

# 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

3<sup>rd</sup> year 2<sup>nd</sup> 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:

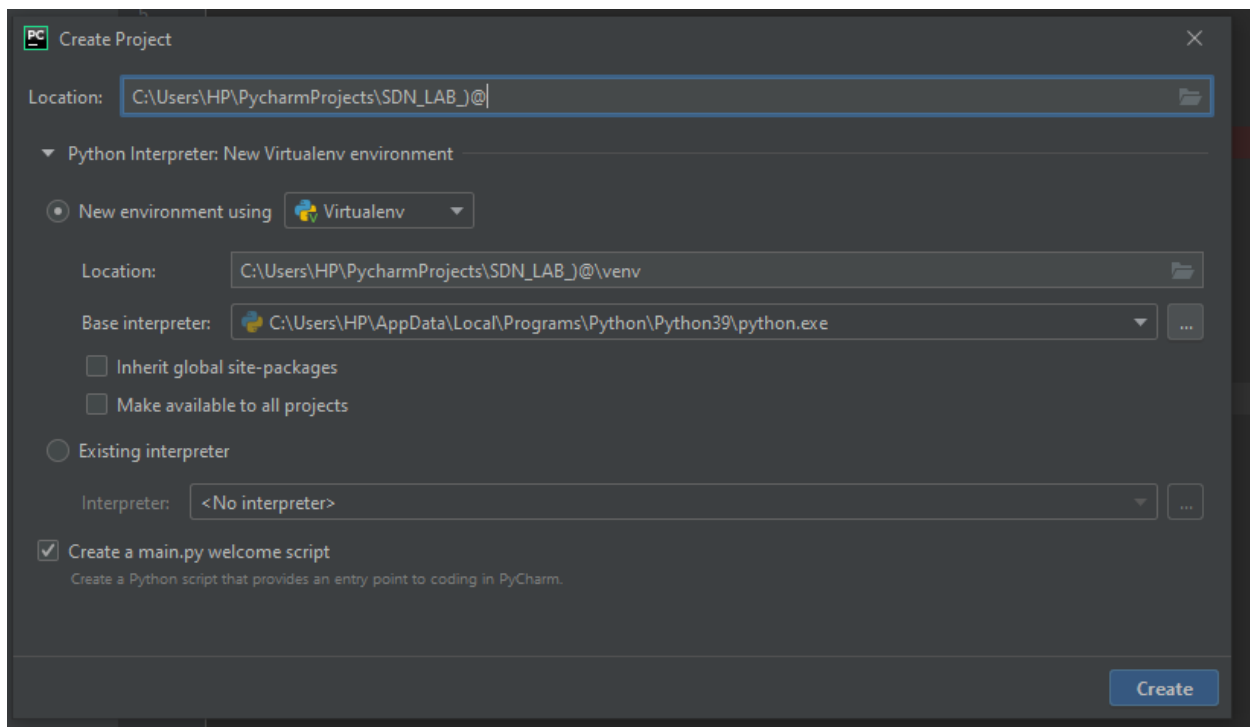
Variables declared inside a function definition are not related in any way to other variables with the same names used outside the function (variable names are local to the function). This is called the scope of the variable. All variables have the scope of the block they are declared in starting from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

## The global statement:

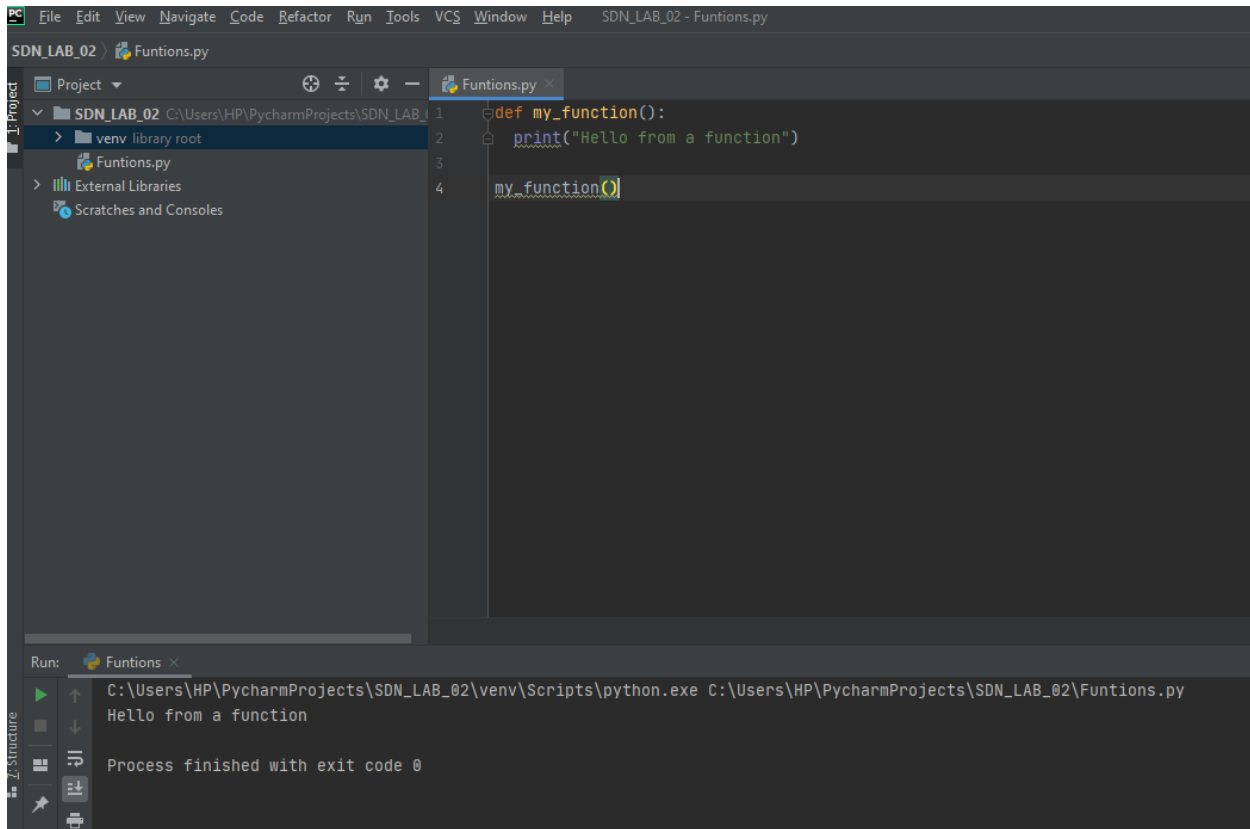
Variables defined at the top level of the program are intended global. Global variables are intended to be used in any functions or classes). Global statement allows defining global variables inside functions as well. Modules: Modules allow reusing a number of functions in other programs.

## Exercises:

Exercise 4.1.1: Create a python project using with SDN\_LAB



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



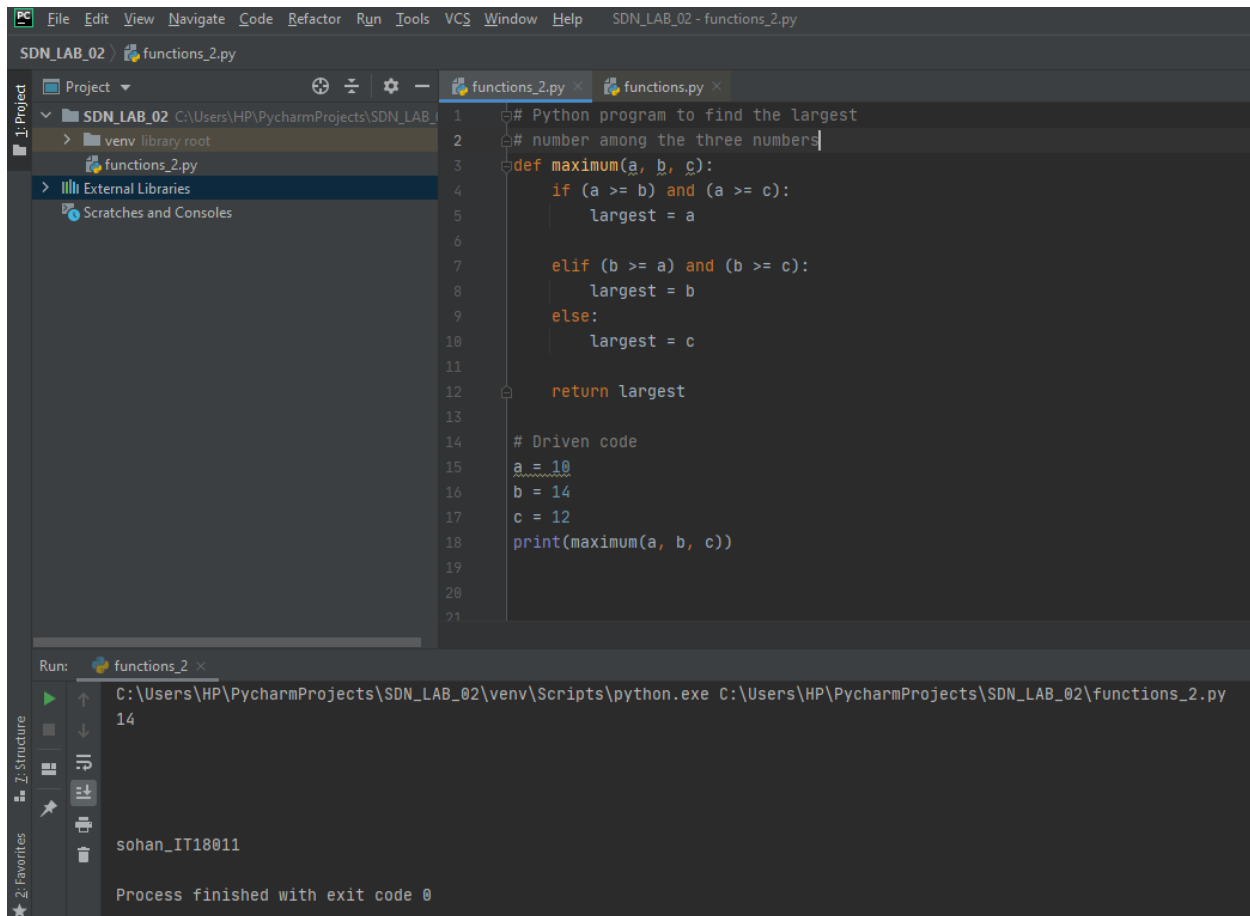
The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The title bar indicates the project is 'SDN\_LAB\_02 - Funtions.py'. The left sidebar shows the project structure with 'SDN\_LAB\_02' containing 'venv library root', 'Funtions.py', 'External Libraries', and 'Scratches and Consoles'. The main editor window displays the code in 'Funtions.py':

```
1 def my_function():  
2     print("Hello from a function")  
3  
4 my_function()
```

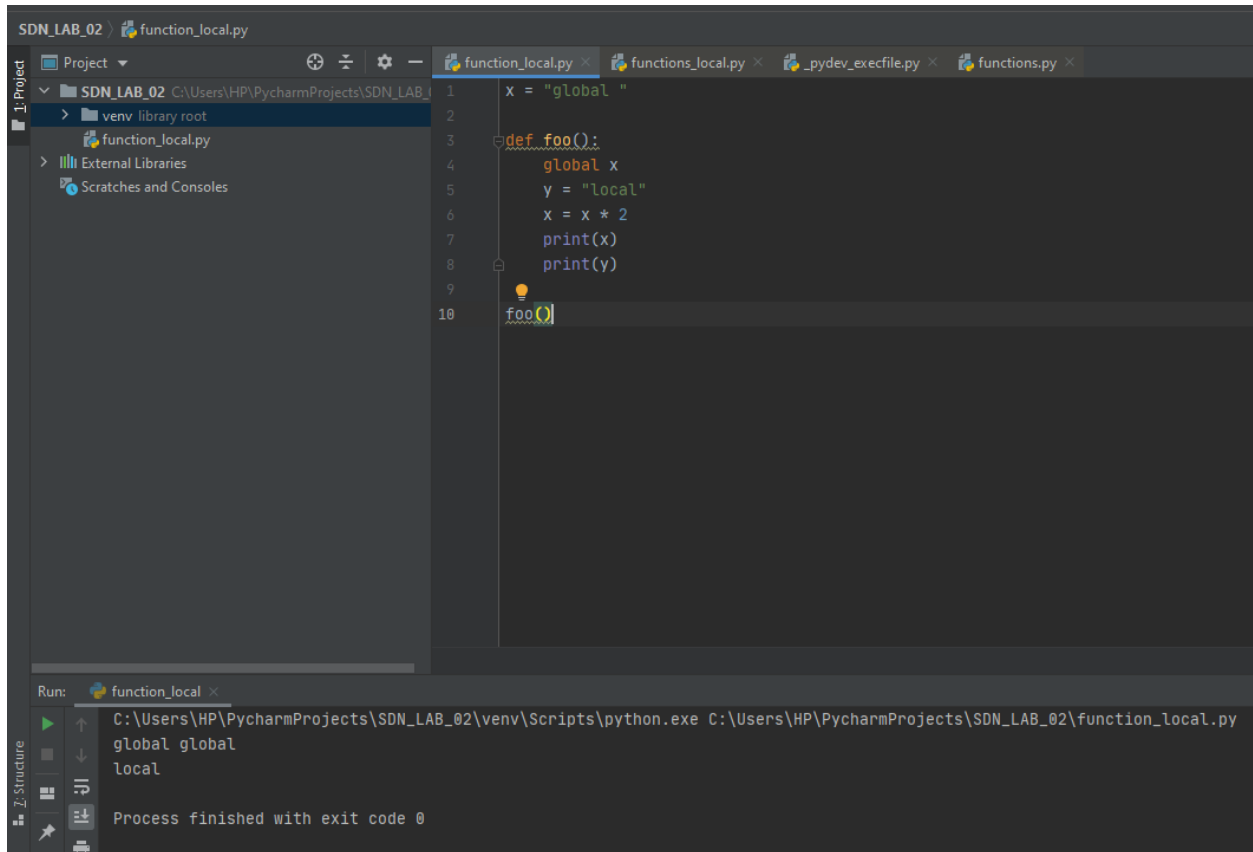
The bottom panel shows the 'Run' output for 'Funtions.py':

```
C:\Users\HP\PycharmProjects\SDN_LAB_02\venv\Scripts\python.exe C:\Users\HP\PycharmProjects\SDN_LAB_02\Funtions.py  
Hello from a function  
  
Process finished with exit code 0
```

#### Exercise 4.1.3: Python function (save as function\_2.py)



#### Exercise 4.1.4: Local variable (save as function\_local.py)

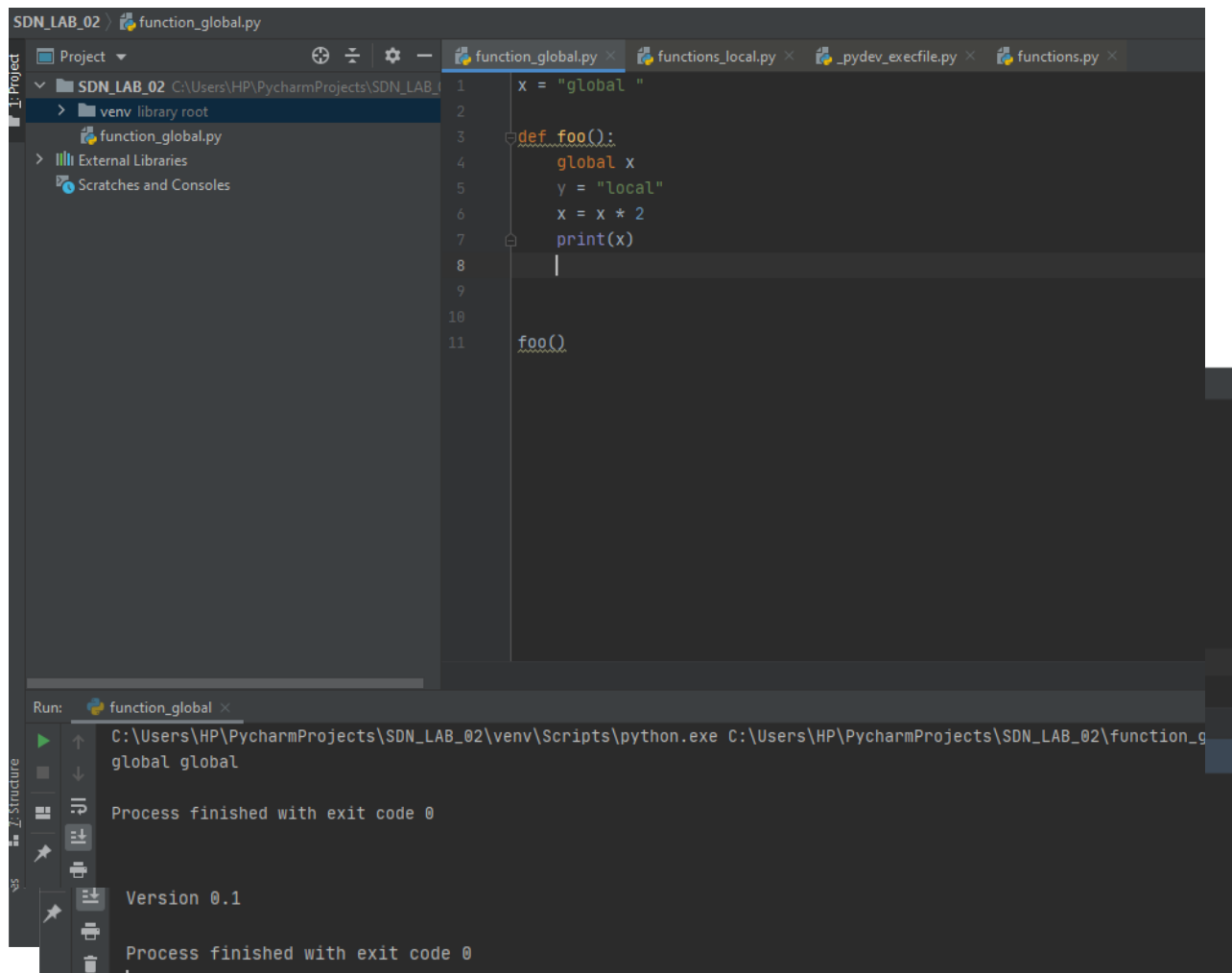


The screenshot shows the PyCharm IDE with a project named 'SDN\_LAB\_02'. The file explorer on the left shows the project structure, including a 'venv' directory and a file named 'function\_local.py'. The main editor window displays the code for 'function\_local.py'.

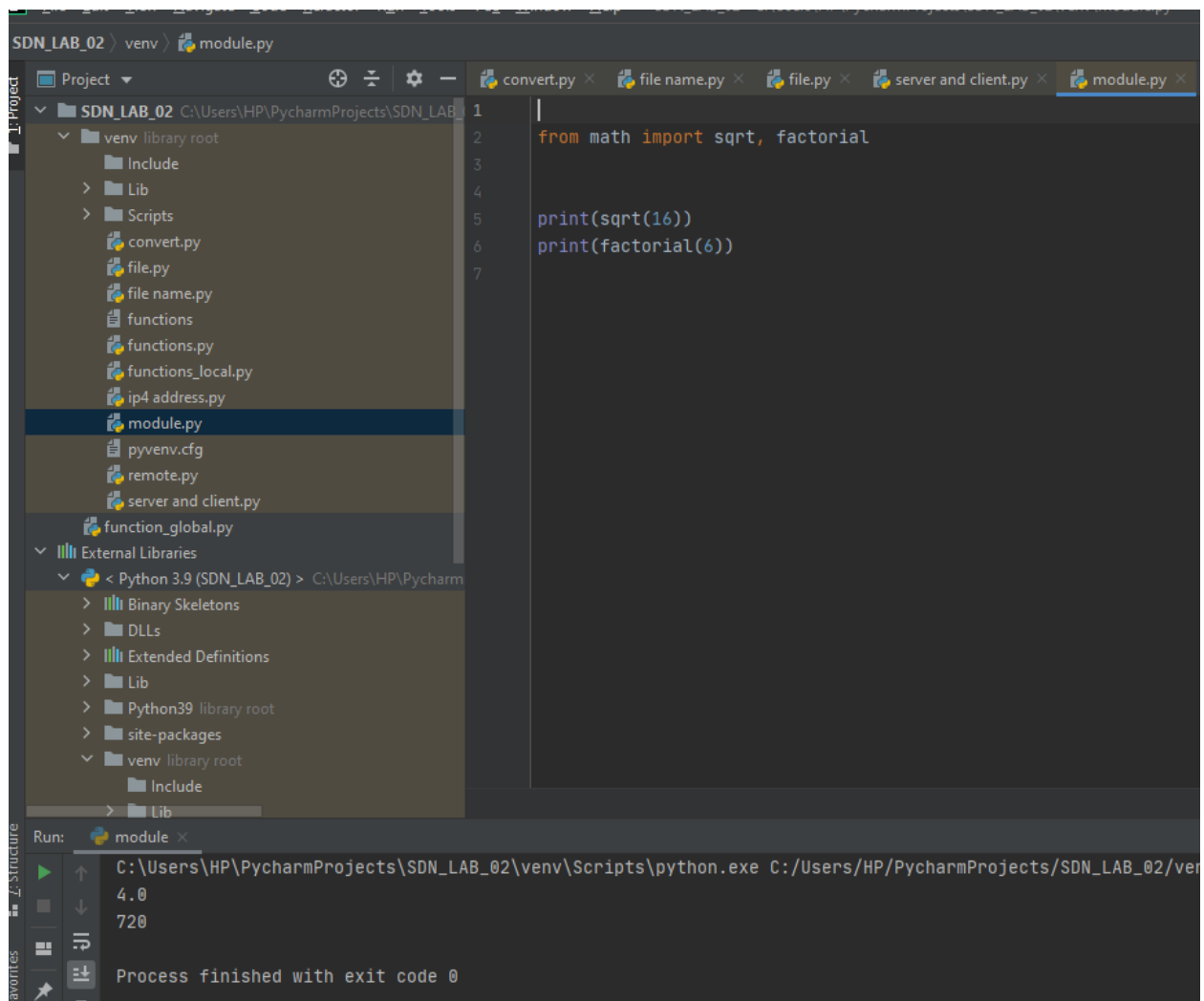
```
1 x = "global "  
2  
3 def foo():  
4     global x  
5     y = "local"  
6     x = x * 2  
7     print(x)  
8     print(y)  
9  
10 foo()
```

The bottom panel shows the Run output for 'function\_local.py'. The command executed is 'C:\Users\HP\PycharmProjects\SDN\_LAB\_02\venv\Scripts\python.exe C:\Users\HP\PycharmProjects\SDN\_LAB\_02\function\_local.py'. The output is 'global global' followed by 'local' on the next line. The process finished with exit code 0.

#### Exercise 4.1.5: Global variable (save as function\_global.py)



#### Exercise 4.1.6: Python modules



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

The screenshot shows the PyCharm IDE interface. The left sidebar displays the project structure for 'SDN\_LAB\_02', with 'ip4 address.py' selected under the 'Scripts' folder. The main editor window shows the code for 'ip4 address.py'.

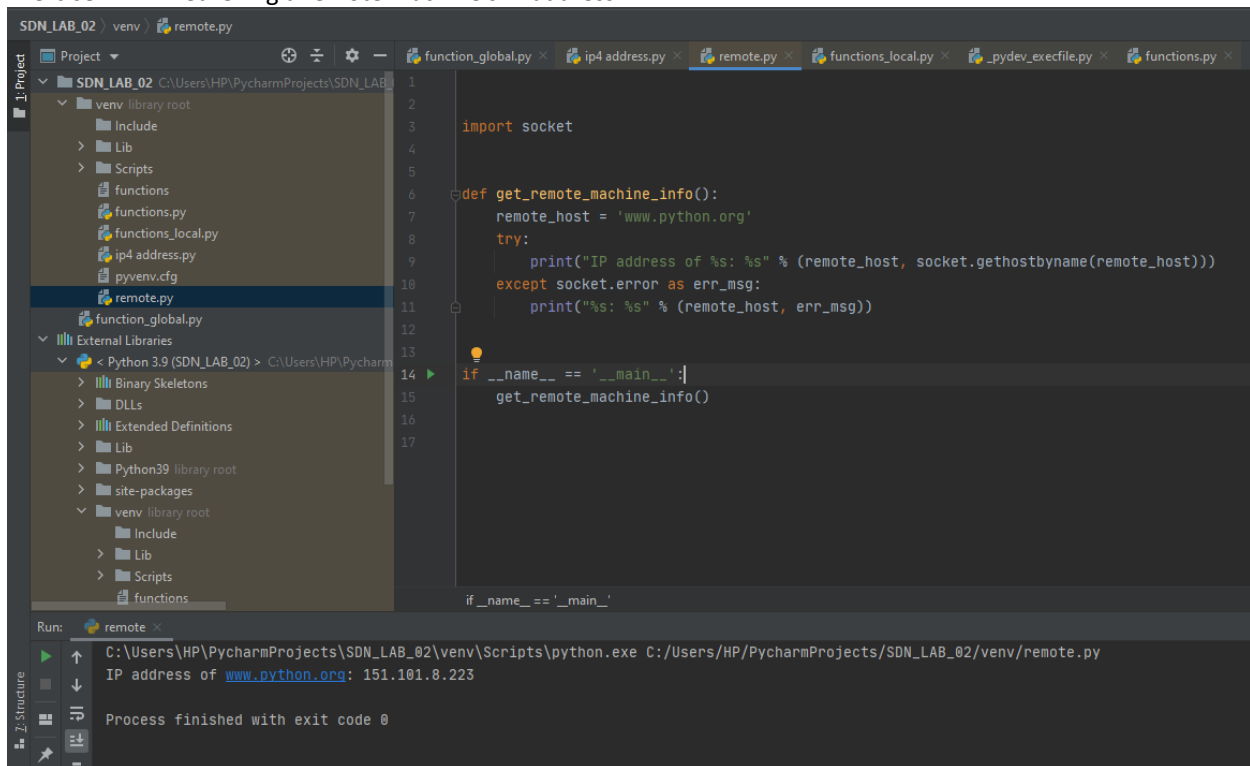
```
1 # Python Program to Get IP Address
2 import socket
3 hostname = socket.gethostname()
4 IPAddr = socket.gethostbyname(hostname)
5 print("Your Computer Name is:" + hostname)
6 print("Your Computer IP Address is:" + IPAddr)
7
```

The bottom panel shows the 'Run' output for 'ip4 address.py':

```
C:\Users\HP\PycharmProjects\SDN_LAB_02\venv\Scripts\python.exe "C:/Users/HP/PycharmProjects/SDN_LAB_02/venv/ip4 address.py"
Your Computer Name is:Sohan_IT
Your Computer IP Address is:192.168.56.1
Process finished with exit code 0
```



#### Exercise 4.2.2: Retrieving a remote machine's IP address



The screenshot shows the PyCharm IDE interface. The left sidebar displays the project structure for 'SDN\_LAB\_02', including a virtual environment 'venv' with its own library root and scripts. The main editor window shows the file 'remote.py' with the following Python code:

```
1
2
3 import socket
4
5
6 def get_remote_machine_info():
7     remote_host = 'www.python.org'
8     try:
9         print("IP address of %s: %s" % (remote_host, socket.gethostbyname(remote_host)))
10    except socket.error as err_msg:
11        print("%s: %s" % (remote_host, err_msg))
12
13
14 if __name__ == '__main__':
15     get_remote_machine_info()
16
17
18 if __name__ == '__main__':
```

The bottom panel shows the 'Run' output for the script:

```
Run: remote x
C:\Users\HP\PycharmProjects\SDN_LAB_02\venv\Scripts\python.exe C:/Users/HP/PycharmProjects/SDN_LAB_02/venv/remote.py
IP address of www.python.org: 151.101.8.223
Process finished with exit code 0
```

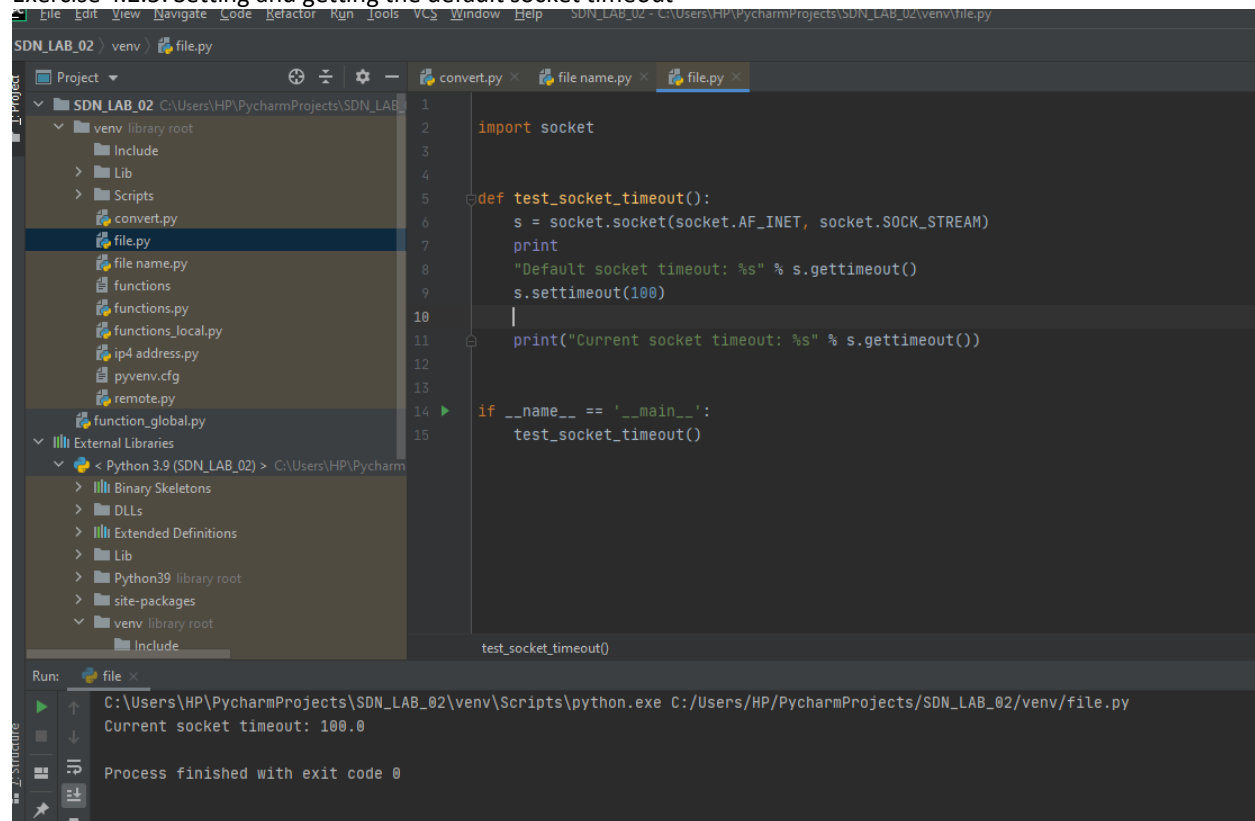
#### Exercise 4.2.3: Converting an IPv4 address to different formats

```
SDN_LAB_02 \ venv \ convert.py
Project
  SDN_LAB_02
    venv
      library root
      Include
      Lib
      Scripts
      convert.py
      functions
      functions.py
      functions_local.py
      ip4 address.py
      pyvenv.cfg
      remote.py
      function_global.py
    External Libraries
      Python 3.9 (SDN_LAB_02)
        Binary Skeletons
        DLLs
        Extended Definitions
        Lib
        Python39 library root
        site-packages
        venv library root
          Include
          Lib
          Scripts
convert.py
3 # This program is optimized for Python 2.7.
4 # It may run on any other version with/without modifications.
5
6 import socket
7 from binascii import hexlify
8
9
10 def convert_ip4_address():
11     for ip_addr in ['127.0.0.1', '192.168.0.1']:
12         packed_ip_addr = socket.inet_aton(ip_addr)
13         unpacked_ip_addr = socket.inet_ntoa(packed_ip_addr)
14
15         print("IP Address: %s => Packed: %s, Unpacked: %s" \
16               % (ip_addr, hexlify(packed_ip_addr), unpacked_ip_addr))
17
18
19 if __name__ == '__main__':
20     convert_ip4_address()
convert_ip4_address() > for ip_addr in ['127.0.0.1', '1...
Run: convert
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

```
SDN_LAB_02 \ venv \ file name.py
Project
  SDN_LAB_02
    venv
      library root
      Include
      Lib
      Scripts
      convert.py
      file name.py
      functions
      functions.py
      functions_local.py
      ip4 address.py
      pyvenv.cfg
      remote.py
      function_global.py
    External Libraries
      Python 3.9 (SDN_LAB_02)
        Binary Skeletons
        DLLs
        Extended Definitions
        Lib
        Python39 library root
        site-packages
        venv library root
          Include
          Lib
          Scripts
convert.py
1
2
3
4 import socket
5
6
7 def find_service_name():
8     protocolname = 'tcp'
9     for port in [80, 25]:
10         print("Port: %s => service name: %s" % (port, socket.getservbyport(port, protocolname)))
11
12     print("Port: %s => service name: %s" % (53, socket.getservbyport(53, 'udp')))
13
14
15 if __name__ == '__main__':
16     find_service_name()
17
18
19 find_service_name()
Run: file 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

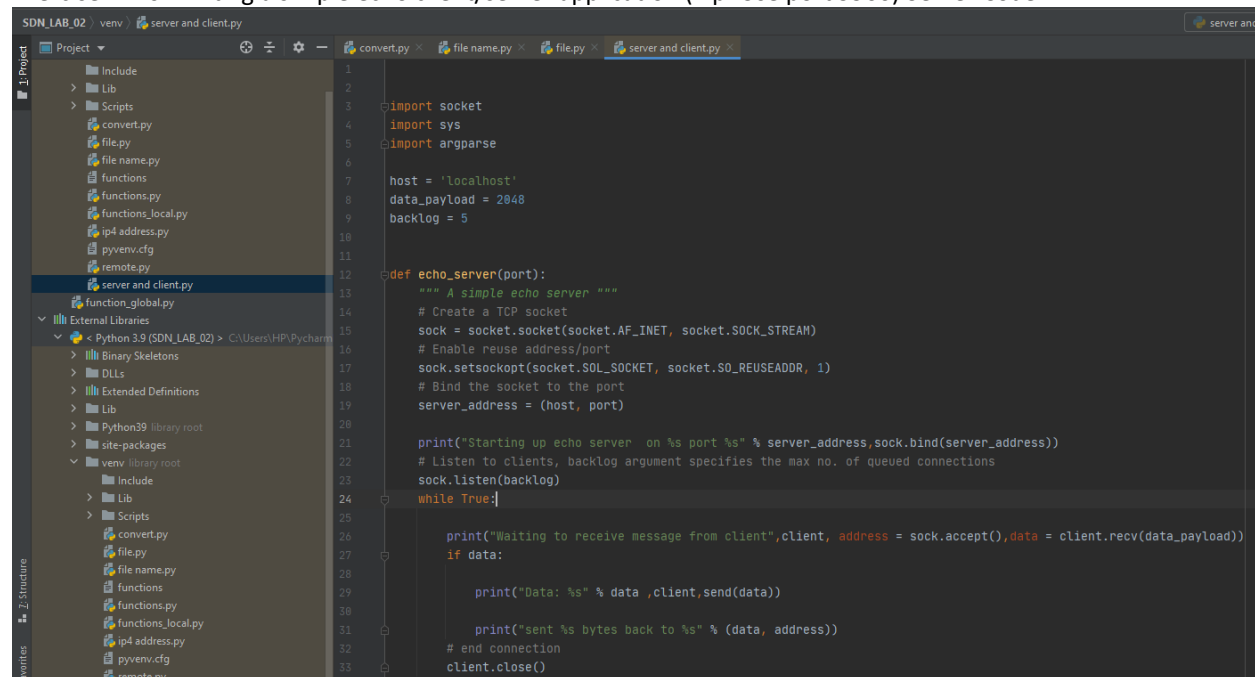


The screenshot shows the PyCharm IDE with a project named 'SDN\_LAB\_02'. The file explorer on the left shows a directory structure with files like 'convert.py', 'file.py', 'file name.py', 'functions.py', 'functions\_local.py', 'ip4 address.py', 'pyvenv.cfg', 'remote.py', and 'function\_global.py'. The 'file.py' file is open in the editor, showing the following code:

```
1 import socket
2
3
4
5 def test_socket_timeout():
6     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7     print
8     "Default socket timeout: %s" % s.gettimeout()
9     s.settimeout(100)
10
11 |
12     print("Current socket timeout: %s" % s.gettimeout())
13
14
15 if __name__ == '__main__':
16     test_socket_timeout()
```

The Run window at the bottom shows the command executed: `C:\Users\HP\PycharmProjects\SDN_LAB_02\venv\Scripts\python.exe C:/Users/HP/PycharmProjects/SDN_LAB_02/venv/file.py`. The output is: `Current socket timeout: 100.0`. The process finished with exit code 0.

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



The screenshot shows the PyCharm IDE with a project named 'SDN\_LAB\_02'. The file explorer on the left shows a directory structure with files like 'convert.py', 'file.py', 'file name.py', 'functions.py', 'functions\_local.py', 'ip4 address.py', 'pyvenv.cfg', 'remote.py', and 'server and client.py'. The 'server and client.py' file is open in the editor, showing the following code:

```
1
2
3 import socket
4 import sys
5 import argparse
6
7 host = 'localhost'
8 data_payload = 2048
9 backlog = 5
10
11
12 def echo_server(port):
13     """ A simple echo server """
14     # Create a TCP socket
15     sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
16     # Enable reuse address/port
17     sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
18     # Bind the socket to the port
19     server_address = (host, port)
20
21     print("Starting up echo server on %s port %s" % server_address, sock.bind(server_address))
22     # Listen to clients, backlog argument specifies the max no. of queued connections
23     sock.listen(backlog)
24
25     while True:
26
27         print("Waiting to receive message from client", client, address = sock.accept(), data = client.recv(data_payload))
28         if data:
29             print("Data: %s" % data, client, send(data))
30
31             print("sent %s bytes back to %s" % (data, address))
32
33         # end connection
34         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•