Mawlana Bhashani Science and Technology University

Lab-Report

Lab Report No: 02

Lab Report Name: Programming with Python.

Course Title: Network Planning and Designing Lab

Course code: ICT-3208

Submitted by

Name: Md Masud Rana

ID: IT-18002

3rd year 2nd semester

Session: 2017-18

Dept. of ICT

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT,

MBSTU.

1. Objectives :

The objective of the lab 2 is to:

- Understand how python function works
- Understand the use of global and local variables
- Understand how python modules works
- Learning the basics of networking programming with python

2. 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.

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
```

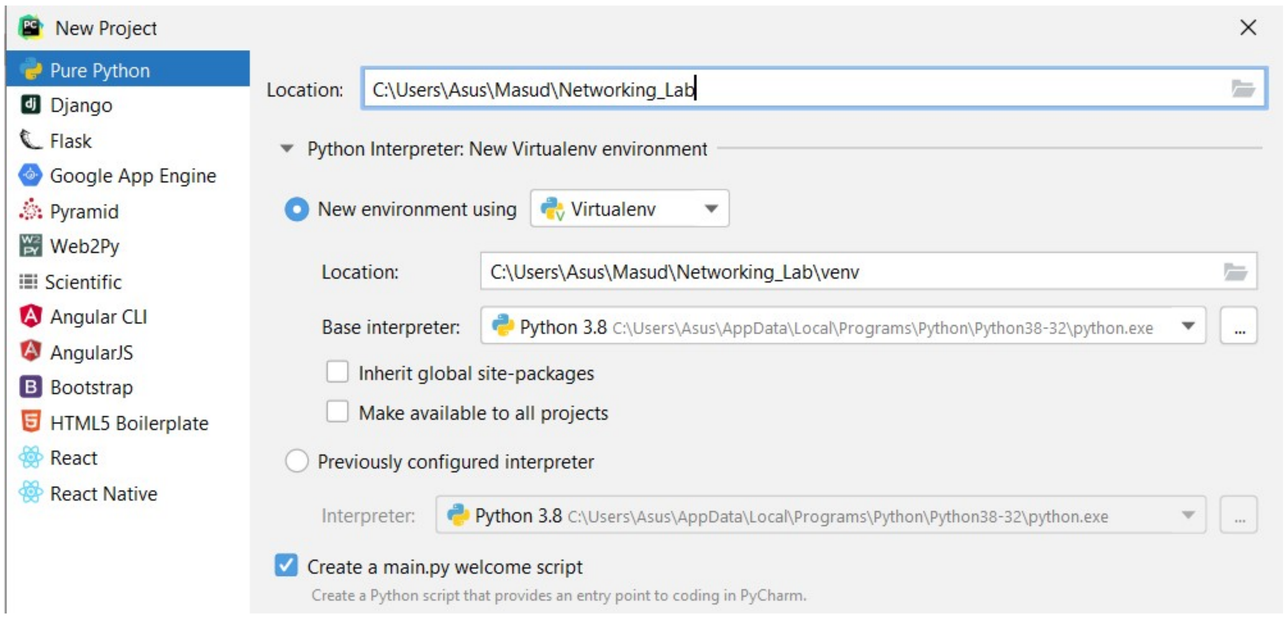
Using modules: A module can be imported by another program to make use of its functionality. This is how we can use the Python standard library as well.

```
import xx_yy
```

4. Exercises

Section 4.1: Python function variables and modules.

Exercise 4.1.1: Create a python project



New Project

Location: C:\Users\Asus\Masud\Networking_Lab

Python Interpreter: New Virtualenv environment

New environment using Virtualenv

Location: C:\Users\Asus\Masud\Networking_Lab\venv

Base interpreter: Python 3.8 C:\Users\Asus\AppData\Local\Programs\Python\Python38-32\python.exe

Inherit global site-packages

Make available to all projects

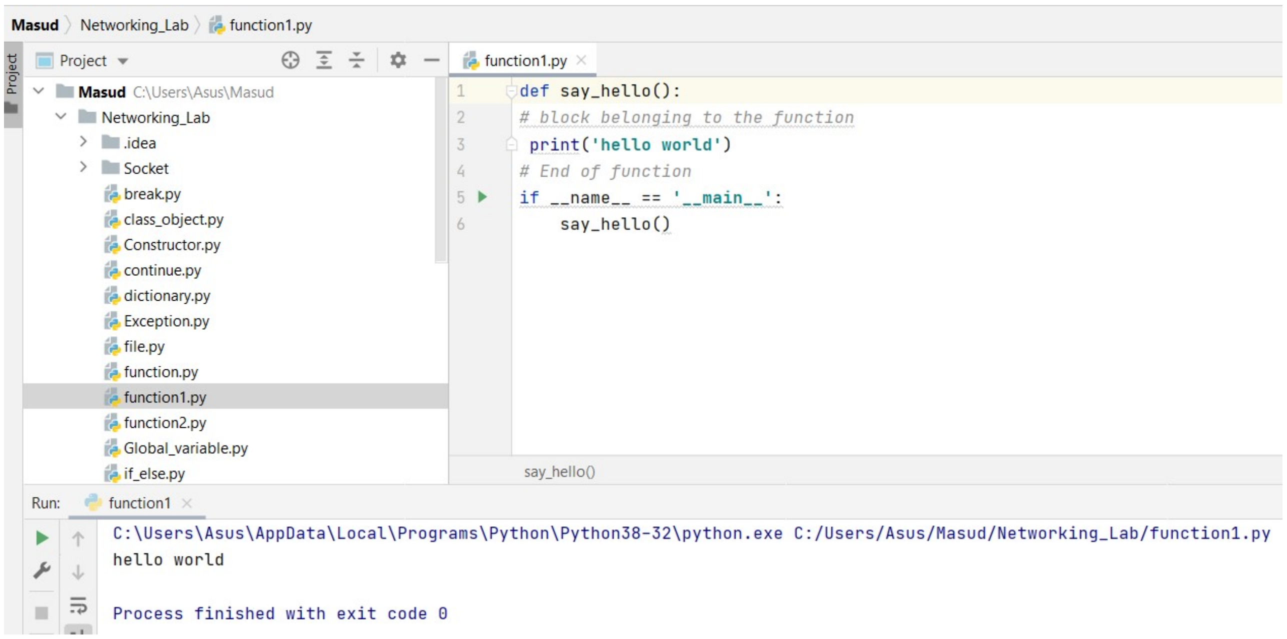
Previously configured interpreter

Interpreter: Python 3.8 C:\Users\Asus\AppData\Local\Programs\Python\Python38-32\python.exe

Create a main.py welcome script

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

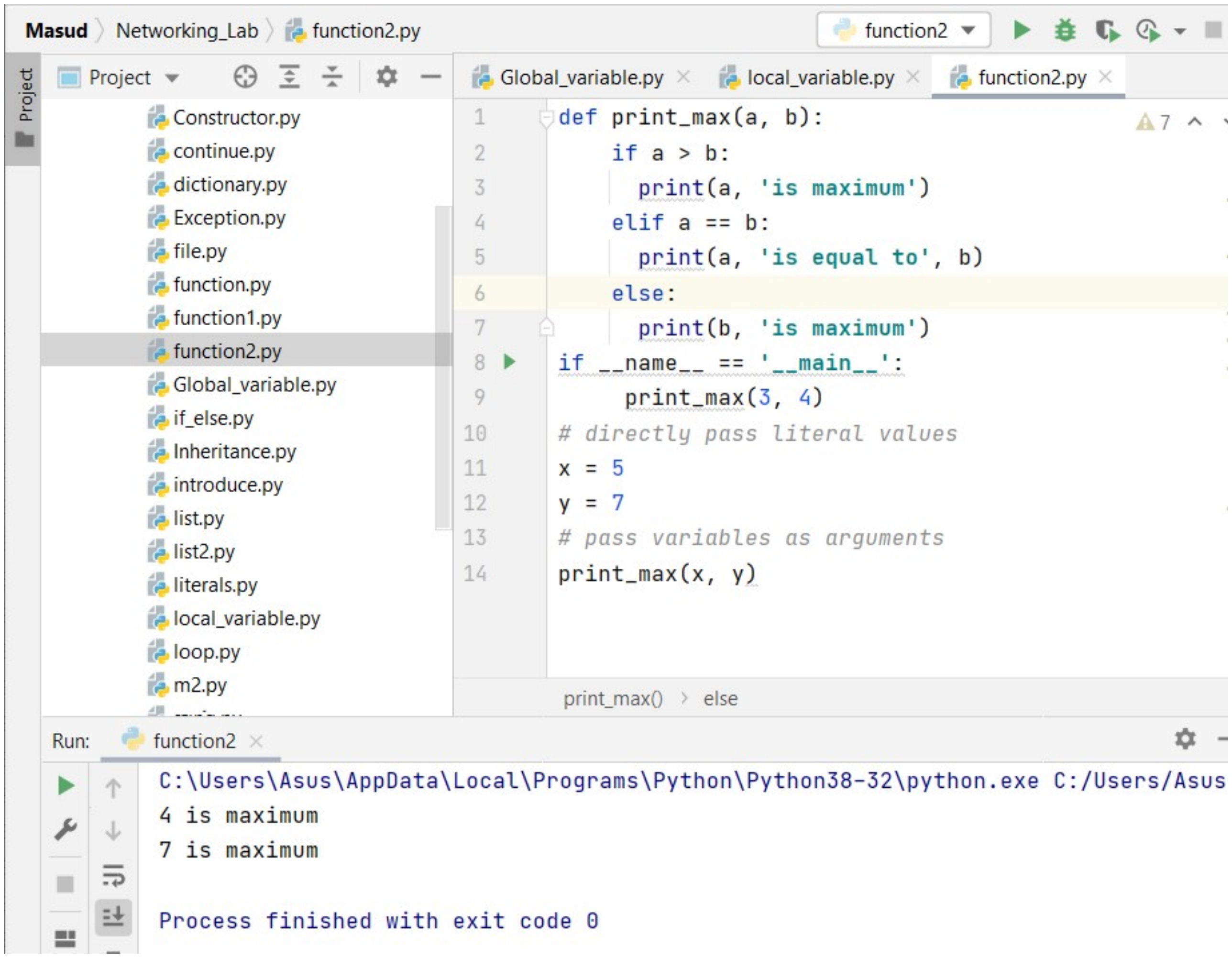
Exercise 4.1.2: Python function (save as function.py)



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

```
C:\Users\Asus\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/Asus/Masud/Networking_Lab/function1.py
hello world

Process finished with exit code 0
```

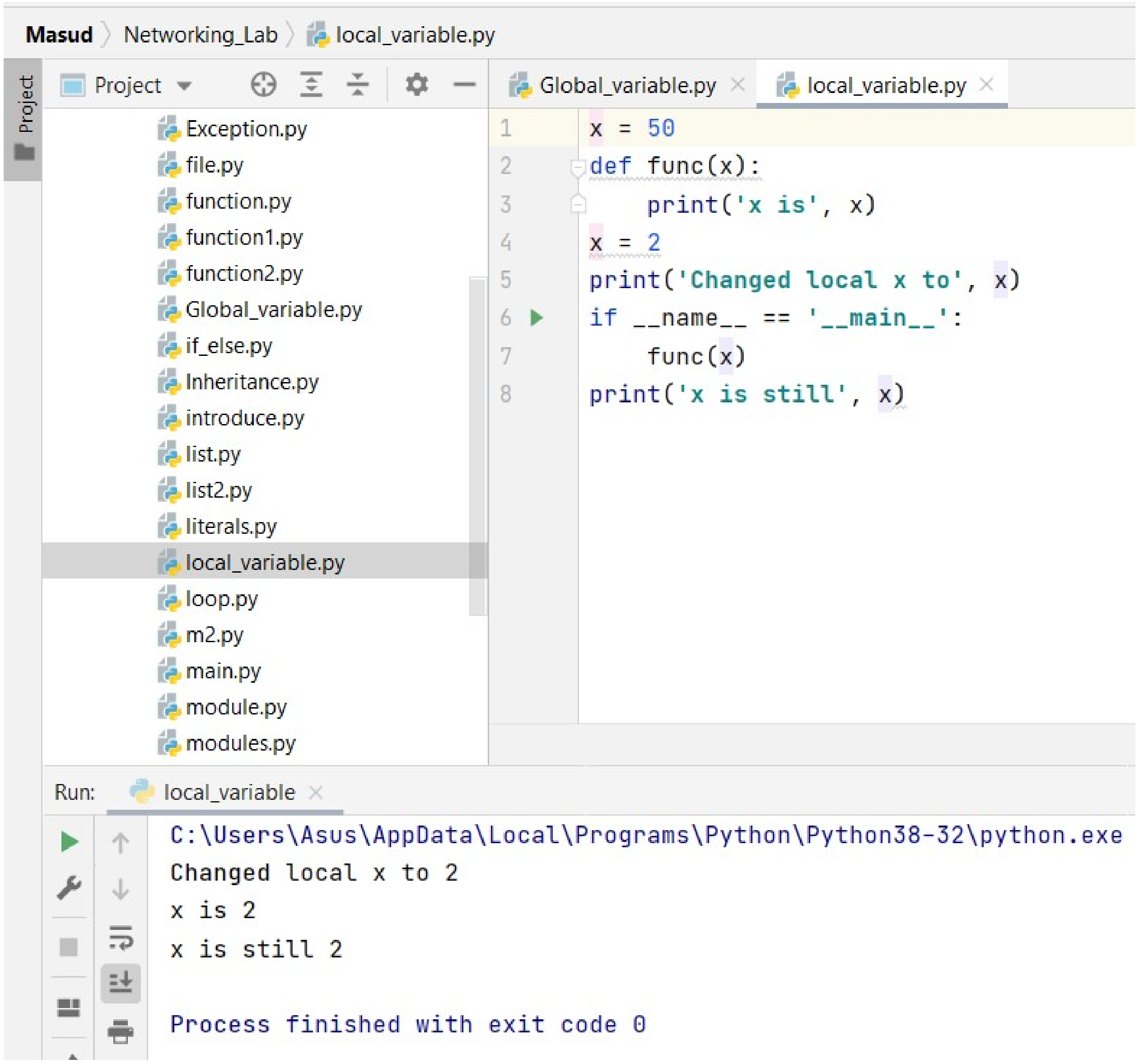
Exercise 4.1.3: Python function (save as function_2.py)



```python
def print_max(a, b):
    if a > b:
      print(a, 'is maximum')
    elif a == b:
      print(a, 'is equal to', b)
    else:
      print(b, 'is maximum')
if __name__ == '__main__':
     print_max(3, 4)
# directly pass literal values
x = 5
y = 7
# pass variables as arguments
print_max(x, y)
```

```
C:\Users\Asus\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/Asus
4 is maximum
7 is maximum

Process finished with exit code 0
```

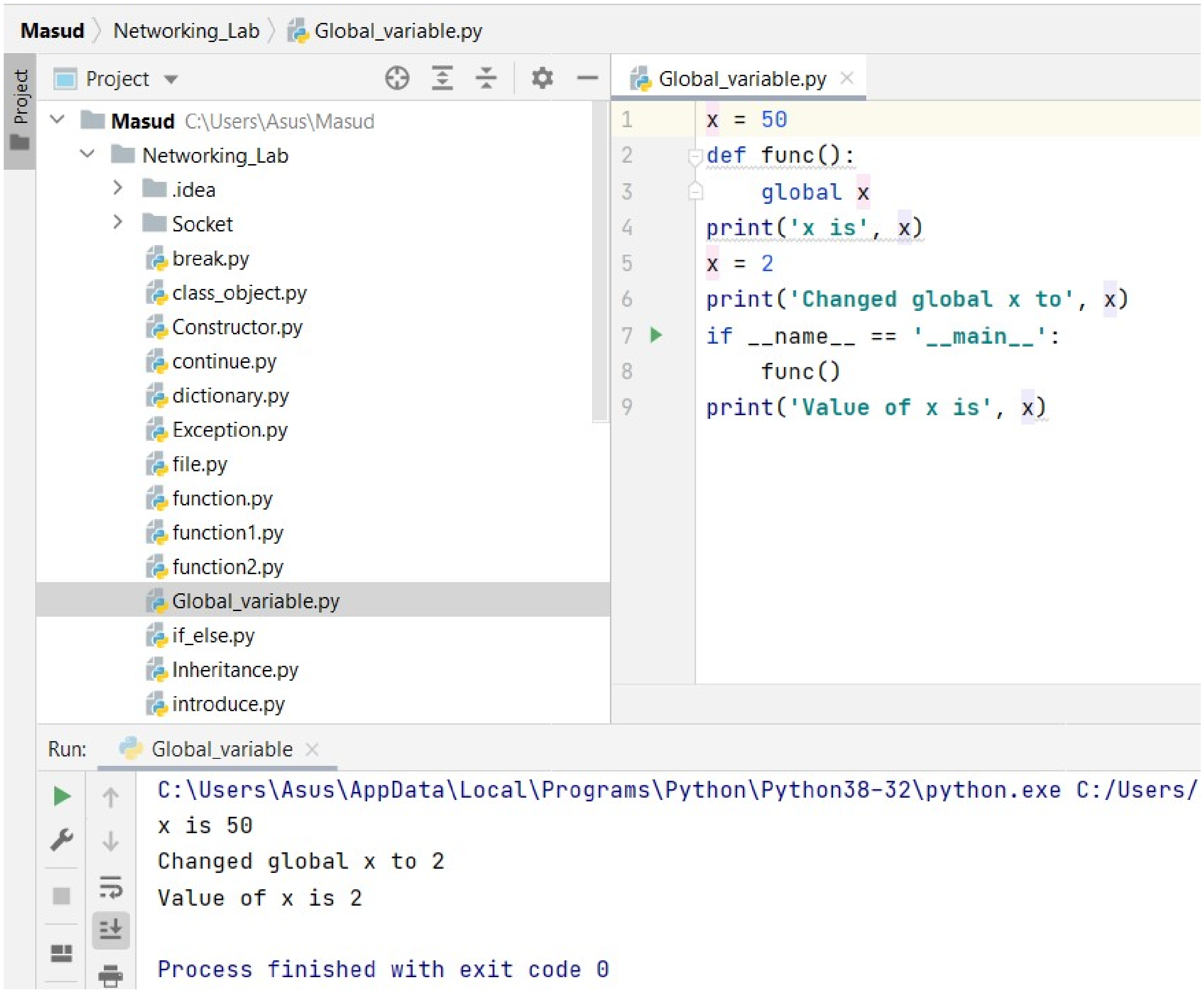
Exercise 4.1.4: Local variable (save as function_local.py)



```python
x = 50
def func(x):
    print('x is', x)
x = 2
print('Changed local x to', x)
if __name__ == '__main__':
    func(x)
print('x is still', x)
```

```
C:\Users\Asus\AppData\Local\Programs\Python\Python38-32\python.exe
Changed local x to 2
x is 2
x is still 2

Process finished with exit code 0
```

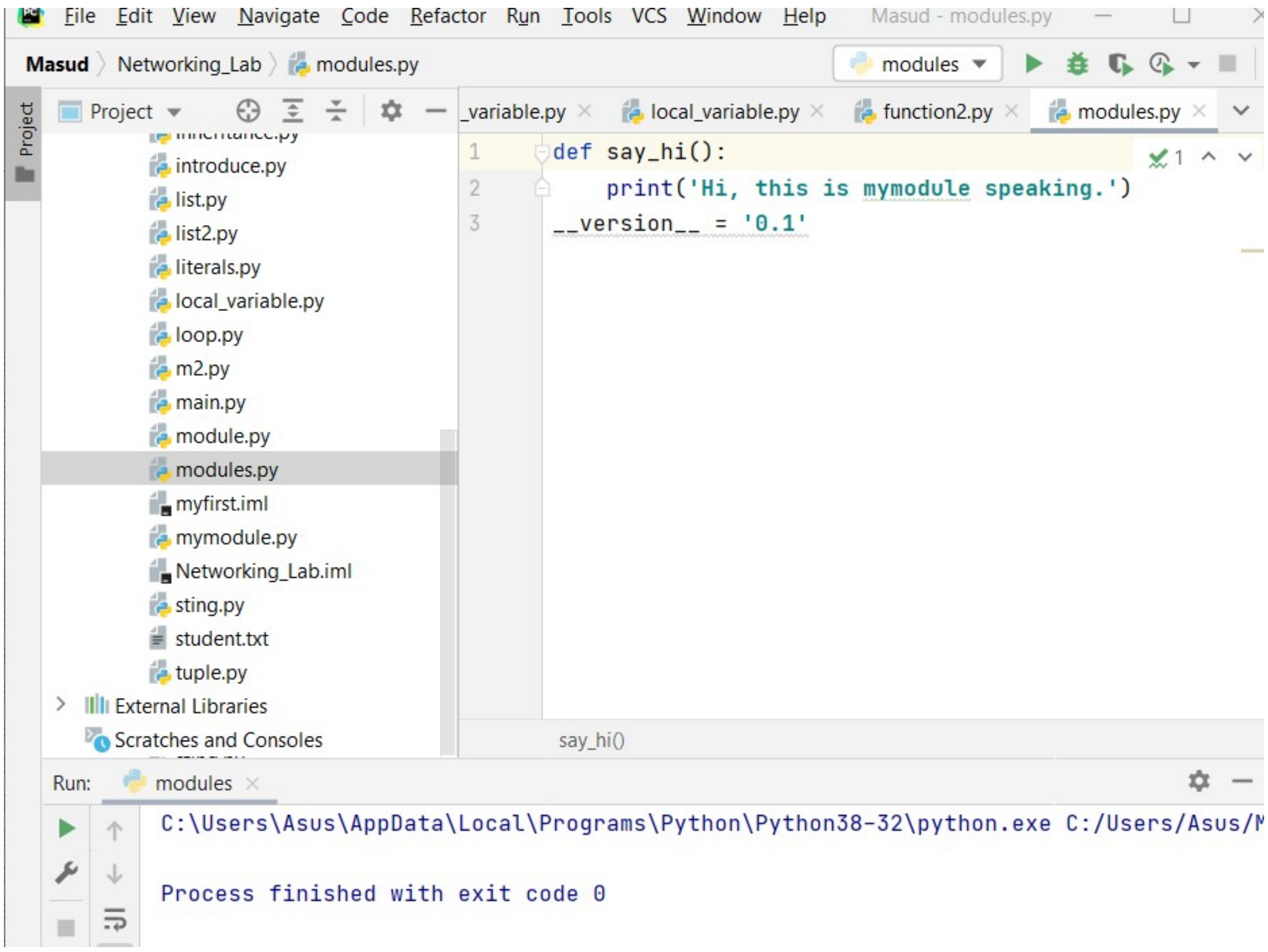
Exercise 4.1.5: Global variable (save as function_global.py)



```python
x = 50
def func():
    global x
print('x is', x)
x = 2
print('Changed global x to', x)
if __name__ == '__main__':
    func()
print('Value of x is', x)
```

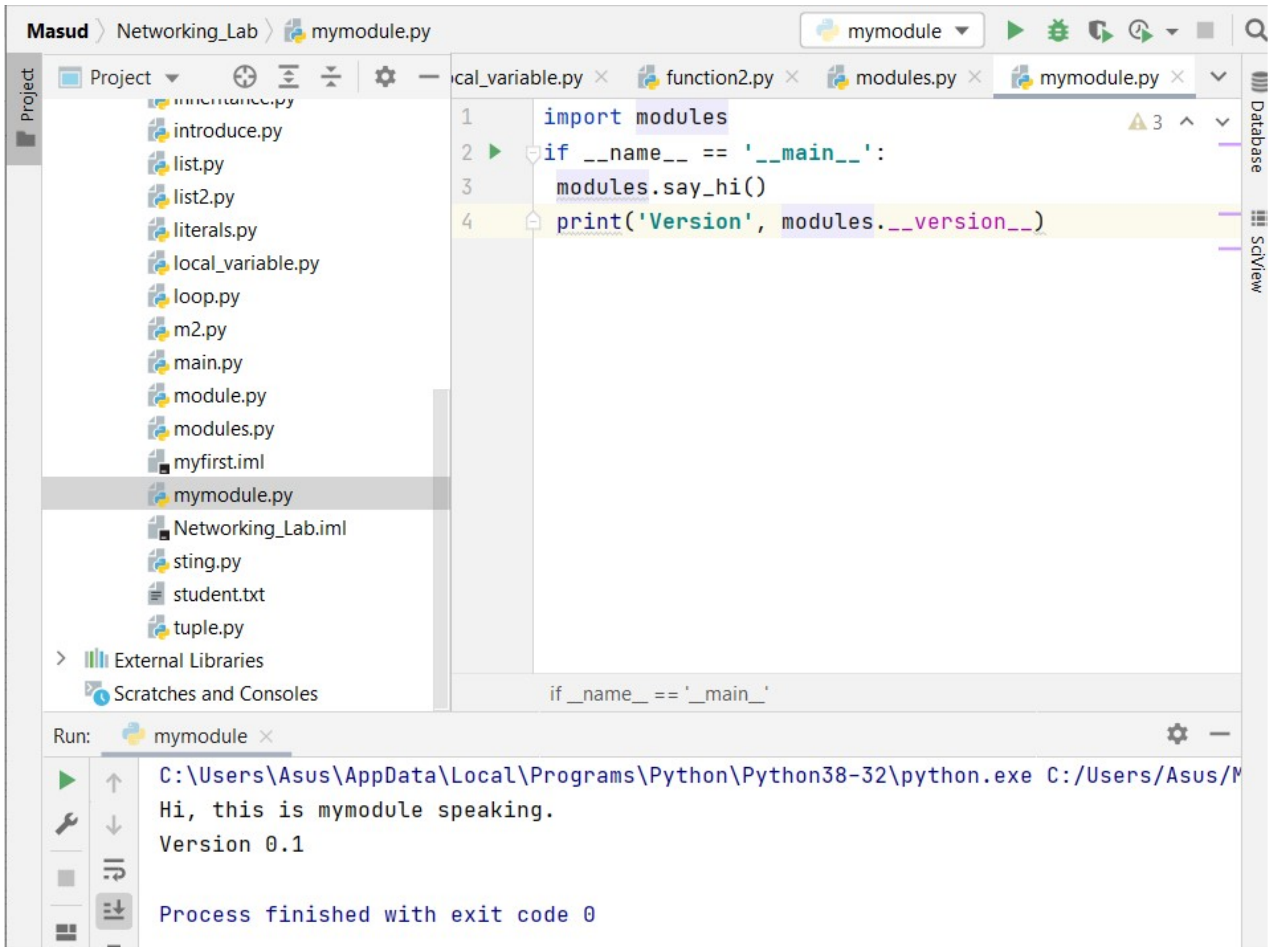
```
C:\Users\Asus\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/
x is 50
Changed global x to 2
Value of x is 2

Process finished with exit code 0
```

Exercise 4.1.6: Python modules



```python
def say_hi():
    print('Hi, this is mymodule speaking.')
__version__ = '0.1'
```

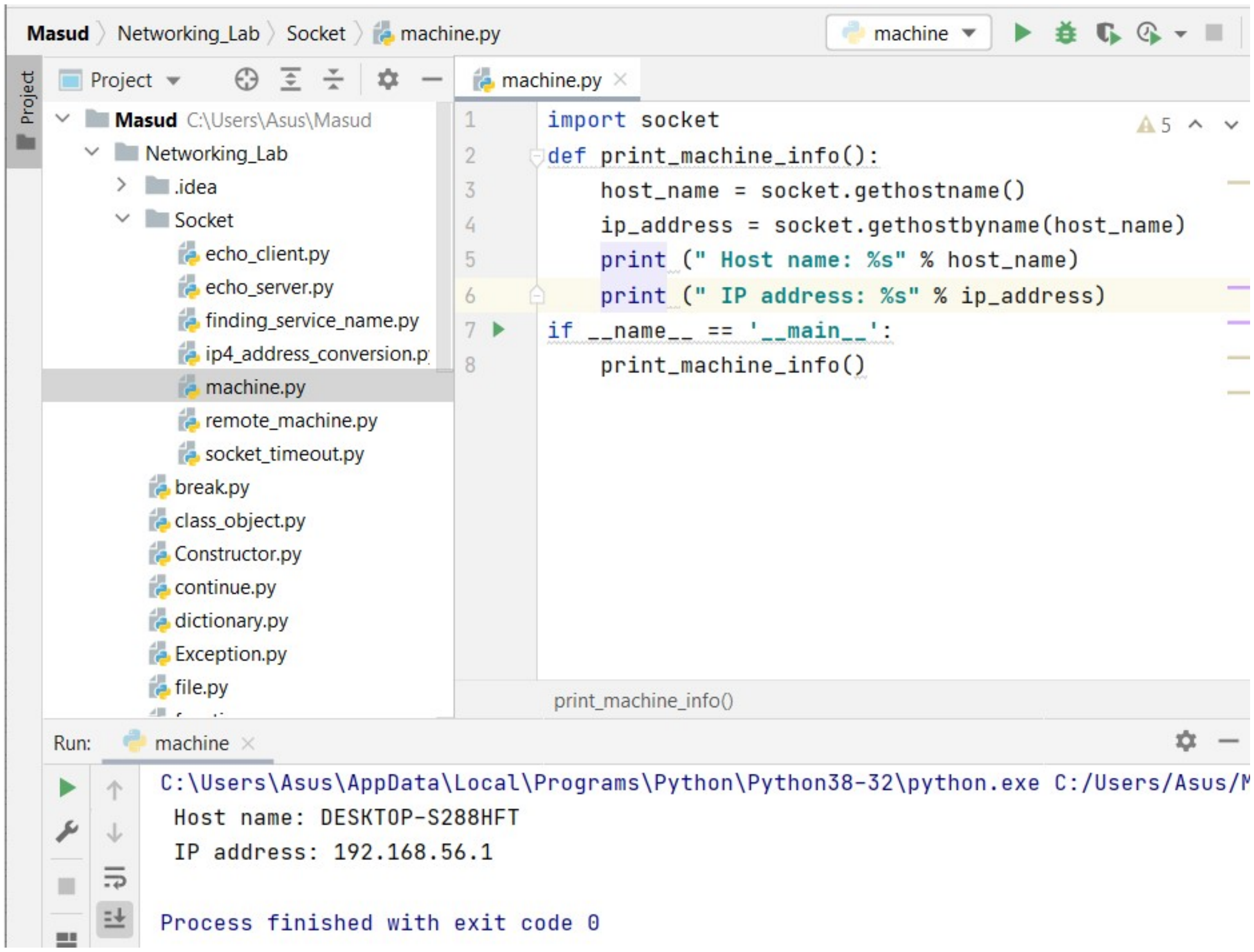
```
C:\Users\Asus\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/Asus/M

Process finished with exit code 0
```


```python
import modules
if __name__ == '__main__':
 modules.say_hi()
 print('Version', modules.__version__)
```

```
C:\Users\Asus\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/Asus/M
Hi, this is mymodule speaking.
Version 0.1

Process finished with exit code 0
```

Section 4.2: Sockets, IPv4, and Simple Client/Server Programming

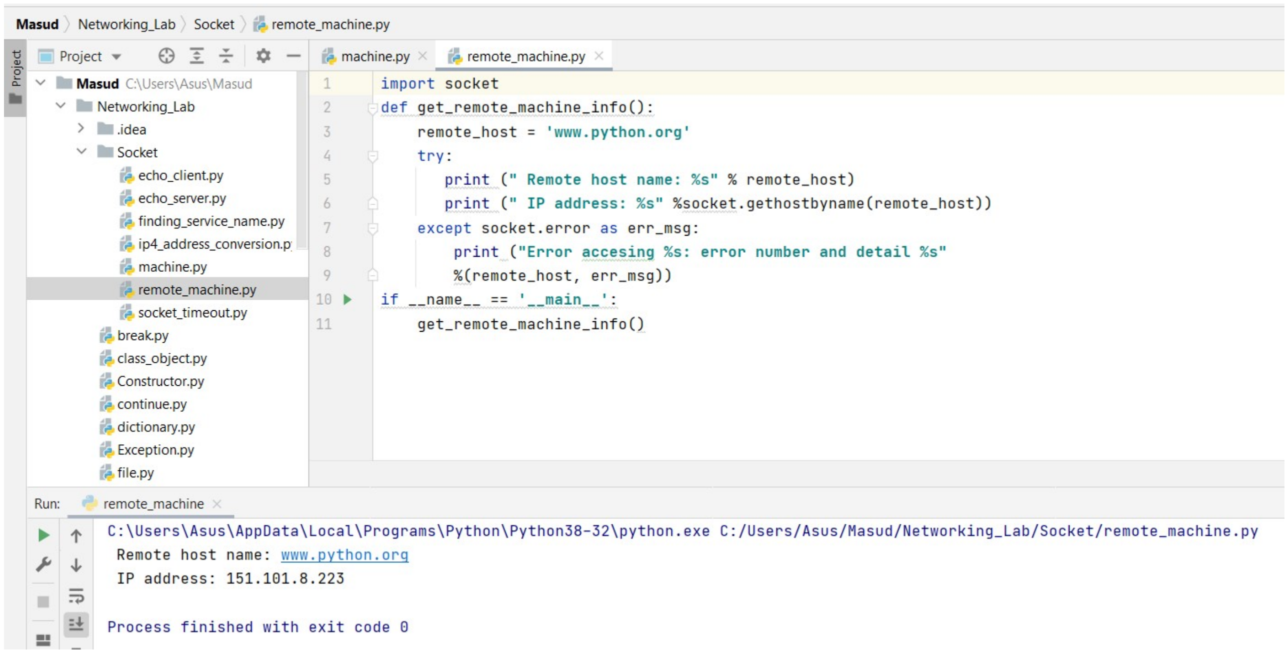
Exercise 4.2.1: Printing your machine's name and IPv4 address



```python
import socket
def print_machine_info():
    host_name = socket.gethostname()
    ip_address = socket.gethostbyname(host_name)
    print (" Host name: %s" % host_name)
    print (" IP address: %s" % ip_address)
if __name__ == '__main__':
    print_machine_info()
```

```
C:\Users\Asus\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/Asus/M
 Host name: DESKTOP-S288HFT
 IP address: 192.168.56.1

Process finished with exit code 0
```

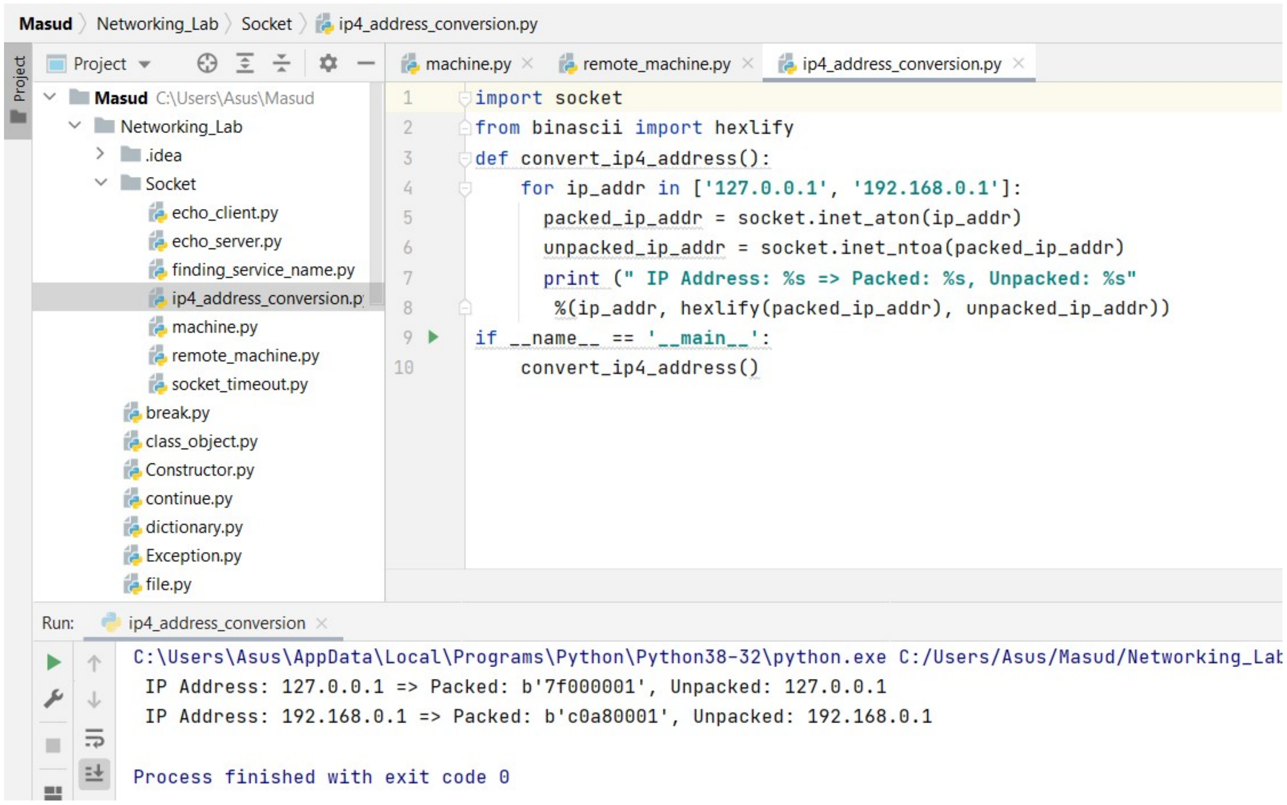
Exercise 4.2.2: Retrieving a remote machine's IP address



```python
import socket
def get_remote_machine_info():
    remote_host = 'www.python.org'
    try:
        print (" Remote host name: %s" % remote_host)
        print (" IP address: %s" %socket.gethostbyname(remote_host))
    except socket.error as err_msg:
         print ("Error accesing %s: error number and detail %s"
         %(remote_host, err_msg))
if __name__ == '__main__':
    get_remote_machine_info()
```

```
C:\Users\Asus\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/Asus/Masud/Networking_Lab/Socket/remote_machine.py
 Remote host name: www.python.org
 IP address: 151.101.8.223

Process finished with exit code 0
```

Exercise 4.2.3: Converting an IPv4 address to different formats



```python
import socket
from binascii import hexlify
def convert_ip4_address():
    for ip_addr in ['127.0.0.1', '192.168.0.1']:
      packed_ip_addr = socket.inet_aton(ip_addr)
      unpacked_ip_addr = socket.inet_ntoa(packed_ip_addr)
      print (" IP Address: %s => Packed: %s, Unpacked: %s"
       %(ip_addr, hexlify(packed_ip_addr), unpacked_ip_addr))
if __name__ == '__main__':
    convert_ip4_address()
```

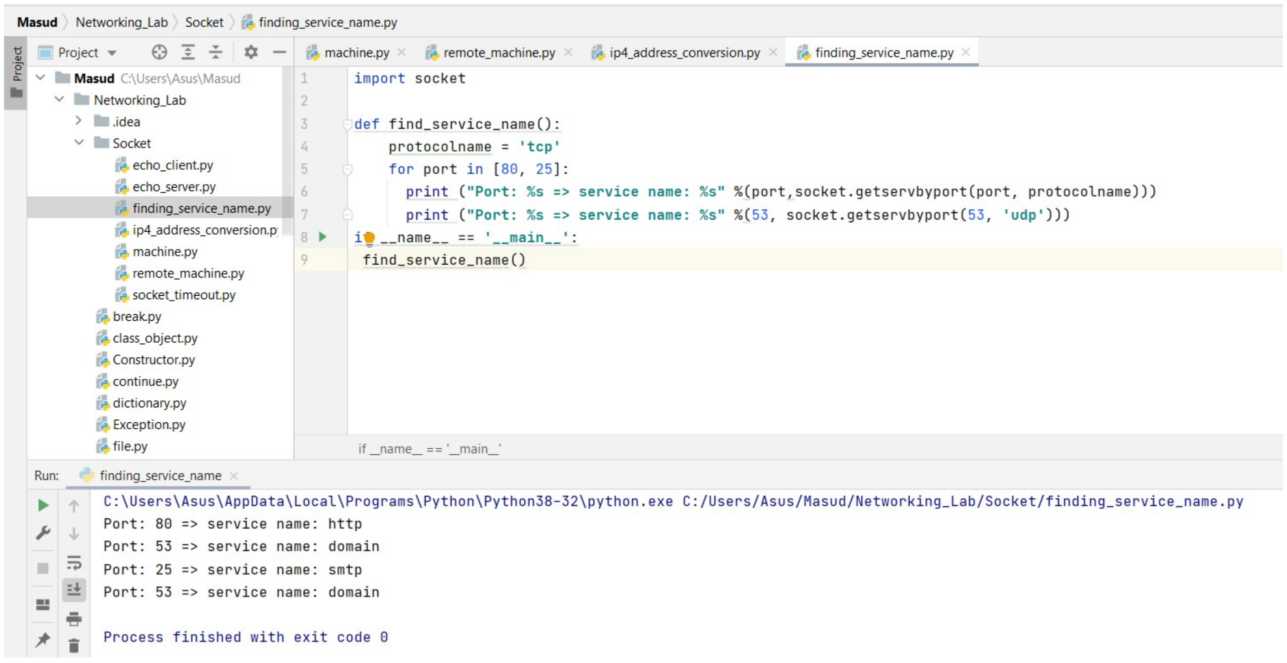
```
C:\Users\Asus\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/Asus/Masud/Networking_Lab
 IP Address: 127.0.0.1 => Packed: b'7f000001', Unpacked: 127.0.0.1
 IP Address: 192.168.0.1 => Packed: b'c0a80001', Unpacked: 192.168.0.1

Process finished with exit code 0
```

How binascii works?

Answer:

The binascii module contains a number of methods to convert between binary and various ASCII-encoded binary representations. Normally, you will not use these functions directly but use wrapper modules like uu, base64, or binhex instead. The binascii module contains low-level functions written in C for greater speed that are used by the higher-level modules.

Exercise 4.2.4: Finding a service name, given the port and protocol



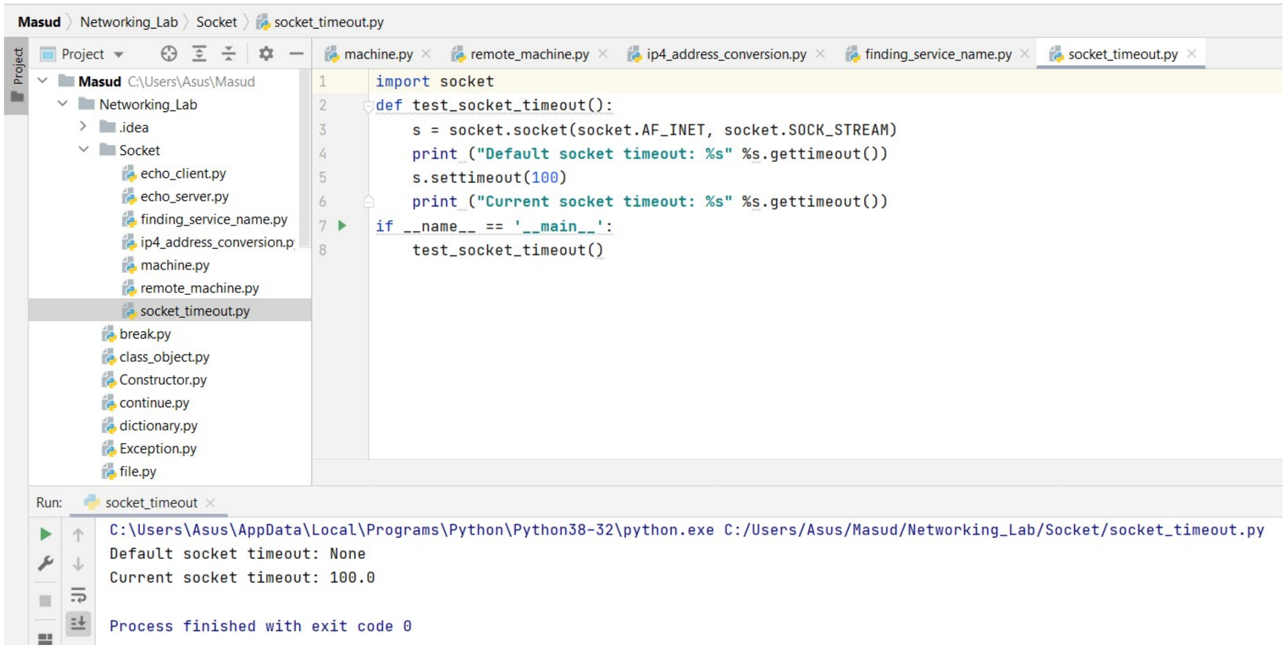
```python
import socket

def find_service_name():
    protocolname = 'tcp'
    for port in [80, 25]:
      print ("Port: %s => service name: %s" %(port,socket.getservbyport(port, protocolname)))
      print ("Port: %s => service name: %s" %(53, socket.getservbyport(53, 'udp')))
if __name__ == '__main__':
 find_service_name()
```

```
C:\Users\Asus\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/Asus/Masud/Networking_Lab/Socket/finding_service_name.py
Port: 80 => service name: http
Port: 53 => service name: domain
Port: 25 => service name: smtp
Port: 53 => service name: domain

Process finished with exit code 0
```

Answer:

Port	Protocol Name
21	ftp
22	ssh
110	pop3

Exercise 4.2.5: Setting and getting the default socket timeout



```python
import socket
def test_socket_timeout():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print ("Default socket timeout: %s" %s.gettimeout())
    s.settimeout(100)
    print ("Current socket timeout: %s" %s.gettimeout())
if __name__ == '__main__':
    test_socket_timeout()
```

```
C:\Users\Asus\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/Asus/Masud/Networking_Lab/Socket/socket_timeout.py
Default socket timeout: None
Current socket timeout: 100.0

Process finished with exit code 0
```

Which is the role of socket timeout in real applications?

Answer:

In complex networks and in consumer computers, there is a digital component called a socket that connects two different platforms. When there is a problem with the socket connection, such as the network being unavailable or there being no Internet, the socket will keep trying to connect. A socket timeout stops this connection after a specified amount of time. The socket timeout command is usually created in object-oriented programming (OOP) or network programming, and keeps the socket from creating inflated problems by severing the connection.

A socket timeout is a designated amount of time from when the socket connects until the connection breaks. Many users believe the timeout itself is a problem, but the timeout is actually made to keep further problems from manifesting. The amount of time between the connection and the timeout is set by programmers of the software or **operating system** (OS). Without a timeout command, the socket will continue to attempt the connection indefinitely.

If the socket timeout is not programmed, then the socket will remain open as it waits for the other side to connect. Allowing it to remain open opens the computer up to potential malicious attacks; more commonly, the computer just uses excess memory to connect to a network that is not responding. This also keeps the socket from being used for anything else, which makes the entire computer slow down.

OS and software programmers have to specify the socket timeout wait time. This is most commonly seen in OOP or network programming, because these are the programs that use sockets the most; most website programming does not use sockets as often and has no timeout commands. The timeout amount is generally measured in milliseconds, but the programmer can make the timeout take several minutes or even hours if he or she desires.

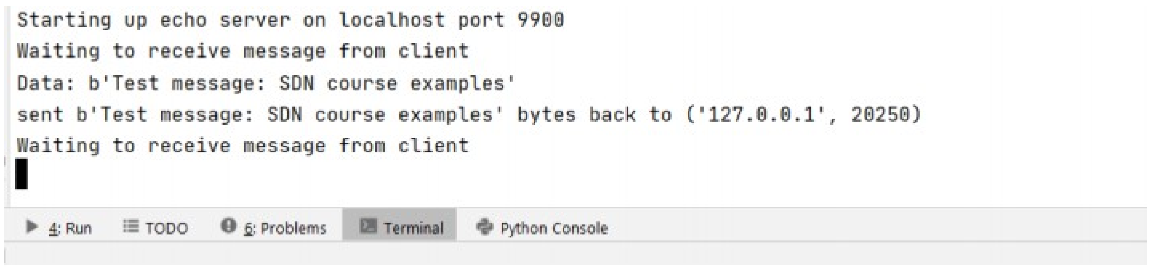
Most programmers have two socket timeout messages, one for a connection that is not responding and another for when the server or network program is closed. A socket timeout is not always needed for a socket to stop the connection. When a server or computer is about to close the connection, it sends a signal to the socket to do the same and close the connection between the two systems. This signal is not always received, including when the Internet suddenly crashes or the **Ethernet** cable is removed during the connection time. In these instances, the socket will just keep waiting for data.

Exercise 4.2.6: Writing a simple echo client/server application (Tip: Use port 9900) Run the script, which is the output?

Answer:

Create python script using the syntax below (save as echo_server.py):

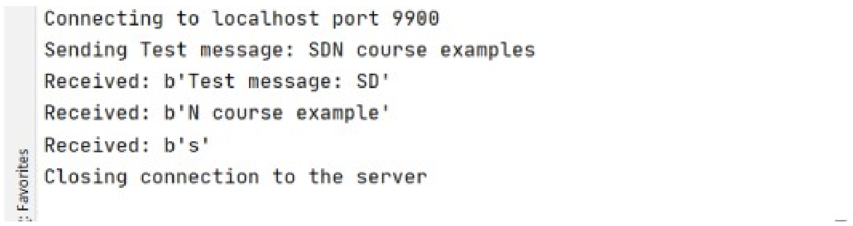
Output:



```
Starting up echo server on localhost port 9900
Waiting to receive message from client
Data: b'Test message: SDN course examples'
sent b'Test message: SDN course examples' bytes back to ('127.0.0.1', 20250)
Waiting to receive message from client
```

Create python script using the syntax below (save as echo_client.py):

Output:



```
Connecting to localhost port 9900
Sending Test message: SDN course examples
Received: b'Test message: SD'
Received: b'N course example'
Received: b's'
Closing connection to the server
```

What you need to do for running the program?

Answer: I did following steps :

1. First I started the server at 9900 port

   echo_server.py --port 9900

2. Secondly I sent the request to the server from the client.

   echo_client.py --port 9900

Which program do you need to run the first client of the server?

Answer: I ran first server program to start the server.

5. Questions

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

Answer:
Python Function : A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

Python Module : A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.

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

Answer:
Local variable is created when the function starts execution and lost when the functions terminate.Global variable is created before the program's global execution starts and lost when the program terminates.

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

Answer:
A socket is one endpoint of a two way communication link between two programs running on the network. The socket mechanism provides a means of inter-process communication (IPC) by establishing named contact points between which the communication takes place.Like ‘Pipe’ is used to create pipes and sockets is created using ‘socket’ system call. The socket provides bidirectional FIFO Communication facility over the network. A socket connecting to the network is created at each end of the communication. Each socket has a specific address. This address is composed of an IP address and a port number.Socket are generally employed in client server applications. The server creates a socket, attaches it to a network port addresses then waits for the client to contact it. The client creates a socket and then attempts to connect to the server socket. When the connection is established, transfer of data takes place.

Several types of Internet socket are available:

Datagram sockets

Connectionless sockets, which use User Datagram Protocol (UDP).[3] Each packet sent or received on a datagram socket is individually addressed and routed. Order and reliability are not guaranteed with datagram sockets, so multiple packets sent from one machine or process to another may arrive in any order or might not arrive at all. Special configuration may be required to send broadcasts on a datagram socket.[4] In order to receive broadcast packets, a datagram socket should not be bound to a specific address, though in some implementations, broadcast packets may also be received when a datagram socket is bound to a specific address.[5]

Stream sockets

Connection-oriented sockets, which use Transmission Control Protocol (TCP), Stream Control Transmission Protocol (SCTP) or Datagram Congestion Control Protocol (DCCP). A stream socket provides a sequenced and unique flow of error-free data without record boundaries, with well-defined mechanisms for creating and destroying connections and reporting errors. A stream socket transmits data reliably, in order, and with out-of-band capabilities. On the Internet, stream sockets are typically implemented using TCP so that applications can run across any networks using TCP/IP protocol.

Raw sockets

Allow direct sending and receiving of IP packets without any protocol-specific transport layer formatting. With other types of sockets, the payload is automatically encapsulated according to the chosen transport layer protocol (e.g. TCP, UDP), and the socket user is unaware of the existence of protocol headers that are broadcast with the payload. When reading from a raw socket, the headers are usually included. When transmitting packets from a raw socket, the automatic addition of a header is optional.

Most socket application programming interfaces (APIs), for example those based on Berkeley sockets, support raw sockets. Windows XP was released in 2001 with raw socket support implemented in the Winsock interface, but three years later, Microsoft limited Winsock's raw socket support because of security concerns.[6]

Raw sockets are used in security-related applications like Nmap. One use case for raw sockets is the implementation of new transport-layer protocols in user space.[7] Raw sockets are typically available in network equipment, and used for routing protocols such as the Internet Group Management Protocol (IGMP) and Open Shortest Path First (OSPF), and in the Internet Control Message Protocol (ICMP) used, among other things, by the ping utility.[8]

Other socket types are implemented over other transport protocols, such as Systems Network Architecture[9] and Unix domain sockets for internal inter-process communication.

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

Answer:

The IPv4 address is a 32-bit number that uniquely identifies a network interface on a machine. An IPv4 address is typically written in decimal digits, formatted as four 8-bit fields that are separated by periods. Each 8-bit field represents a byte of the IPv4 address. This form of representing the bytes of an IPv4 address is often referred to as the dotted-decimal format.

When a user opens a web browser and types www.tutorialpoints.com which is a domain name and a PC does not understand how to communicate with the server using domain names, then the PC sends a DNS query out on the network in order to obtain the IP address pertaining to the domain name. The pre-configured DNS server responds to the query with the IP address of the domain name specified.