Mawlana Bhashani Science and Technology University

Lab Report

Lab Report No: 02

Course code: ICT-3208

Course title: Computer Network

Date of Performance:8-01-2021

Date of Submission:16-01-2021

Submitted by

Name: Md Sohanur Islam

ID: IT-18011

3rd year 2nd semester

Session: 2017-2018

Dept. of ICT

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Python functions:

A function is a block of code which only runs when it is called .You can pass data, known as parameters, into a function .A function can return data as a result.

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:

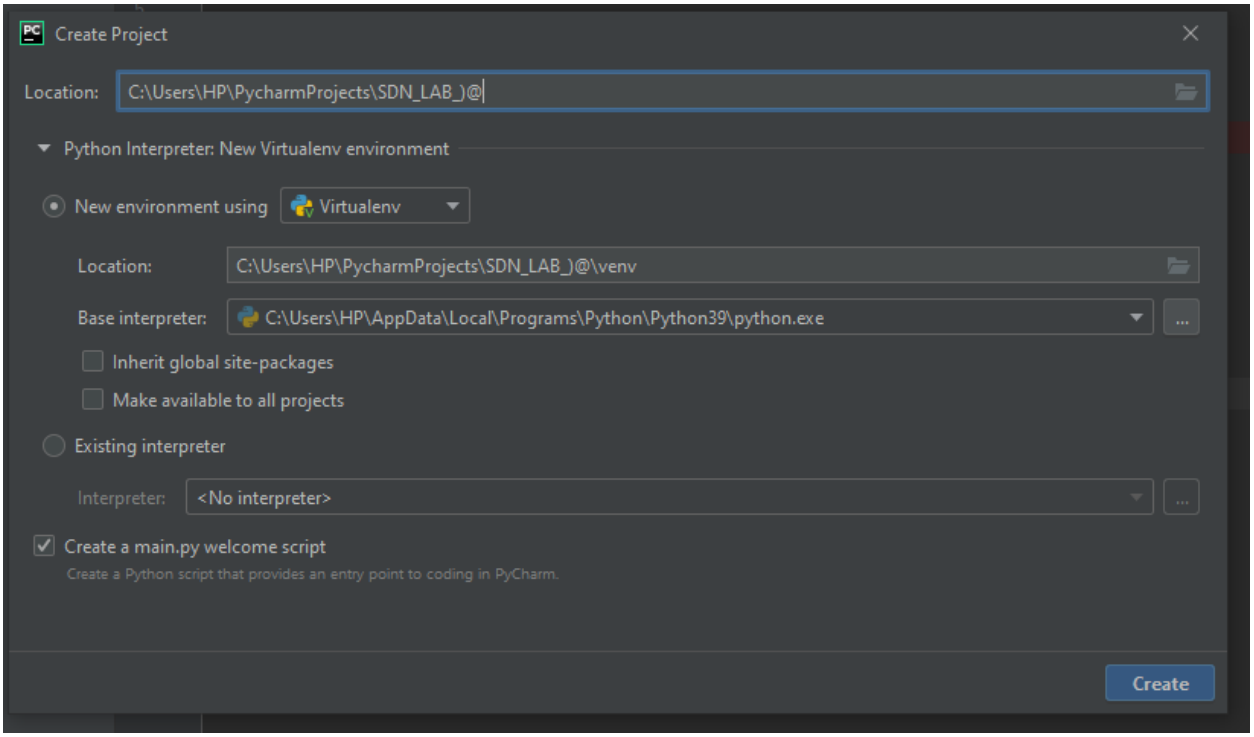
Variables declared inside a function definition are not related in any way to other variables with the same names used outside the function (variable names are local to the function). This is called the scope of the variable. All variables have the scope of the block they are declared in starting from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

The global statement:

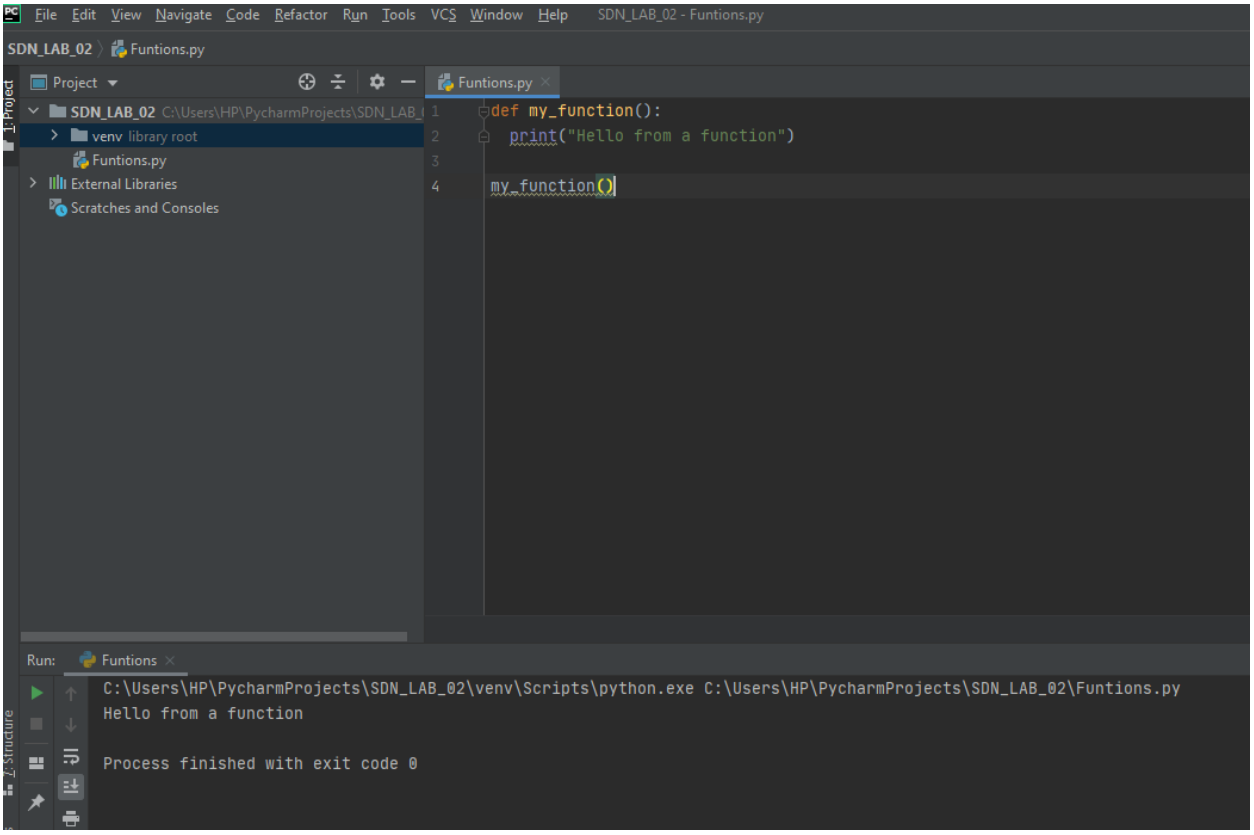
Variables defined at the top level of the program are intended global. Global variables are intended to be used in any functions or classes). Global statement allows defining global variables inside functions as well. Modules: Modules allow reusing a number of functions in other programs.

Exercises:

Exercise 4.1.1: Create a python project using with SDN_LAB



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



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

File Edit View Navigate Code Refactor Run Tools VCS Window Help SDN_LAB_02 - functions_2.py

SDN_LAB_02 functions_2.py

Project

SDN_LAB_02 C:\Users\HP\PycharmProjects\SDN_LAB_

venv library root

functions_2.py

External Libraries

Scratches and Consoles

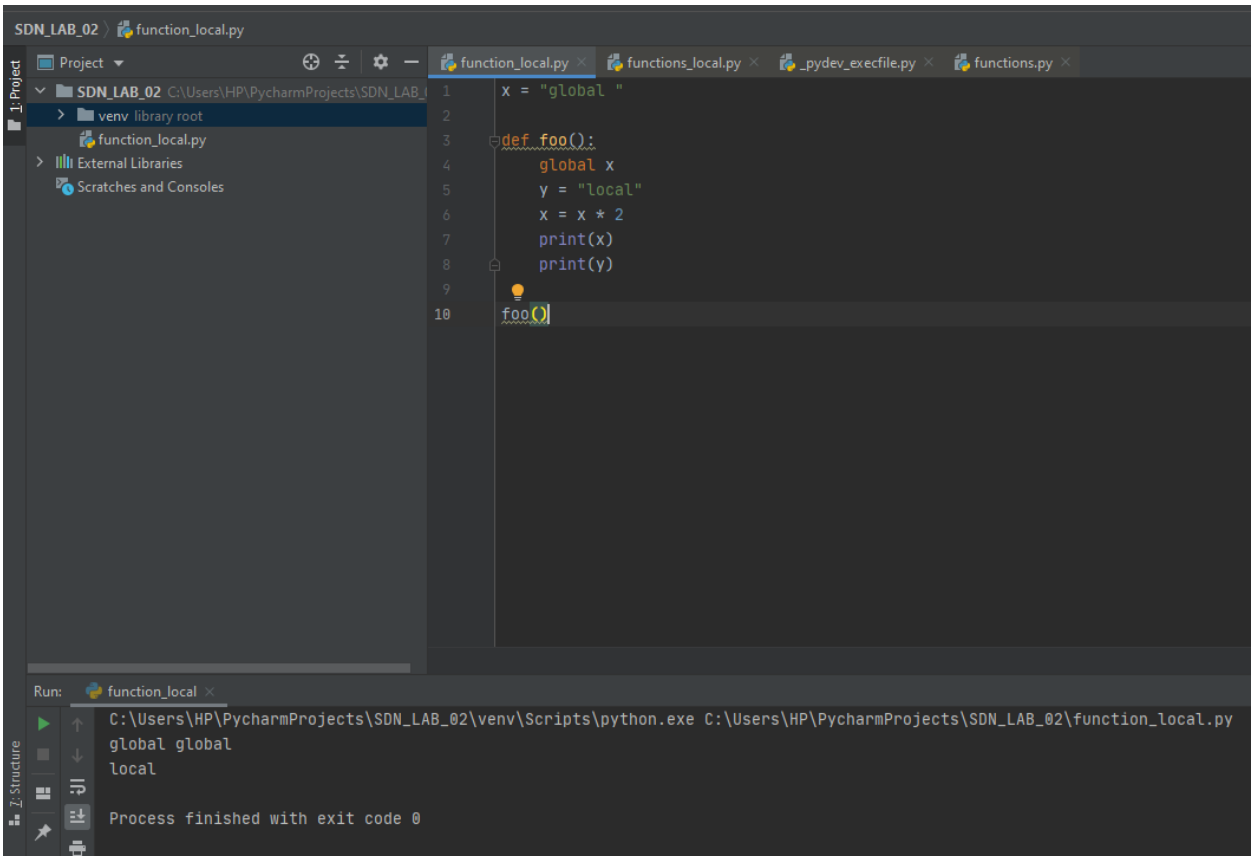
functions_2.py functions.py

```python
# Python program to find the largest
# number among the three numbers
def maximum(a, b, c):
    if (a >= b) and (a >= c):
        largest = a

    elif (b >= a) and (b >= c):
        largest = b
    else:
        largest = c

    return largest

# Driven code
a = 10
b = 14
c = 12
print(maximum(a, b, c))
```

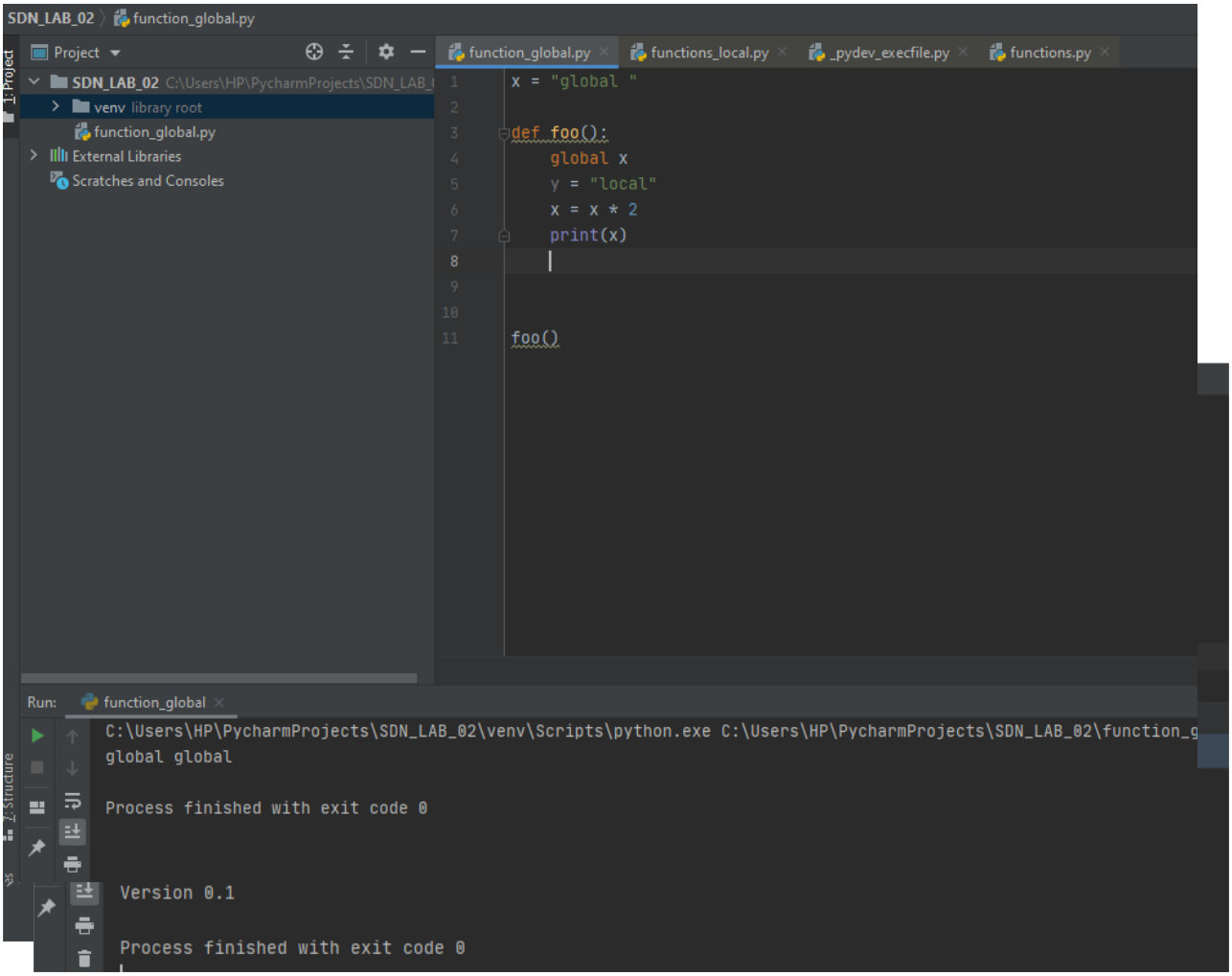
Run: functions_2

```
C:\Users\HP\PycharmProjects\SDN_LAB_02\venv\Scripts\python.exe C:\Users\HP\PycharmProjects\SDN_LAB_02\functions_2.py
14



sohan_IT18011

Process finished with exit code 0
```

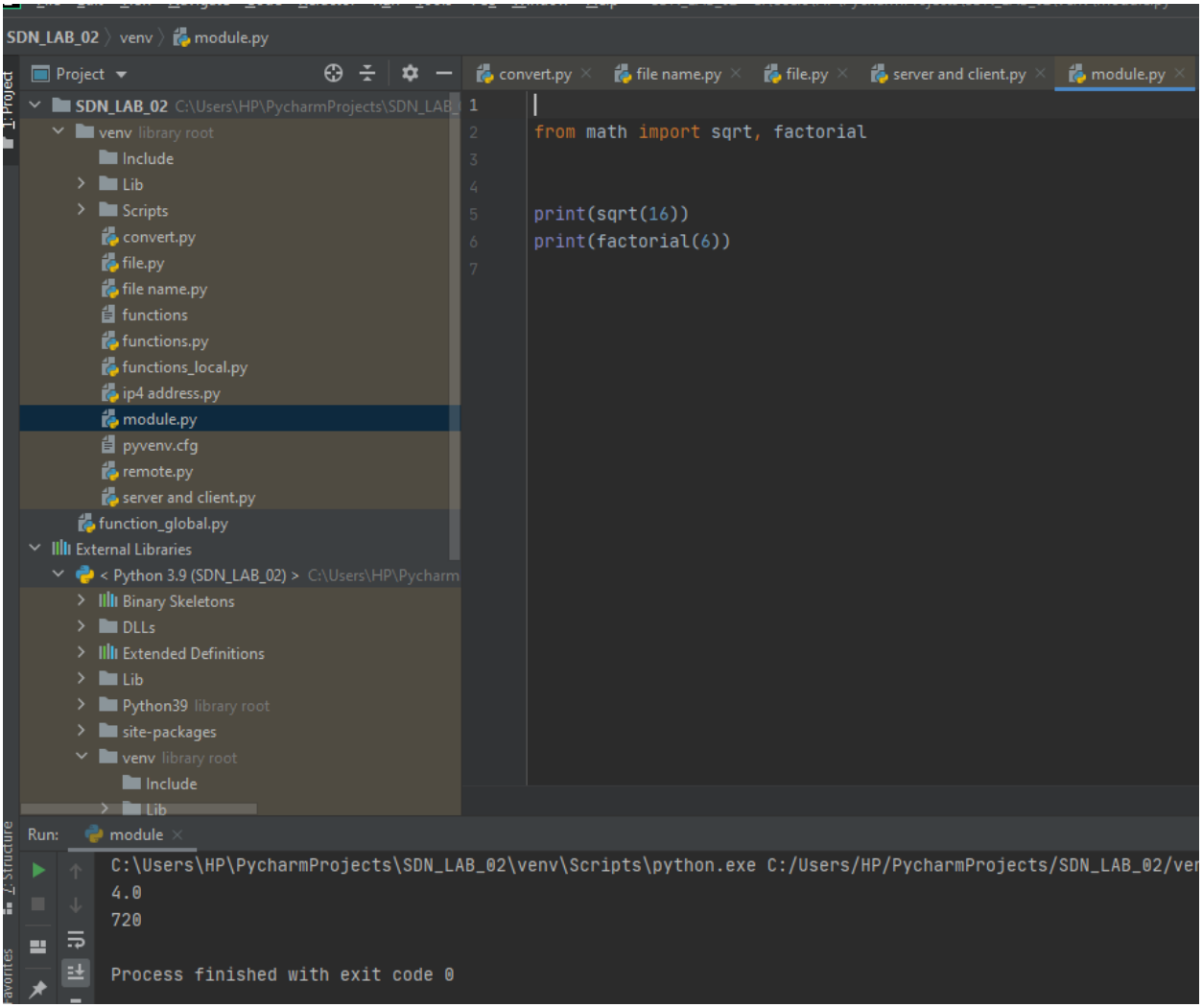
Exercise 4.1.4: Local variable (save as function_local.py)



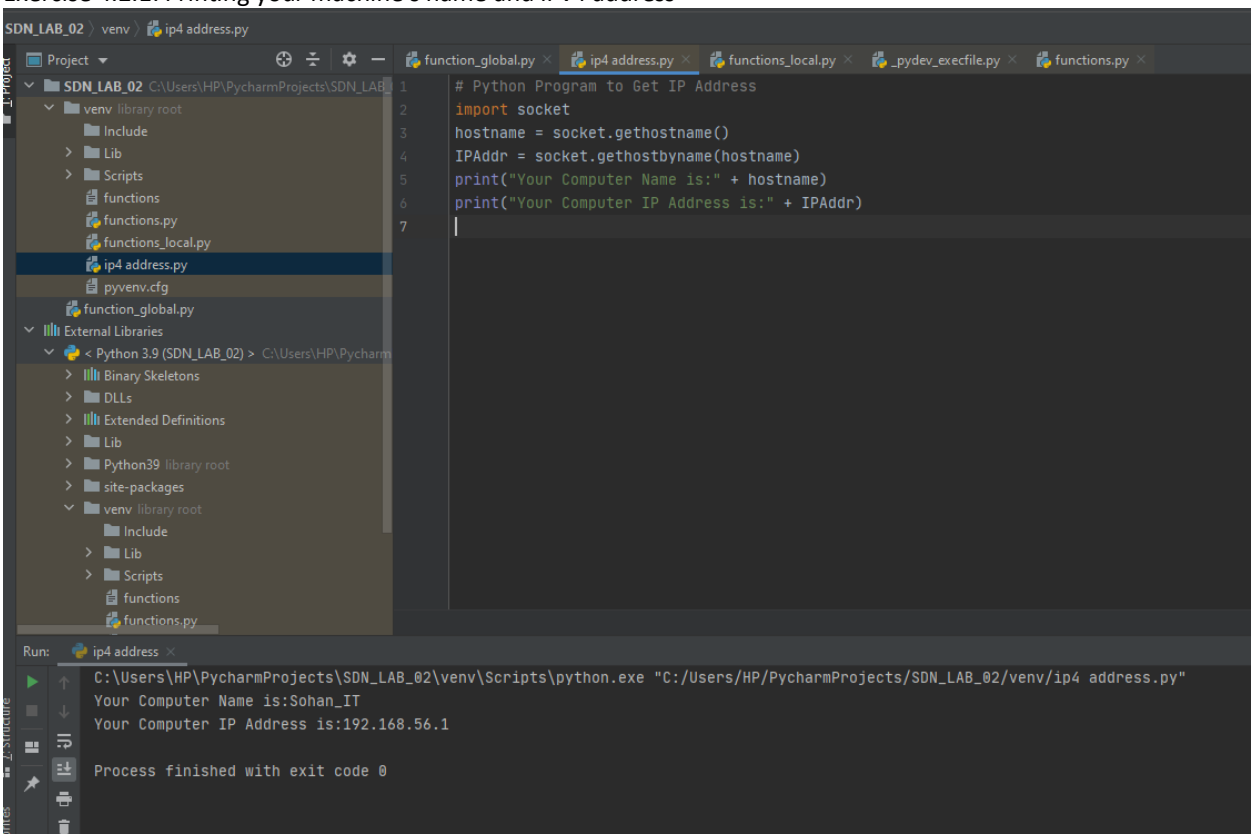
Exercise 4.1.5: Global variable (save as function_global.py)



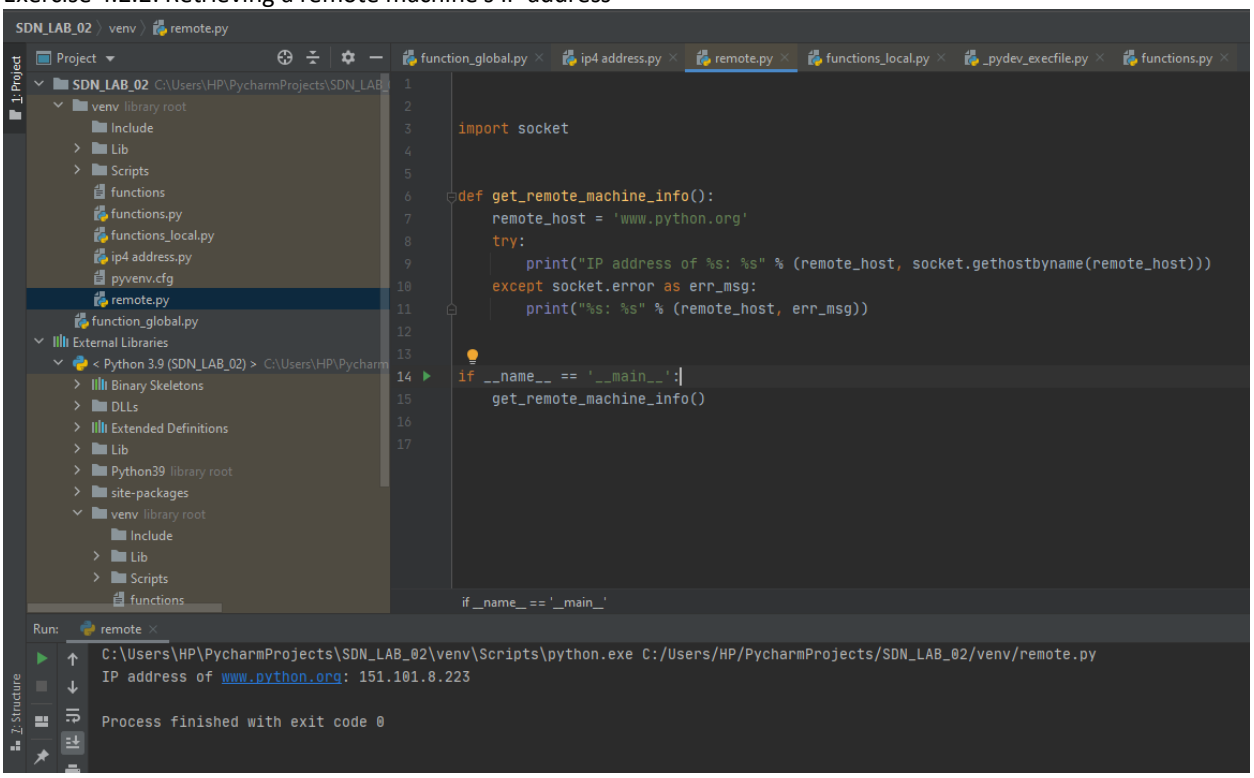
Exercise 4.1.6: Python modules


SDN_LAB_02 venv module.py
Project
convert.py
file name.py
file.py
server and client.py
module.py
SDN_LAB_02 C:\Users\HP\PycharmProjects\SDN_LAB_
venv library root
Include
Lib
Scripts
convert.py
file.py
file name.py
functions
functions.py
functions_local.py
ip4 address.py
module.py
pyvenv.cfg
remote.py
server and client.py
function_global.py
External Libraries
< Python 3.9 (SDN_LAB_02) > C:\Users\HP\Pycharm
Binary Skeletons
DLLs
Extended Definitions
Lib
Python39 library root
site-packages
venv library root
Include
Lib
from math import sqrt, factorial
print(sqrt(16))
print(factorial(6))
Run: module
C:\Users\HP\PycharmProjects\SDN_LAB_02\venv\Scripts\python.exe C:/Users/HP/PycharmProjects/SDN_LAB_02/ver
4.0
720
Process finished with exit code 0

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



Exercise 4.2.2: Retrieving a remote machine's IP address



Exercise 4.2.3: Converting an IPv4 address to different formats

```python
# This program is optimized for Python 2.7.
# It may run on any other version with/without modifications.

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\HP\PycharmProjects\SDN_LAB_02\venv\Scripts\python.exe C:/Users/HP/PycharmProjects/SDN_LAB_02/venv/convert.py
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
```

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\HP\PycharmProjects\SDN_LAB_02\venv\Scripts\python.exe "C:/Users/HP/PycharmProjects/SDN_LAB_02/venv/file name.py"
Port: 80 => service name: http
Port: 25 => service name: smtp
Port: 53 => service name: domain

Process finished with exit code 0
```

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\HP\PycharmProjects\SDN_LAB_02\venv\Scripts\python.exe C:/Users/HP/PycharmProjects/SDN_LAB_02/venv/file.py
Current socket timeout: 100.0

Process finished with exit code 0
```

Exercise 4.2.6: Writing a simple echo client/server application (Tip: Use port 9900) Server Code:

```python
import socket
import sys
import argparse

host = 'localhost'
data_payload = 2048
backlog = 5


def echo_server(port):
    """ A simple echo server """
    # Create a TCP socket
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # Enable reuse address/port
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    # Bind the socket to the port
    server_address = (host, port)

    print("Starting up echo server  on %s port %s" % server_address,sock.bind(server_address))
    # Listen to clients, backlog argument specifies the max no. of queued connections
    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))
        # end connection
        client.close()
```

Conclusion:

Python plays an essential role in network programming. The standard library of Python has full support for network protocols, encoding, and decoding of data and other networking concepts, and it is simpler to write network programs in Python than that of C++. There are two levels of network service access in Python. These are:

Low-Level Access

- High-Level Access

- In the first case, programmers can use and access the basic socket support for the operating system using Python's libraries, and programmers can implement both connection-less and connection-oriented protocols for programming. Application-level network protocols can also be accessed using high-level access provided by Python libraries. These protocols are HTTP, FTP, etc. A socket is the end-point in a flow of communication between two programs or communication channels operating over a network. They are created using a set of programming requests called socket API (Application Programming Interface). Python's socket library offers classes for handling common transports as a generic interface. Sockets use protocols for determining the connection type for port-to-port communication between client and server machines. The protocols are used for:

Domain Name Servers (DNS)

- IP addressing
- E-mail
- FTP (File Transfer Protocol) e•