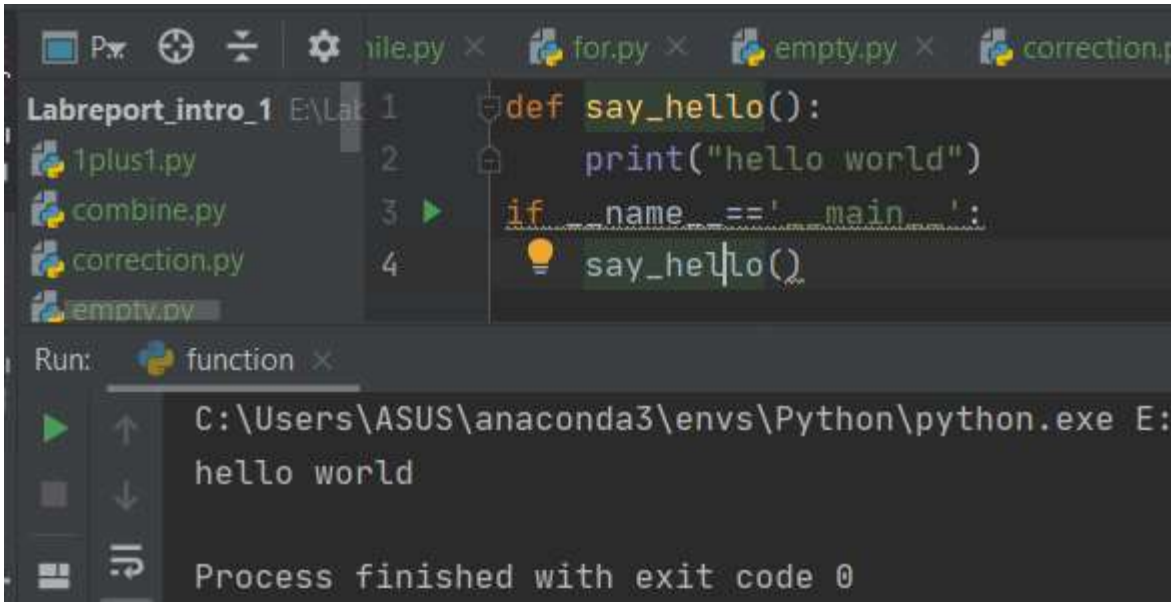
**Section 4.1:** Python function variables and modules.

- **Exercise 4.1.1:** Create a python project using with SDN_LAB
- **Exercise 4.1.2:** Python function (save as function.py)

Create python scrip using the syntax provided below.

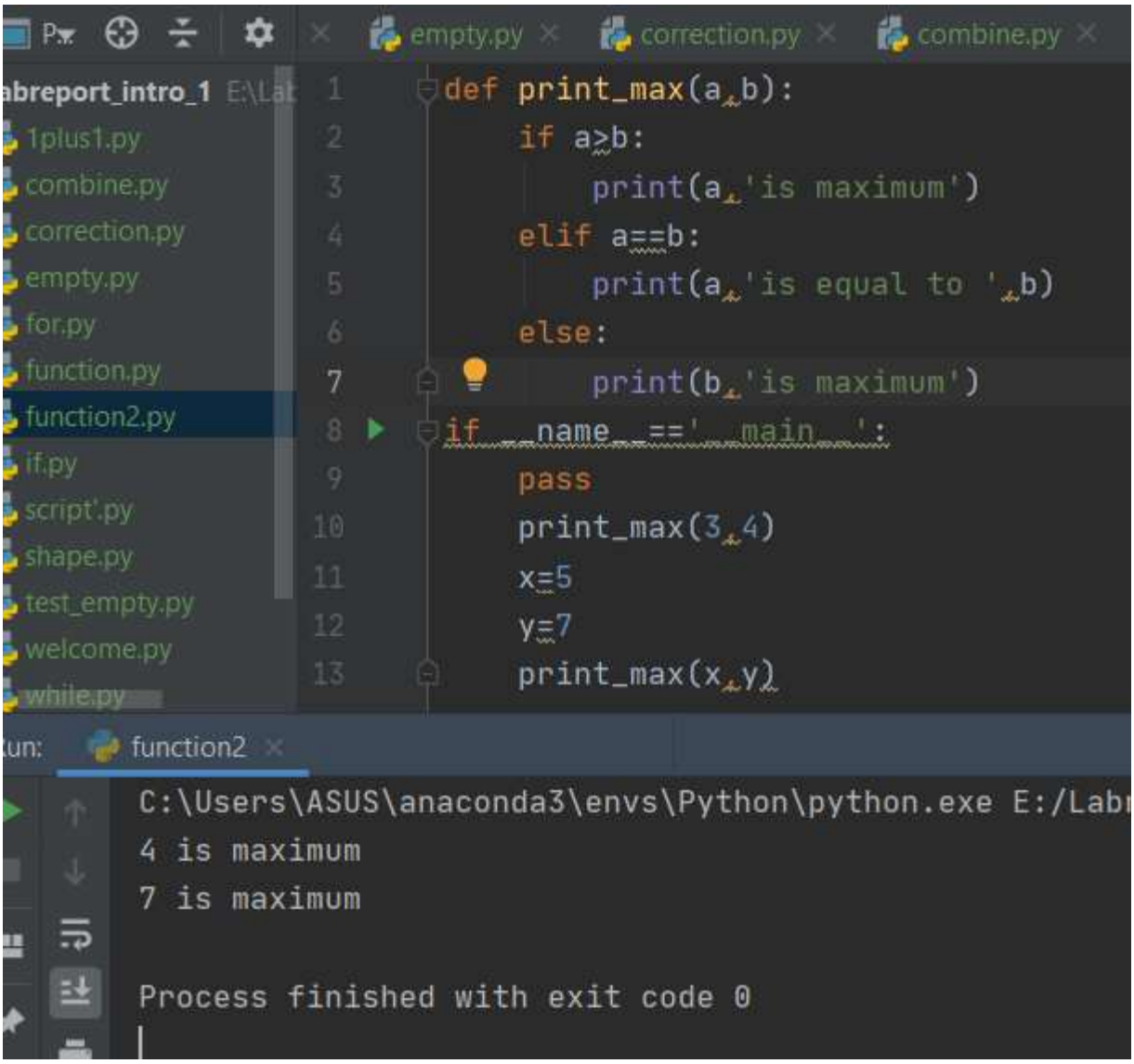
```
def say_hello():
# block belonging to the function
print('hello world')
# End of function
if __name__ == '__main__':
say_hello()
```

Which is the output of this function? Does the function need any parameter?



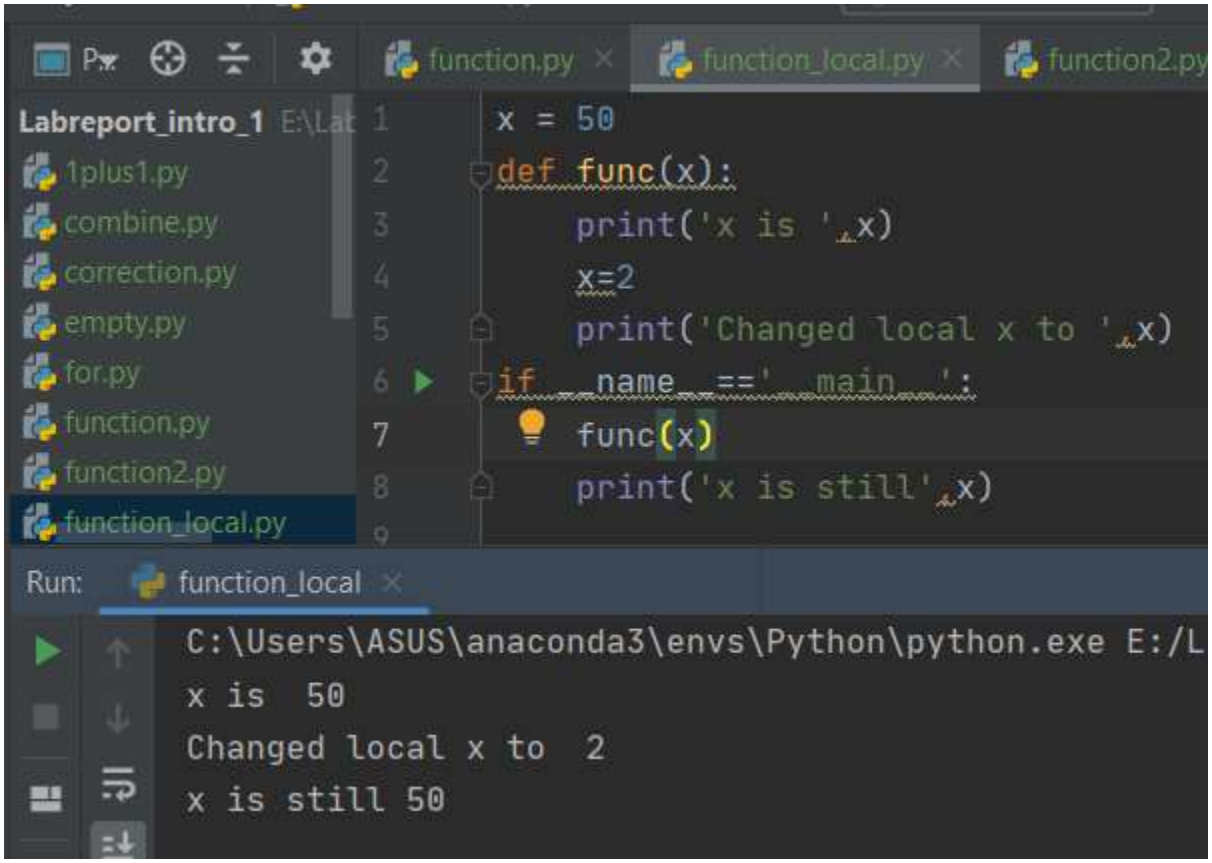

The function print ‘hellow world’ . It doesn’t need any parameter.

**Exercise 4.1.3:** Python function (save as function_2.py)
Create python scrip using the syntax provided below. Which is the output of this function? Does the function need any parameter?



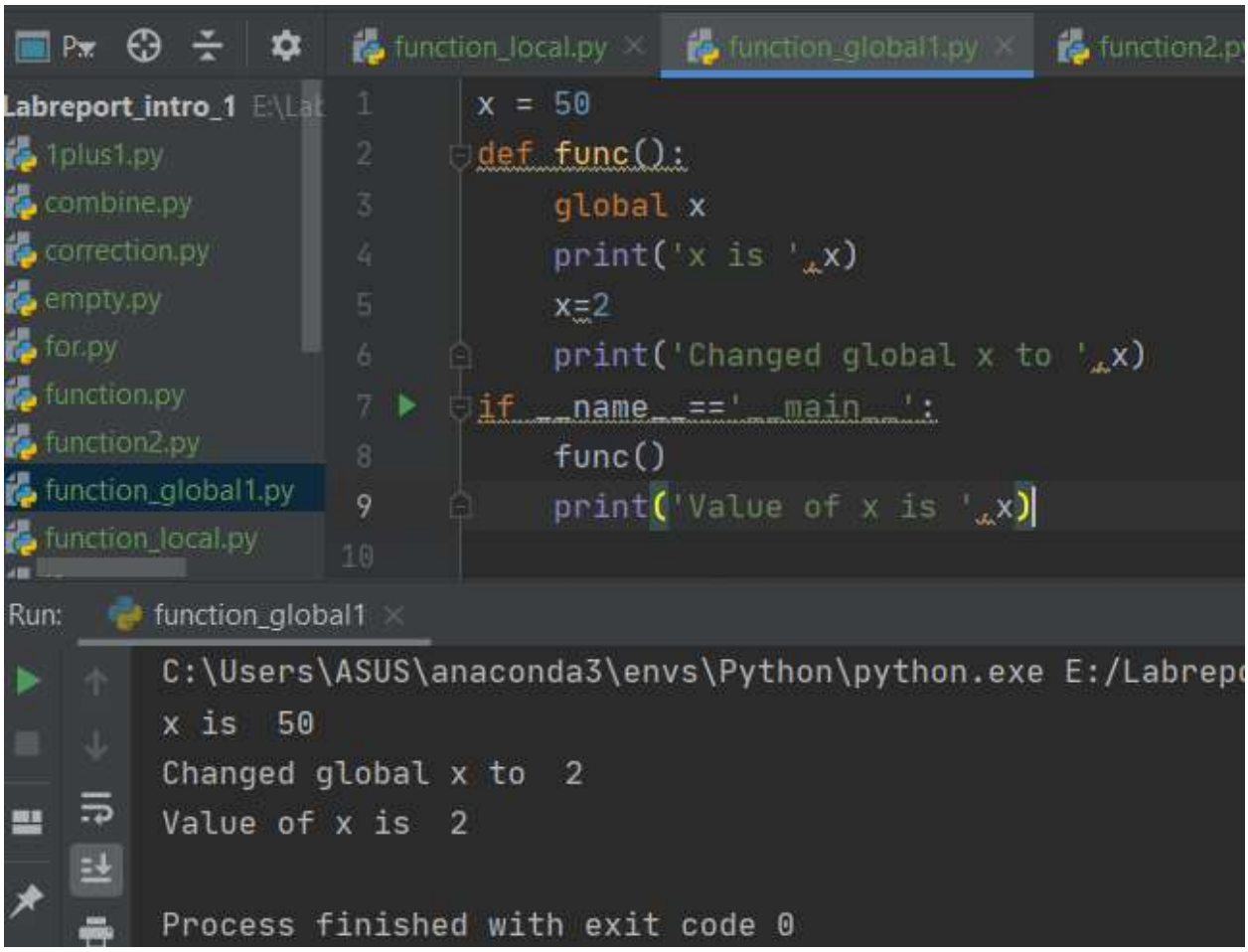

Here the function takes two parameter and the output is : 4 is maximum and 7 is maximum.

**Exercise 4.1.4:** Local variable (save as function_local.py) Create python scrip using the syntax provided below . Which is the final value of variable x? Why variable x does not change to 2?



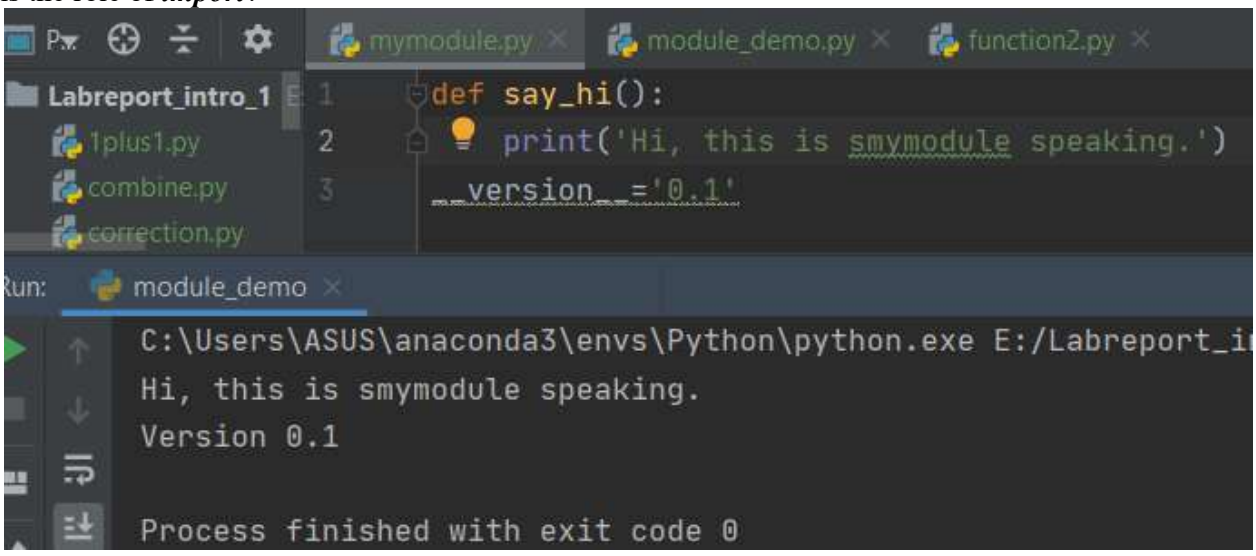

The final value of x is 50 . The value of x doesn't change because it is a local variable and it's scope is under the function .

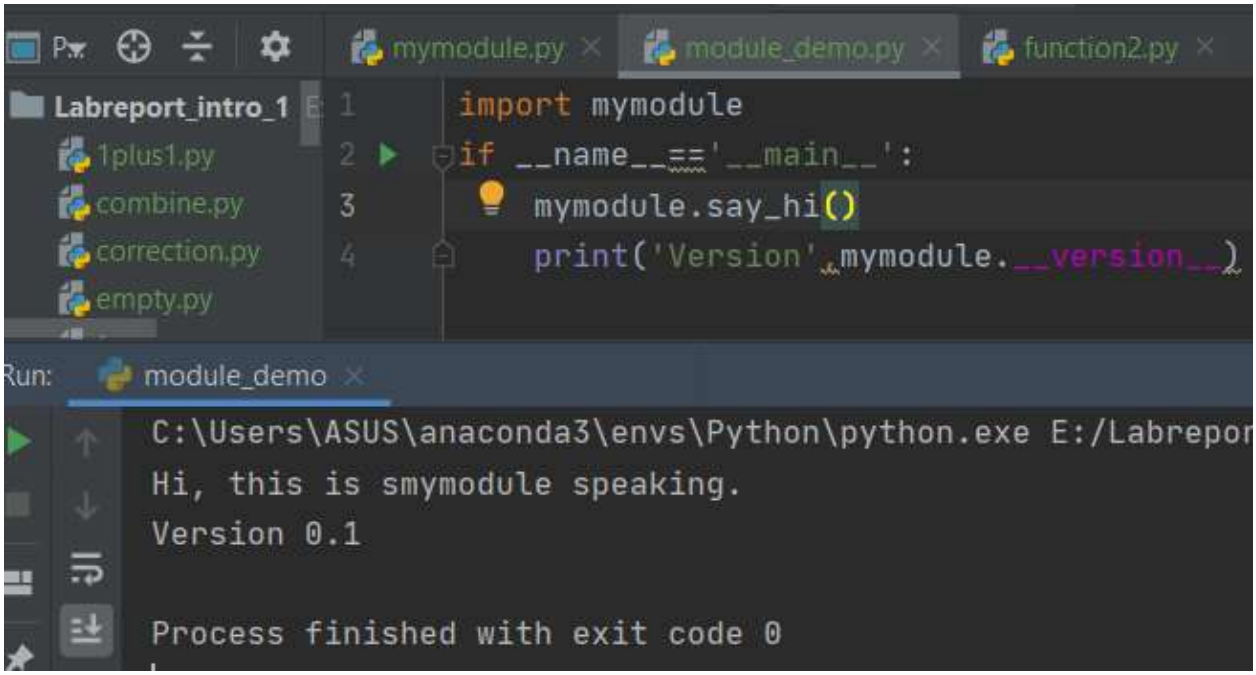
• **Exercise 4.1.5:** Global variable (save as function_global.py) Create python scrip using the syntax provided below. Which is the final value of variable x? Why variable x change this time?



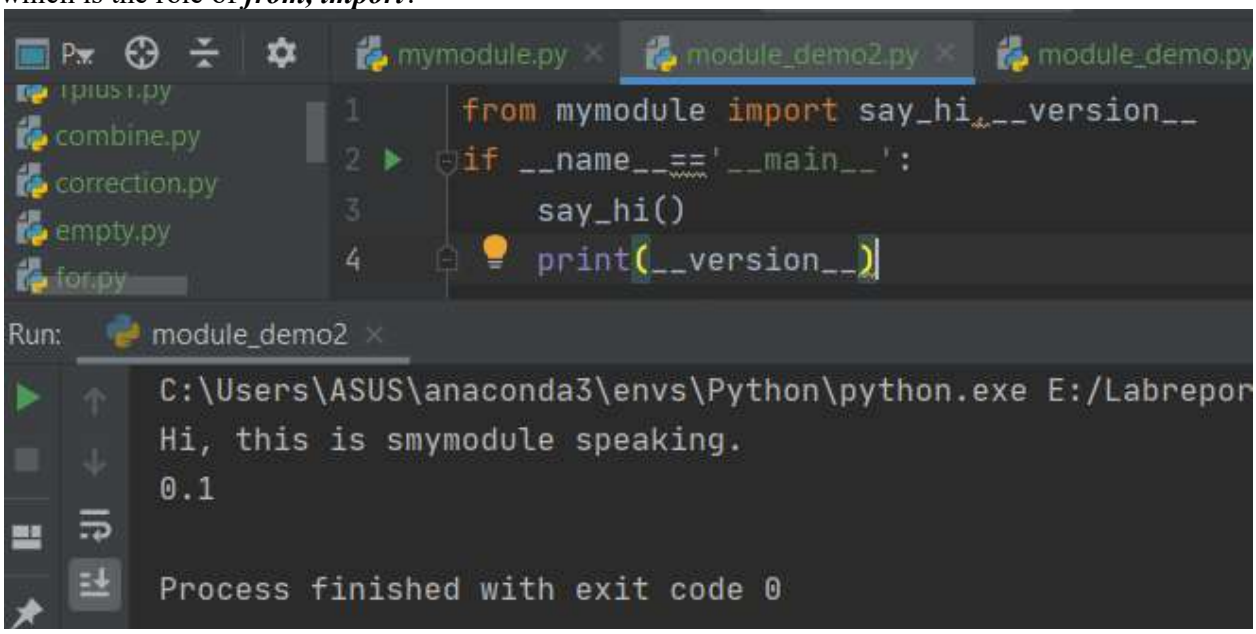

The final value of x is 2 . The value of x changed because of its global declaration.

• **Exercise 4.1.6:** Python modules
Create python script using the syntax provided below (save as mymodule.py). Run the script, which is the role of ***import***?



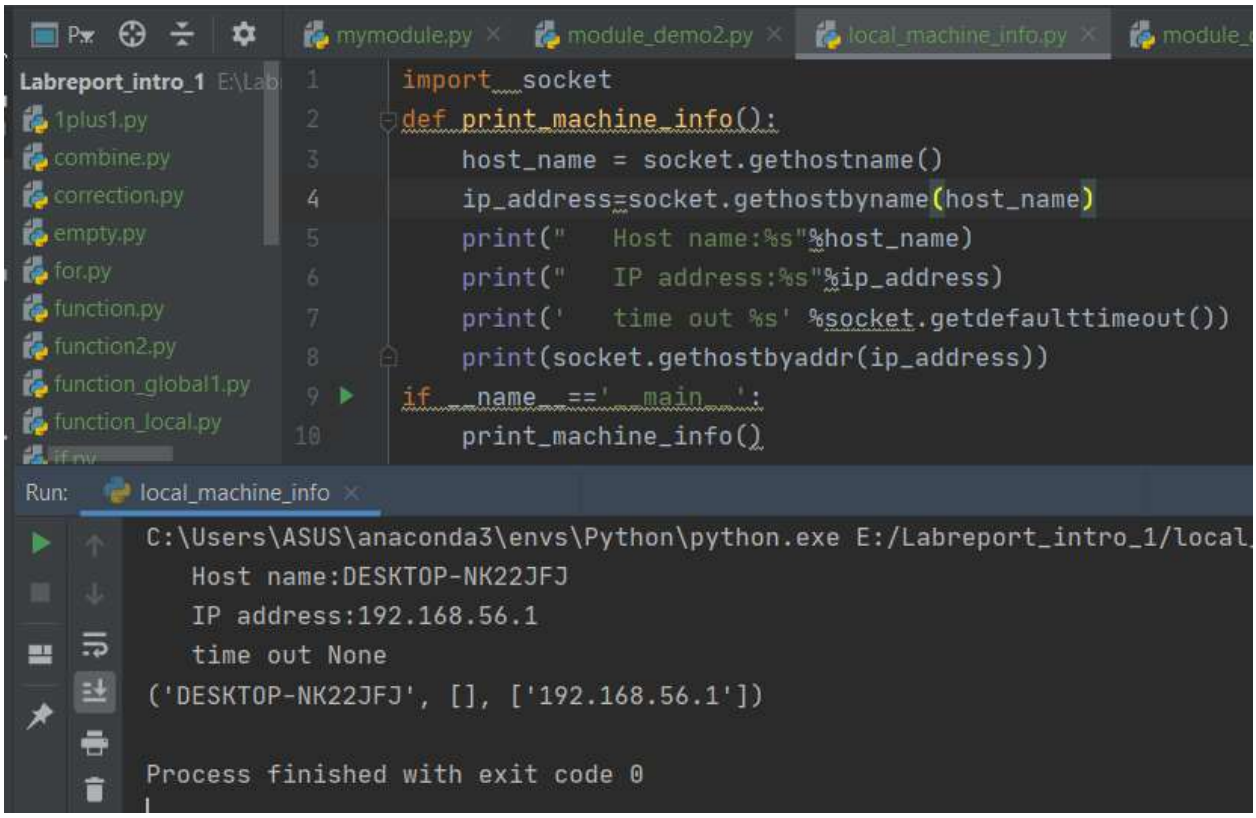



The role of import is that inherit the properties of mymdoule. Thus we can use the component of this module.
Create python scrip using the syntax provided below (save as module_demo2.py). Run the script, which is the role of ***from, import***?



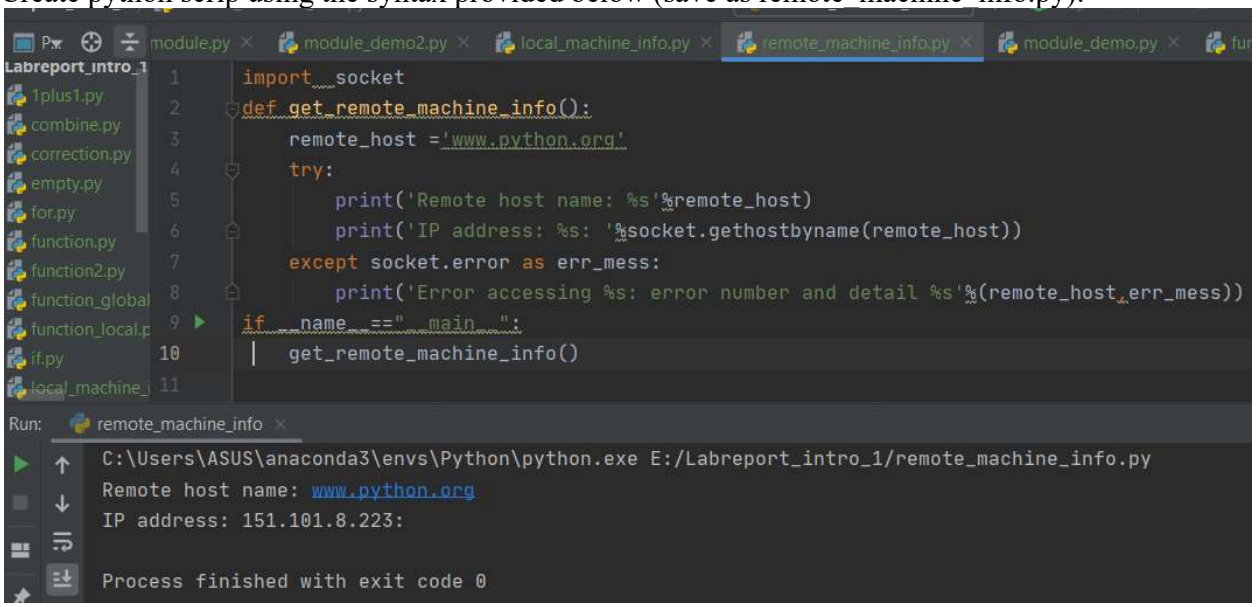

The role of using from , import is that we can now directly use the say_hi() and __version__
**Section 4.2:** Sockets, IPv4, and Simple Client/Server Programming
 **Exercise 4.2.1:** Printing your machine's name and IPv4 address

Create python scrip using the syntax provided below (save as local_machine_info.py): Run the script, which module the program uses? Provide two additional functions of socket?

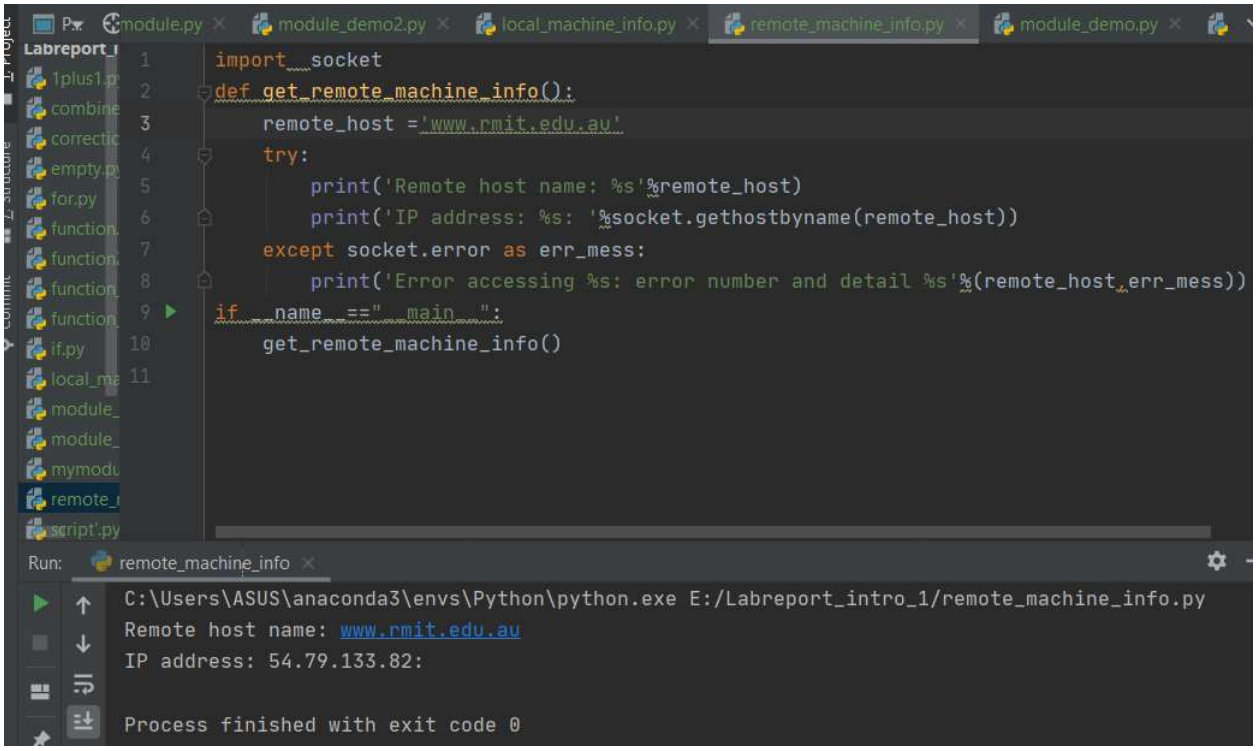



The program use the socket module .

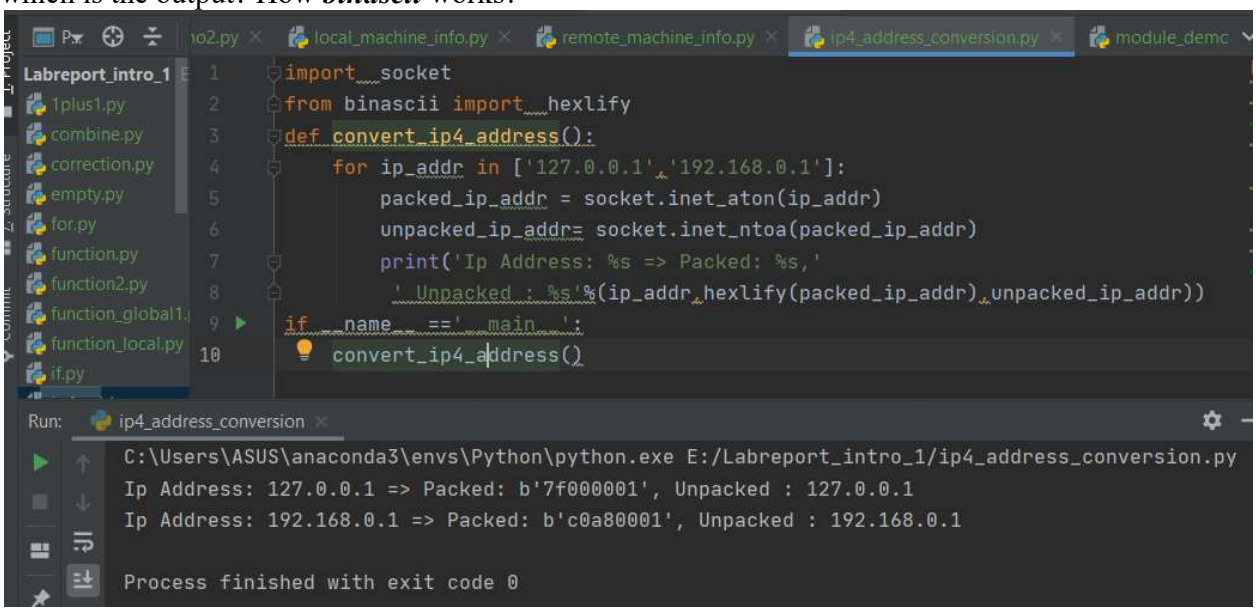
**Exercise 4.2.2:** Retrieving a remote machine's IP address
Create python scrip using the syntax provided below (save as remote_machine_info.py):




Modify the code for getting the RMIT website info.

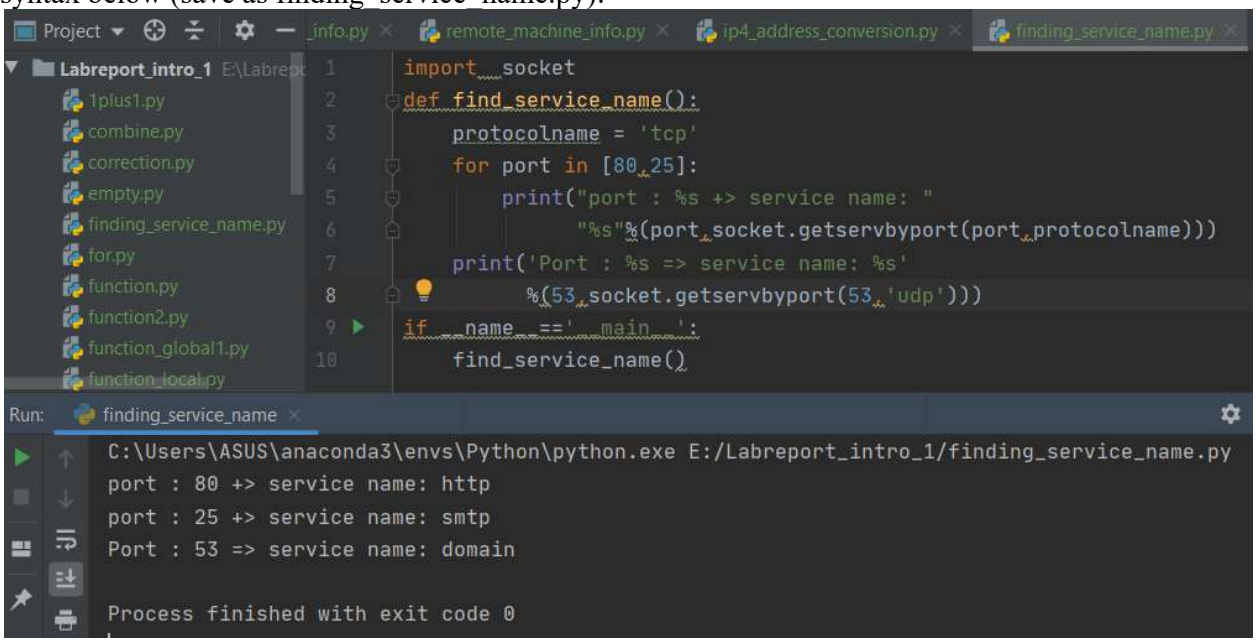



• **Exercise 4.2.3:** Converting an IPv4 address to different formats
Create python scrip using the syntax below (save as ip4_address_conversion.py): Run the script, which is the output? How ***binascii*** works?



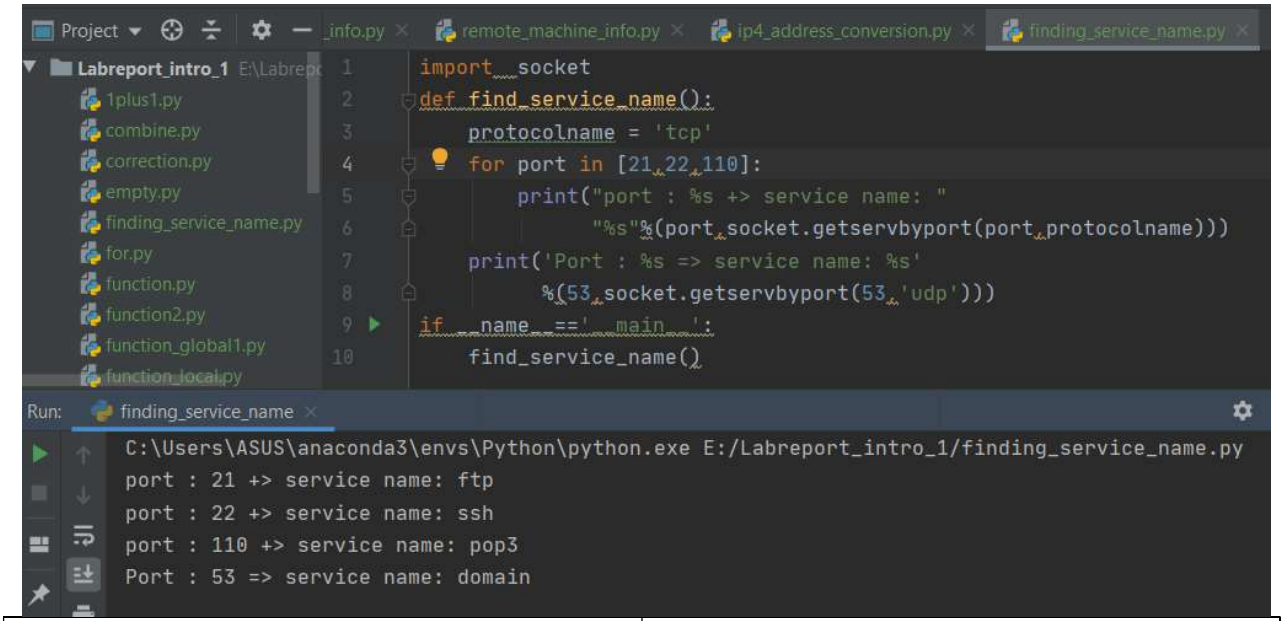

Binascii is a module which contains  a number of methods to convert between binary and various ASCII-encoded binary representation . So using binascii we are able to use  hexlify() function to convert the binary data to hexadecimal representation.

**Exercise 4.2.4:** Finding a service name, given the port and protocol . Create python scrip using the syntax below (save as finding_service_name.py):



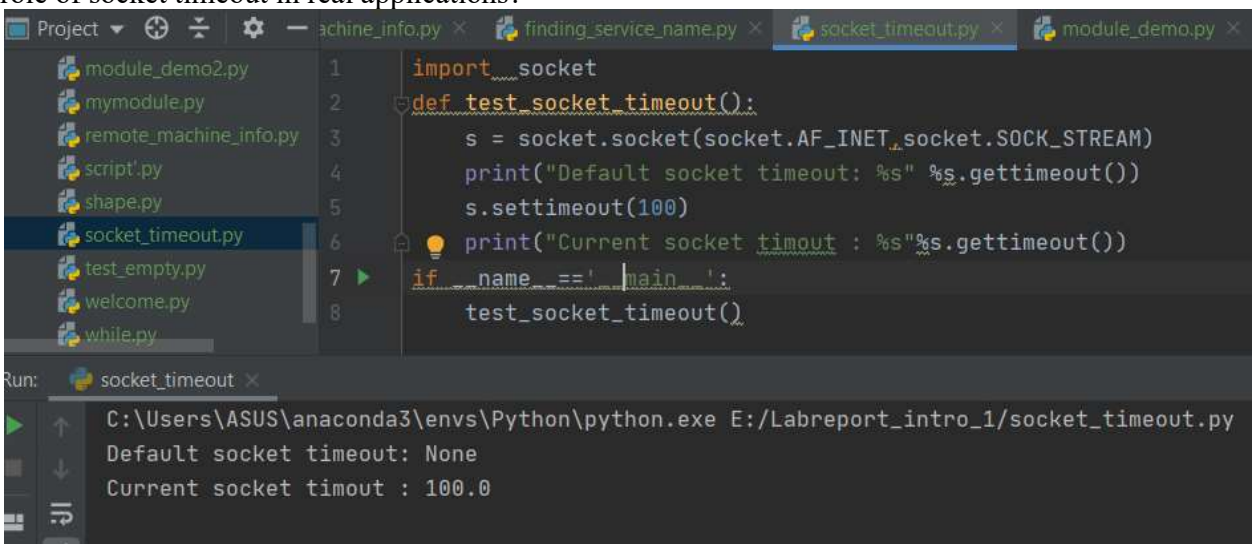

Run the script, which is the output? Modify the code for getting complete the table:

| Port | Protocol Name |
|---|---|
| 21 | |
| 22 | |
| 110 | |




| Port | Protocol Name |
|---|---|
| 21 | ftp |
| 22 | ssh |
| 110 | Pop3 |

**Exercise 4.2.5:** Setting and getting the default socket timeout
Create python scrip using the syntax below (save as socket_timeout.py): Run the script, which is the role of socket timeout in real applications?




The role of socket timeout : A *socket timeout* is the timeout when waiting for individual packets. When a client request a server there is a response time but it is silence that's why it is hanging . In order to avoid the annoying problem of waiting for an undetermined amount of time, sockets (which are responsible for network communication) support a *timeout* option which raises an error based on a time limit

**Exercise 4.2.6:** Writing a simple echo client/server application (**Tip:** Use port 9900)
Create python scrip using the syntax below (save as echo_server.py):

- Run the script, which is the output?

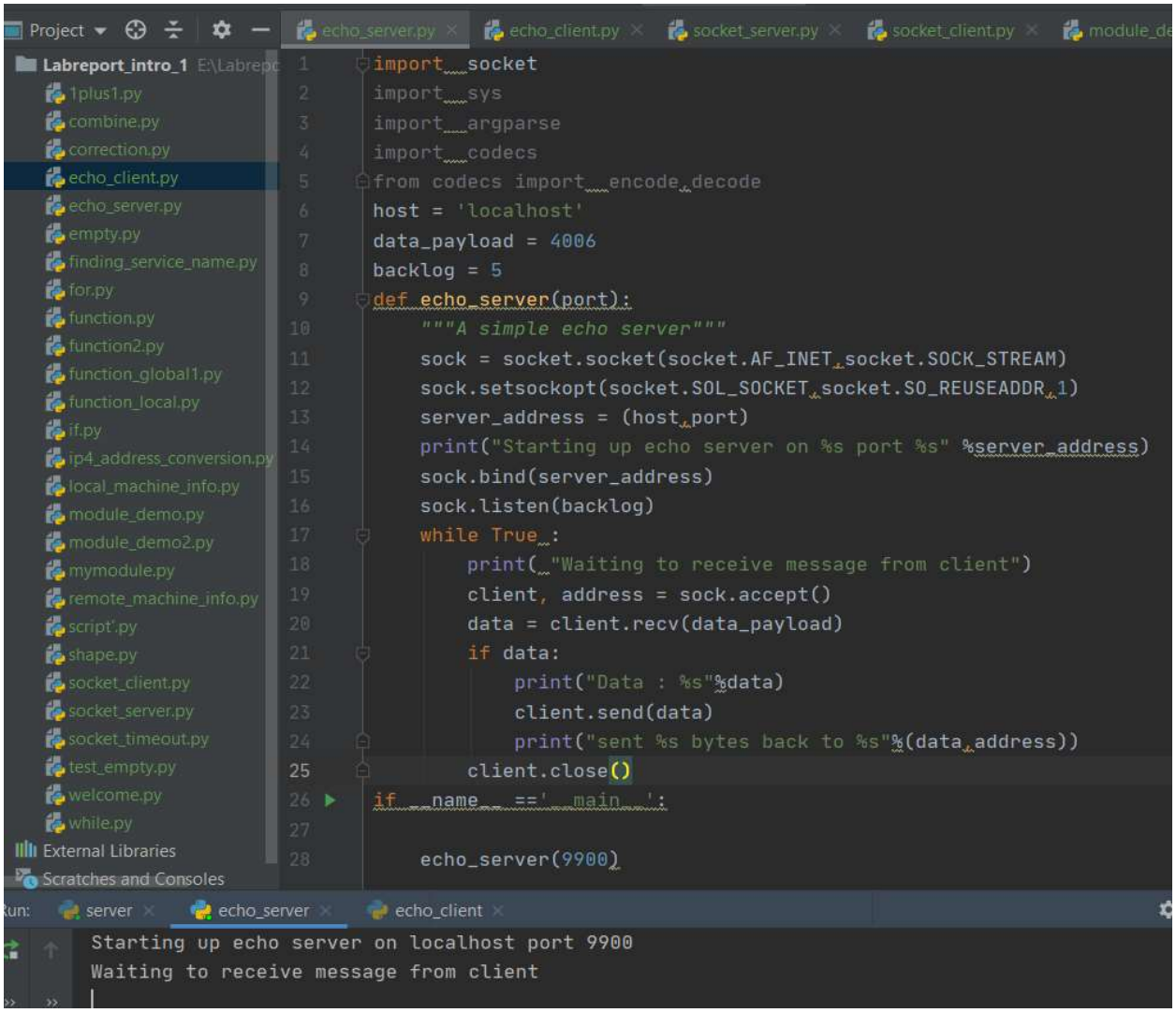
 What you need to do for running the program?

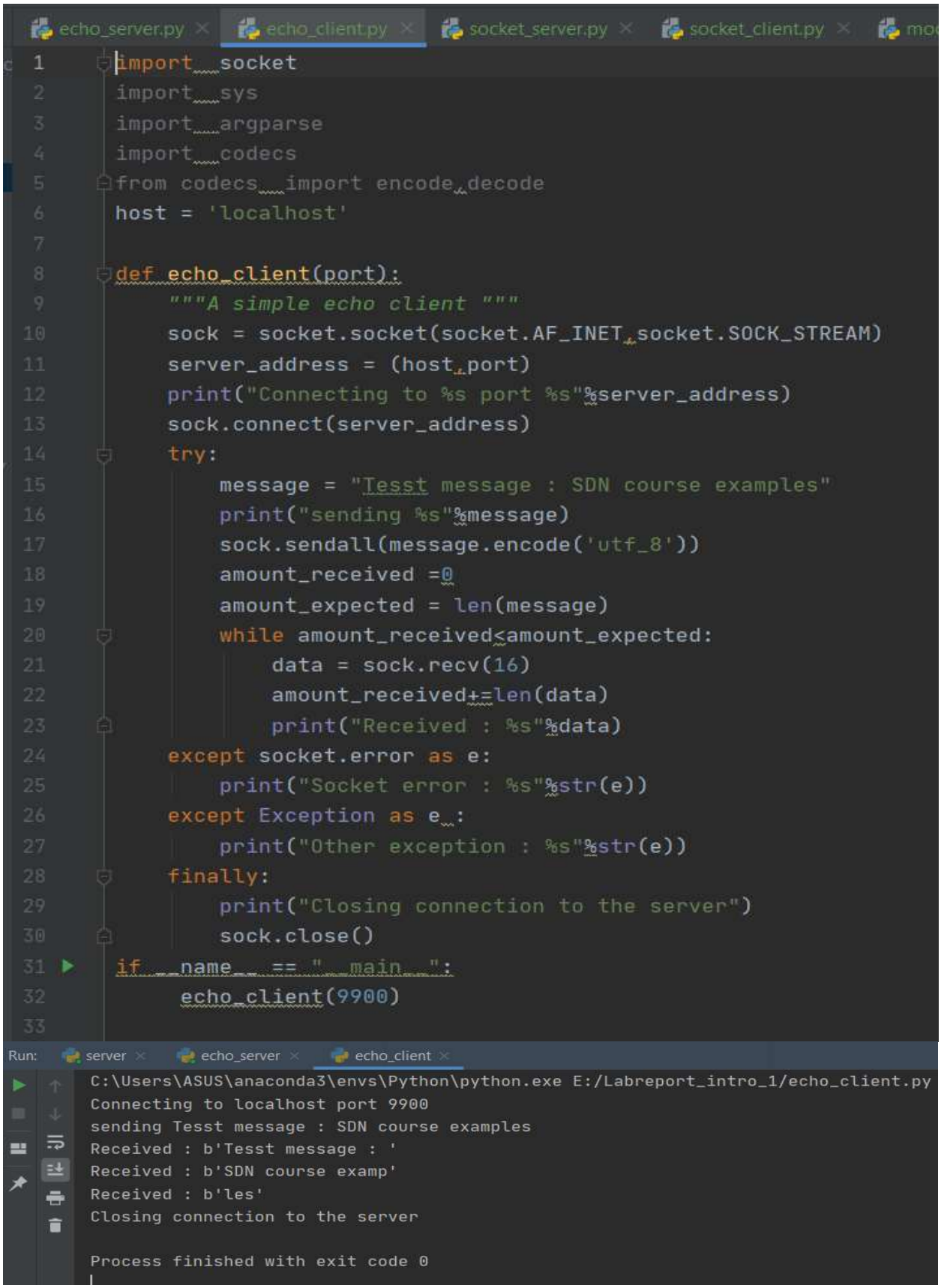
 Which program you need to run first client of server?

First I need to run the echo_server program and then echo_client.

 Explain how the program works?

The server program run on the port 9900 and then it wait for client request .On the other side client run and it request the server program using the same port . In the server program we need to bind the port and host . And their establish a communication .


Project
Labreport_intro_1 E:\Labrepo
1plus1.py
combine.py
correction.py
echo_client.py
echo_server.py
empty.py
finding_service_name.py
for.py
function.py
function2.py
function_global1.py
function_local.py
if.py
ip4_address_conversion.py
local_machine_info.py
module_demo.py
module_demo2.py
mymodule.py
remote_machine_info.py
script'.py
shape.py
socket_client.py
socket_server.py
socket_timeout.py
test_empty.py
welcome.py
while.py
External Libraries
Scratches and Consoles
echo_server.py
echo_client.py
socket_server.py
socket_client.py
import socket
import sys
import argparse
import codecs
from codecs import encode,decode
host = 'localhost'
data_payload = 4006
backlog = 5
def echo_server(port):
"""A simple echo server"""
sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
sock.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
server_address = (host,port)
print("Starting up echo server on %s port %s" %server_address)
sock.bind(server_address)
sock.listen(backlog)
while True:
print("Waiting to receive message from client")
client, address = sock.accept()
data = client.recv(data_payload)
if data:
print("Data : %s"%data)
client.send(data)
print("sent %s bytes back to %s"%(data,address))
client.close()
if __name__ == '__main__':
echo_server(9900)
Run:
server
echo_server
echo_client
Starting up echo server on localhost port 9900
Waiting to receive message from client


echo_server.py
echo_client.py
socket_server.py
socket_client.py
import socket
import sys
import argparse
import codecs
from codecs import encode,decode
host = 'localhost'
def echo_client(port):
"""A simple echo client """
sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
server_address = (host,port)
print("Connecting to %s port %s"%server_address)
sock.connect(server_address)
try:
message = "Tesst message : SDN course examples"
print("sending %s"%message)
sock.sendall(message.encode('utf_8'))
amount_received =0
amount_expected = len(message)
while amount_received<amount_expected:
data = sock.recv(16)
amount_received+=len(data)
print("Received : %s"%data)
except socket.error as e:
print("Socket error : %s"%str(e))
except Exception as e:
print("Other exception : %s"%str(e))
finally:
print("Closing connection to the server")
sock.close()
if __name__ == "__main__":
echo_client(9900)
Run:
server
echo_server
echo_client
C:\Users\ASUS\anaconda3\envs\Python\python.exe E:/Labreport_intro_1/echo_client.py
Connecting to localhost port 9900
sending Tesst message : SDN course examples
Received : b'Tesst message : '
Received : b'SDN course examp'
Received : b'les'
Closing connection to the server
Process finished with exit code 0

 What you need to do for communicating with another server in the classroom?
To communicate with another server of the classroom we can use the ip address of that computer and make sure that both computer are in Local network . so using the local network ip we can communicate with each other.

## *5.* Questions

 **Question 5.1:** Explain in your own words which are the difference between functions and modules?

The modules can be used in the different program where a function is in a single program . To use function in different program it must be included under a module. A module may have many function and variable that can be use by declaring using from keyword

 **Question 5.2:** Explain in your own words when to use local and global variables?

The local variable is use for temporary use which value may change and same within a certain scope but global variable all time takes a common value in all the program .

 **Question 5.3:** Which is the role of sockets in computing networking? Are the sockets defined random or there is a rule?

The socket is a way to bind the client side and server side so that the communication between two or more nodes can be possible. It is the endpoint of the server and client . Y

Yes the socket follow the specific rule to establish the connection.

 **Question 5.4:** Why is relevant to have the IPv4 address of remote server? Explain what is Domain Name System (DNS)?

Simply put to communicate via the internet we need a source and destination address and a way to get from the source to destination. So every device , end point whatever is connected has an assigned Ip address as a way to identify our source and destination. The internet can then be used to route packets of data between source and destination ( including diversions) . IPv4, was created back in the 1970s. Vint Cerf led the team who invented it. It however is made up of four 8-bit numbers or 32-bits total, providing for 4.3 billion addresses. It also support DNS .

The Domain Name System (DNS) supports both IPV4 and IPV6s. The DNS stores the IP addresses for either or both and responds to each request for domain name resolution with both IP addresses (A site can have multiple addresses for either protocol).

DNS puts IPv4 addresses into the A record. The DNS system stores IPv6 addresses in the AAAA record.

The client can then decide which protocol to use.

 **Question 5.5:** Create a program that allows exchange messages increased by the user between client and server.

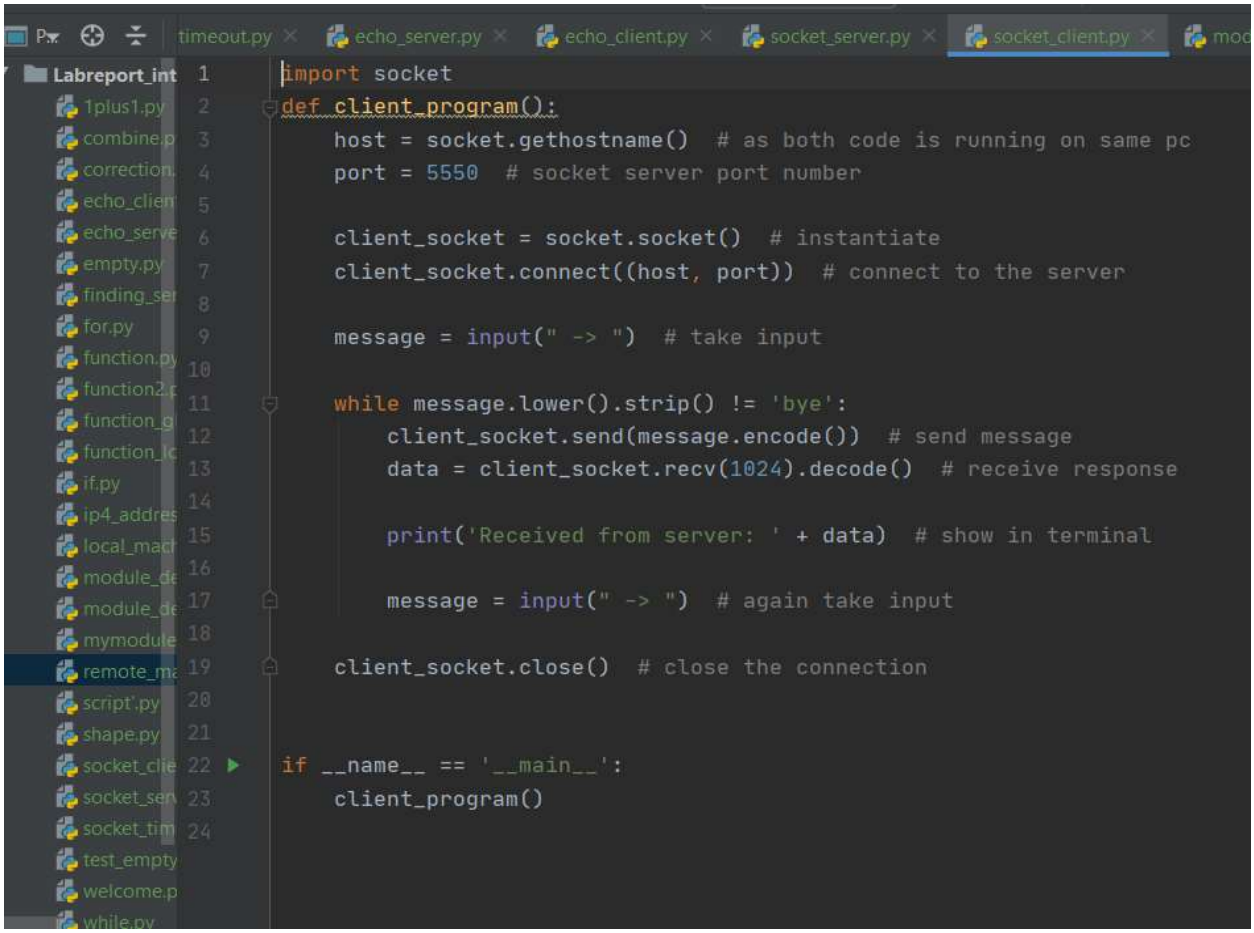
Server side code :

finding_service_name.py | socket_timeout.py | echo_server.py | echo_client.py | socket_server.py

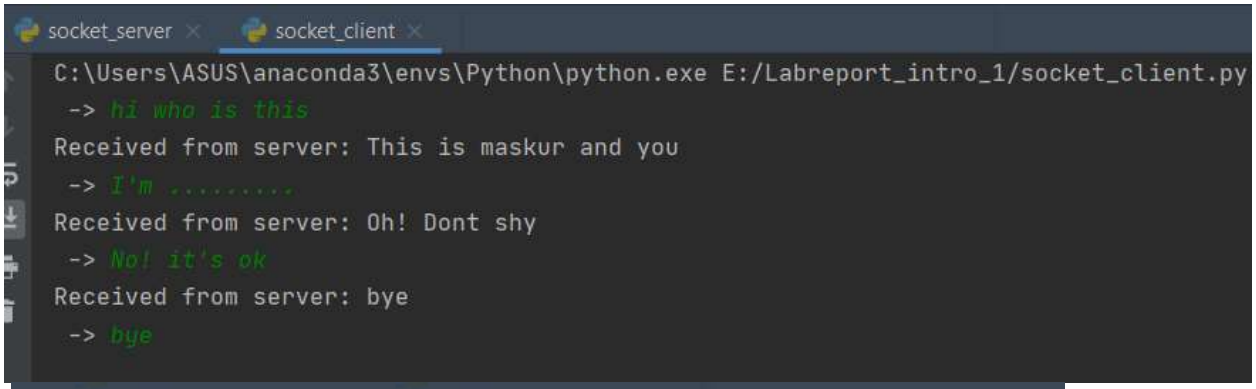
```
import socket
def server_program():
    # get the hostname
    host = socket.gethostname()
    print('host name',host)
    port = 5550  # initiate port no above 1024

    server_socket = socket.socket()  # get instance
    # look closely. The bind() function takes tuple as argument
    server_socket.bind((host, port))  # bind host address and port together

    # configure how many client the server can listen simultaneously
    server_socket.listen(2)
    conn, address = server_socket.accept()  # accept new connection
    print("Connection from: " + str(address))
    while True:
        # receive data stream. it won't accept data packet greater than 1024
        data = conn.recv(1024).decode()
        if not data:
            # if data is not received break
            break
        print("from connected user: " + str(data))
        data = input(' -> ')
        conn.send(data.encode())  # send data to the client

    conn.close()  # close the connection


if __name__ == '__main__':
    server_program()
```

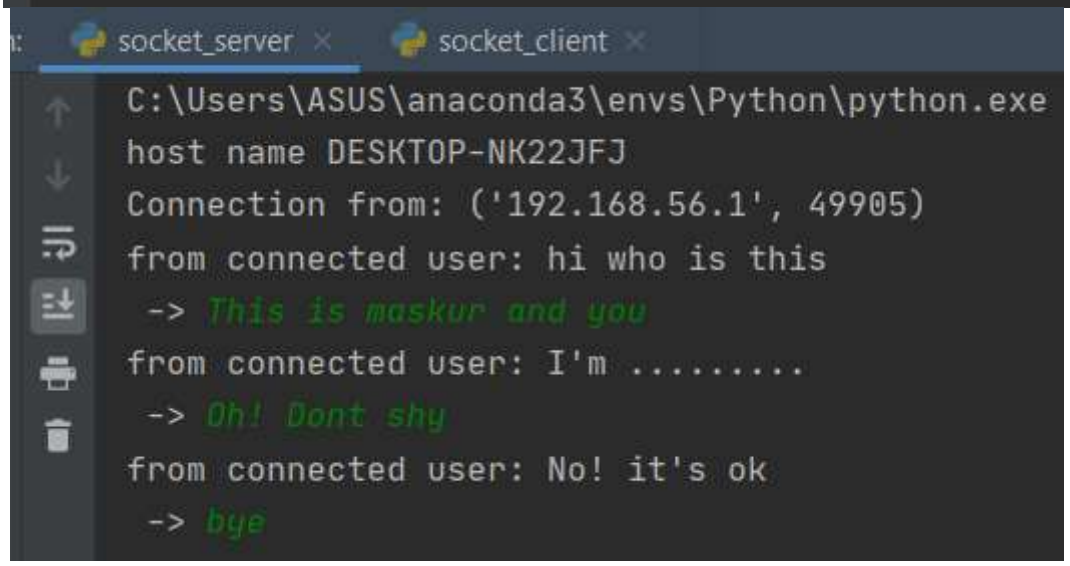
PyCharm 2020.1.4 available

Client side code :




The output :




Conclusion : To do this lab report I have taken help from the slide given by my course teacher .That was so good and I have learned many things about network .Besides I also taken help from Internet resource to construct the last exercise . In this lab report I have learned how to get the ip address of my computer and other host . And communicate a server code and a client code . I have learned how the data transfer from one host to another and I enjoyed this lab . .