Mawlana Bhashani Science and Technology University

Lab-Report

Course Title : Computer Networks Lab
Lab Report No: 02
Lab Report Name: Programming with Python

Submitted by

Name: Ali Ashadullah Arif &
Ahadul Haque
ID: IT-18031 & IT-18045
3rd Year 2nd Semester
Session: 2017-2018
Dept. of ICT,
MBSTU.

Submitted To

Nazrul Islam
Assistant Professor
Dept. of ICT,
MBSTU.

Lab Report No: 02
Lab Report Name: Programming with Python

Theory:

<u>Python Functions:</u> Functions are a convenient way to divide your code into useful blocks, allowing us to order our code, make it more readable, reuse it and save some time. Also functions are a key way to define interfaces so programmers can share their code. 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.

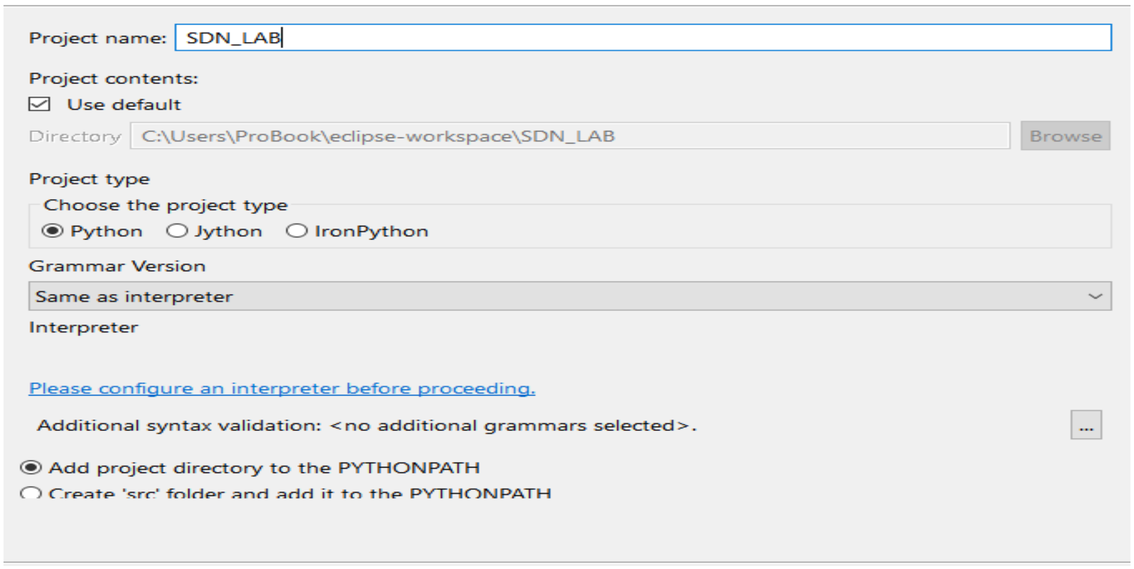
<u>Local Variables:</u> Variables declared inside a function definition are not related in any way to other variables with the same names used outside the function (variable names are local to the function). This is called the scope of the variable. All variables have the scope of the block they are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

<u>The Global Statement:</u> 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.

<u>Modules:</u> Modules allow reusing a number of functions in other programs.

Exercises:

Exercise 4.1.1: Create a python project using with SDN_LAB



Project name: SDN_LAB

Project contents:

☑ Use default

Directory C:\Users\ProBook\eclipse-workspace\SDN_LAB

Project type

Choose the project type

● Python ○ Jython ○ IronPython

Grammar Version

Same as interpreter

Interpreter

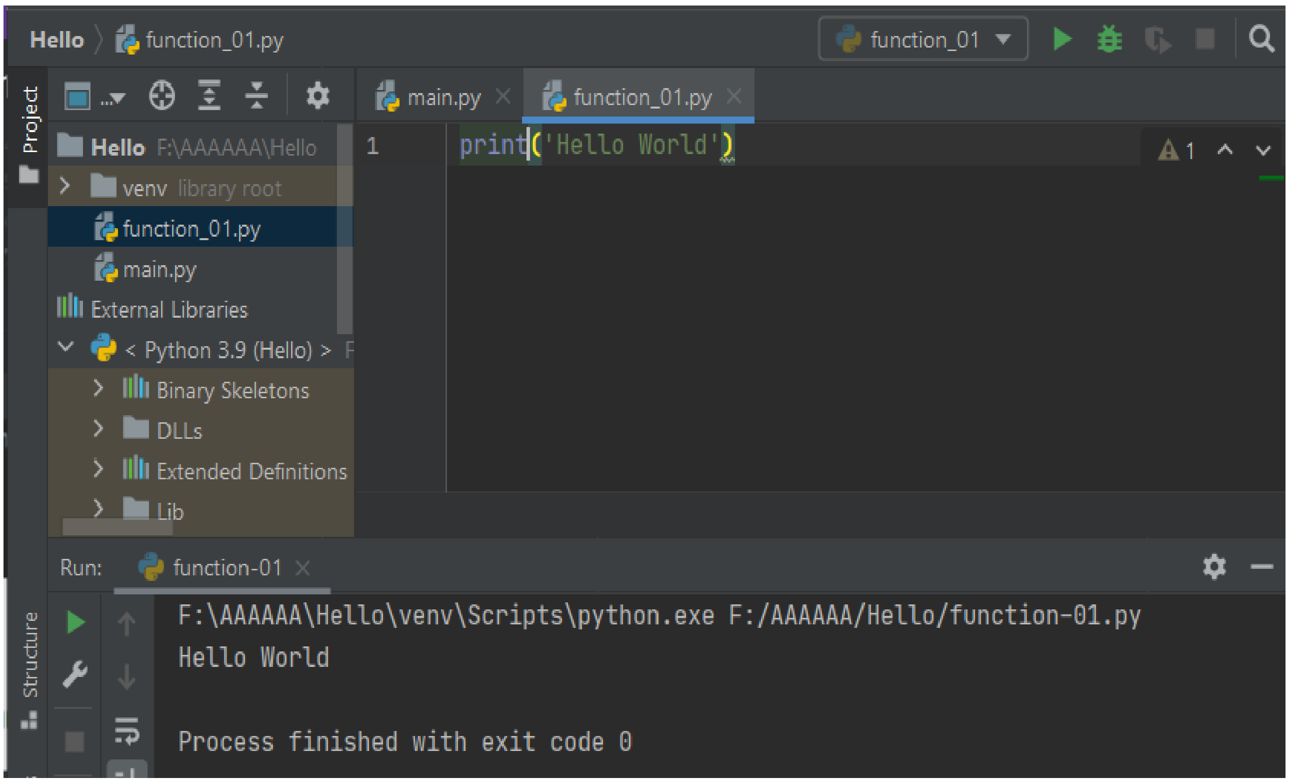
Please configure an interpreter before proceeding.

Additional syntax validation: <no additional grammars selected>.

● Add project directory to the PYTHONPATH

○ Create 'src' folder and add it to the PYTHONPATH

Exercise 4.1.2: Python function (save as function_01.py)

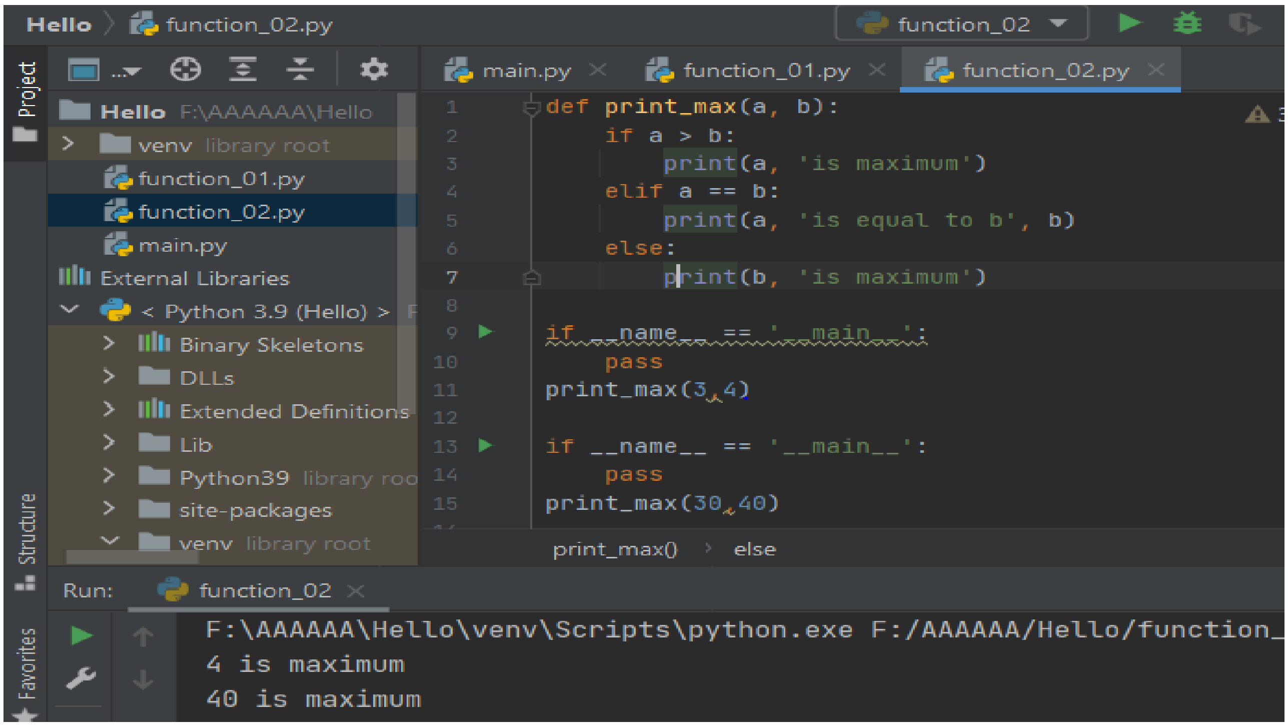


```
print('Hello World')
```

```
F:\AAAAAA\Hello\venv\Scripts\python.exe F:/AAAAAA/Hello/function-01.py
Hello World

Process finished with exit code 0
```

Exercise 4.1.3: Python function (save as function_02.py)



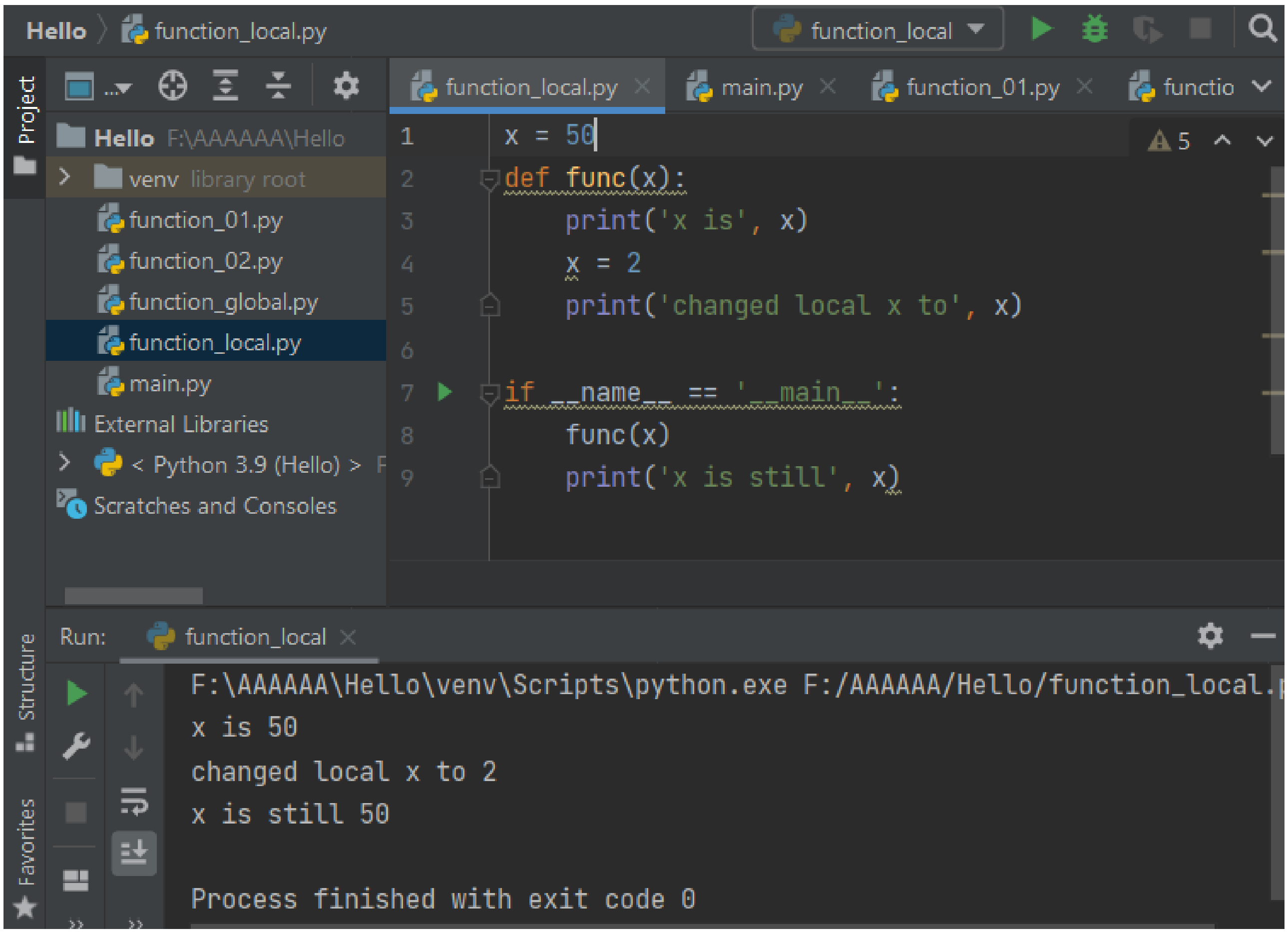
```python
def print_max(a, b):
    if a > b:
        print(a, 'is maximum')
    elif a == b:
        print(a, 'is equal to b', b)
    else:
        print(b, 'is maximum')

if __name__ == '__main__':
    pass
print_max(3,4)

if __name__ == '__main__':
    pass
print_max(30,40)
```

```
F:\AAAAAA\Hello\venv\Scripts\python.exe F:/AAAAAA/Hello/function_
4 is maximum
40 is maximum
```

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



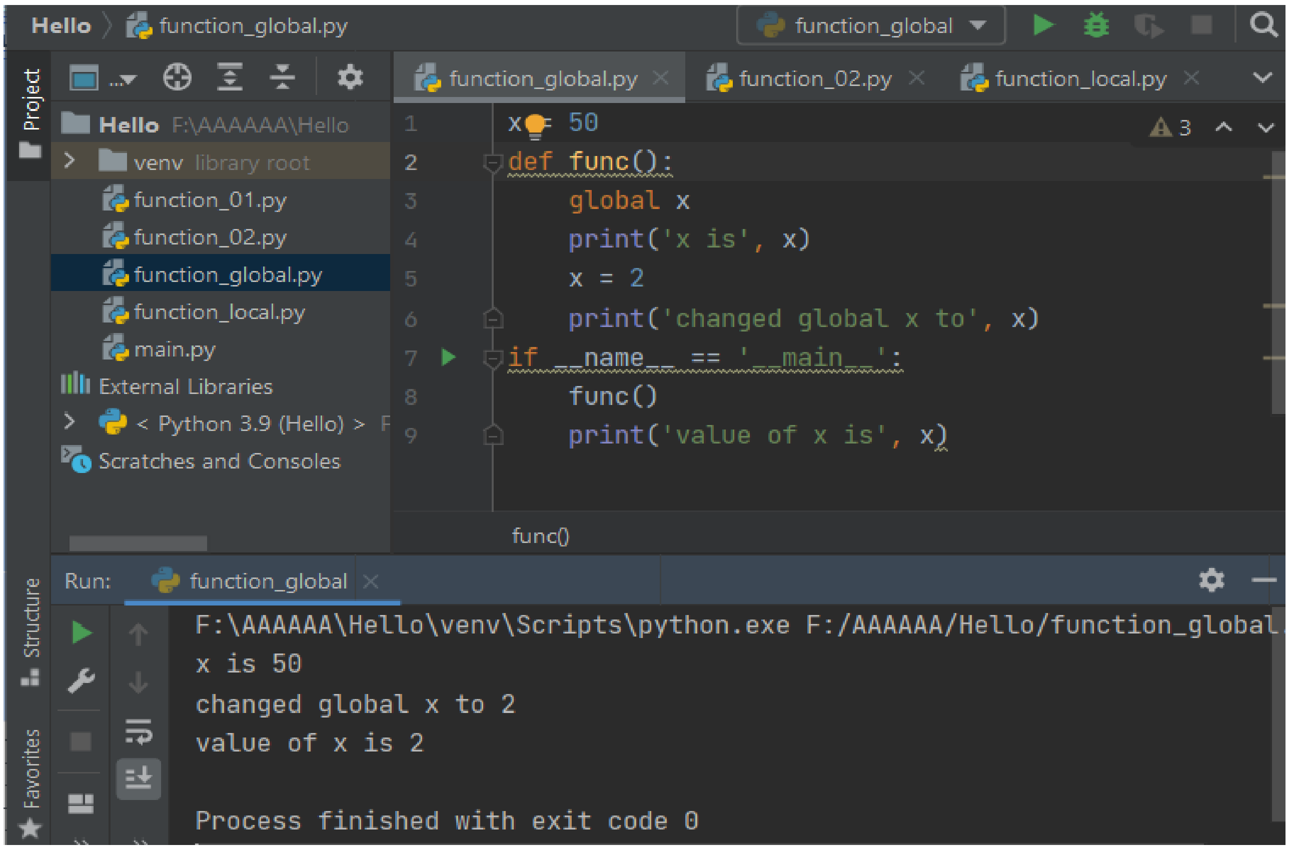
```python
x = 50
def func(x):
    print('x is', x)
    x = 2
    print('changed local x to', x)

if __name__ == '__main__':
    func(x)
    print('x is still', x)
```

```
F:\AAAAAA\Hello\venv\Scripts\python.exe F:/AAAAAA/Hello/function_local.p
x is 50
changed local x to 2
x is still 50

Process finished with exit code 0
```

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

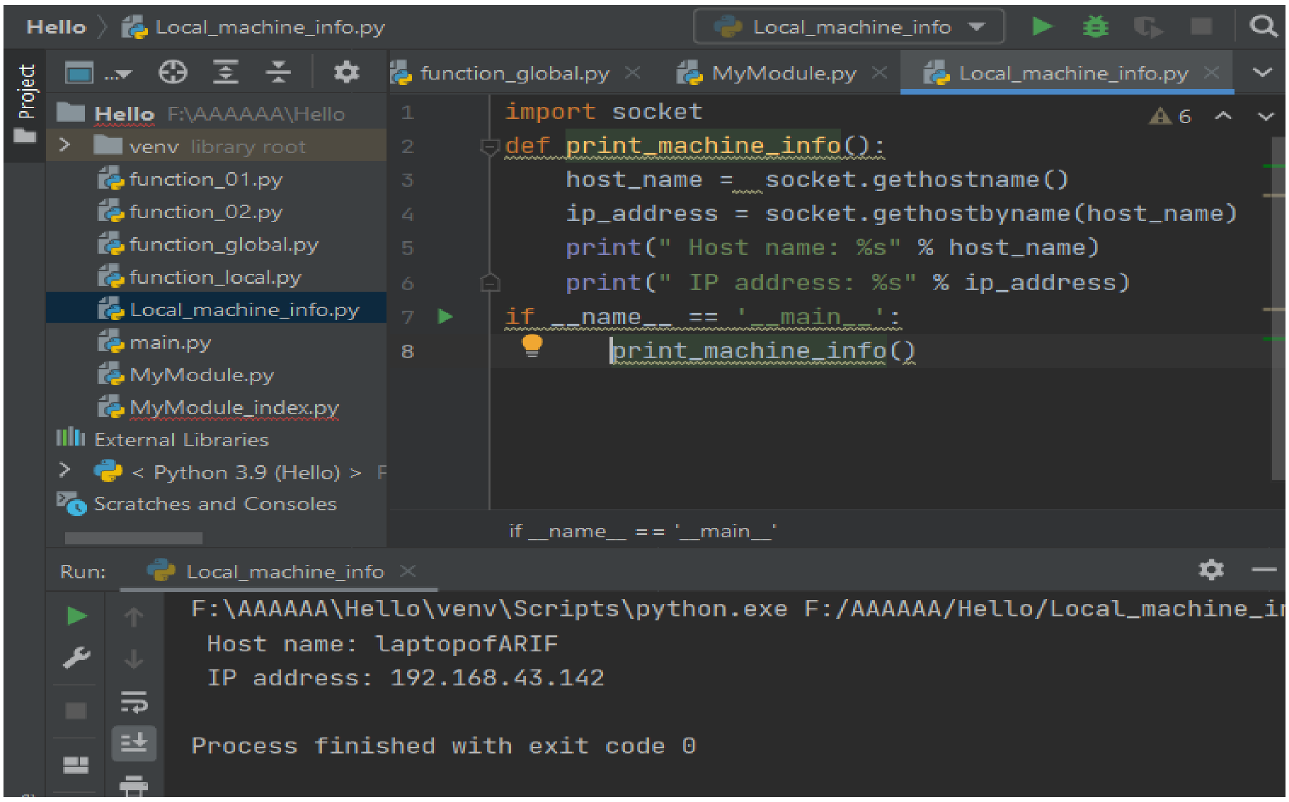


```
x = 50
def func():
    global x
    print('x is', x)
    x = 2
    print('changed global x to', x)
if __name__ == '__main__':
    func()
    print('value of x is', x)
```

```
F:\AAAAAA\Hello\venv\Scripts\python.exe F:/AAAAAA/Hello/function_global
x is 50
changed global x to 2
value of x is 2

Process finished with exit code 0
```

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

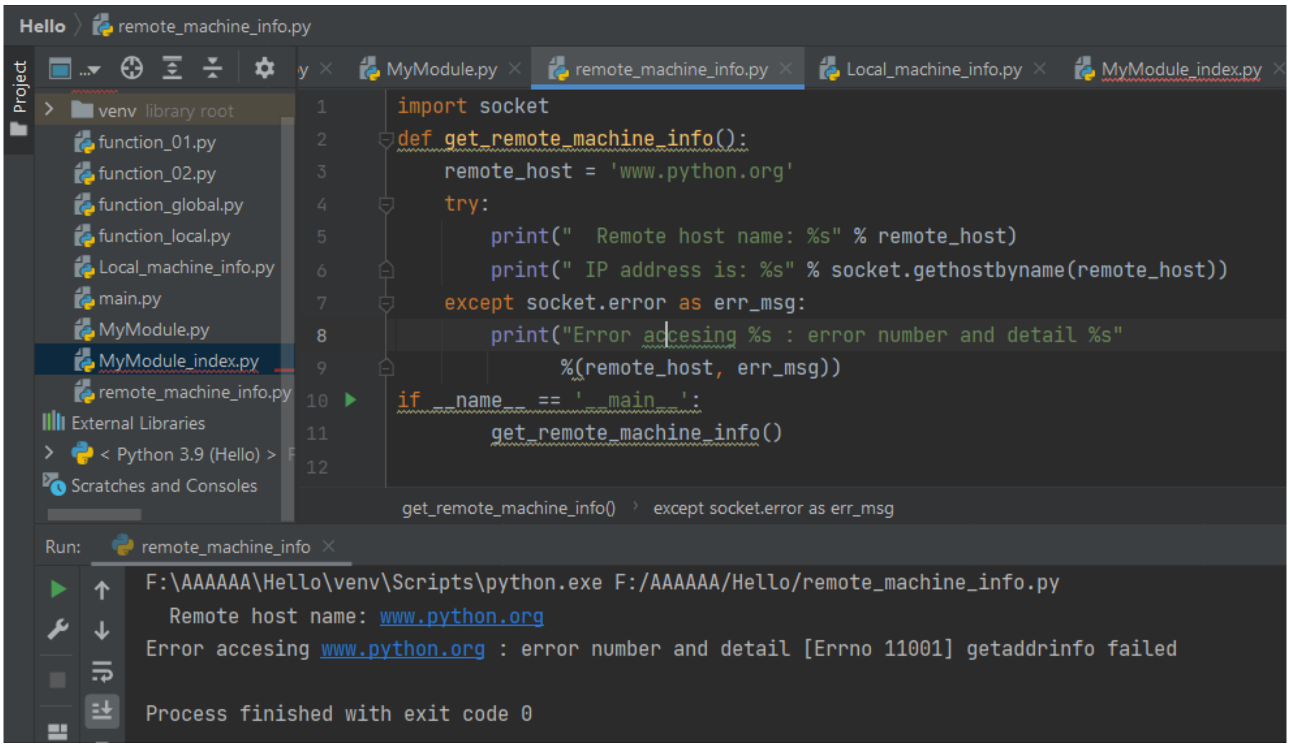


```
import socket
def print_machine_info():
    host_name = socket.gethostname()
    ip_address = socket.gethostbyname(host_name)
    print(" Host name: %s" % host_name)
    print(" IP address: %s" % ip_address)
if __name__ == '__main__':
    print_machine_info()
```

```
F:\AAAAAA\Hello\venv\Scripts\python.exe F:/AAAAAA/Hello/Local_machine_i
 Host name: laptopofARIF
 IP address: 192.168.43.142

Process finished with exit code 0
```

Exercise 4.2.2: Retrieving a remote machine's IP address

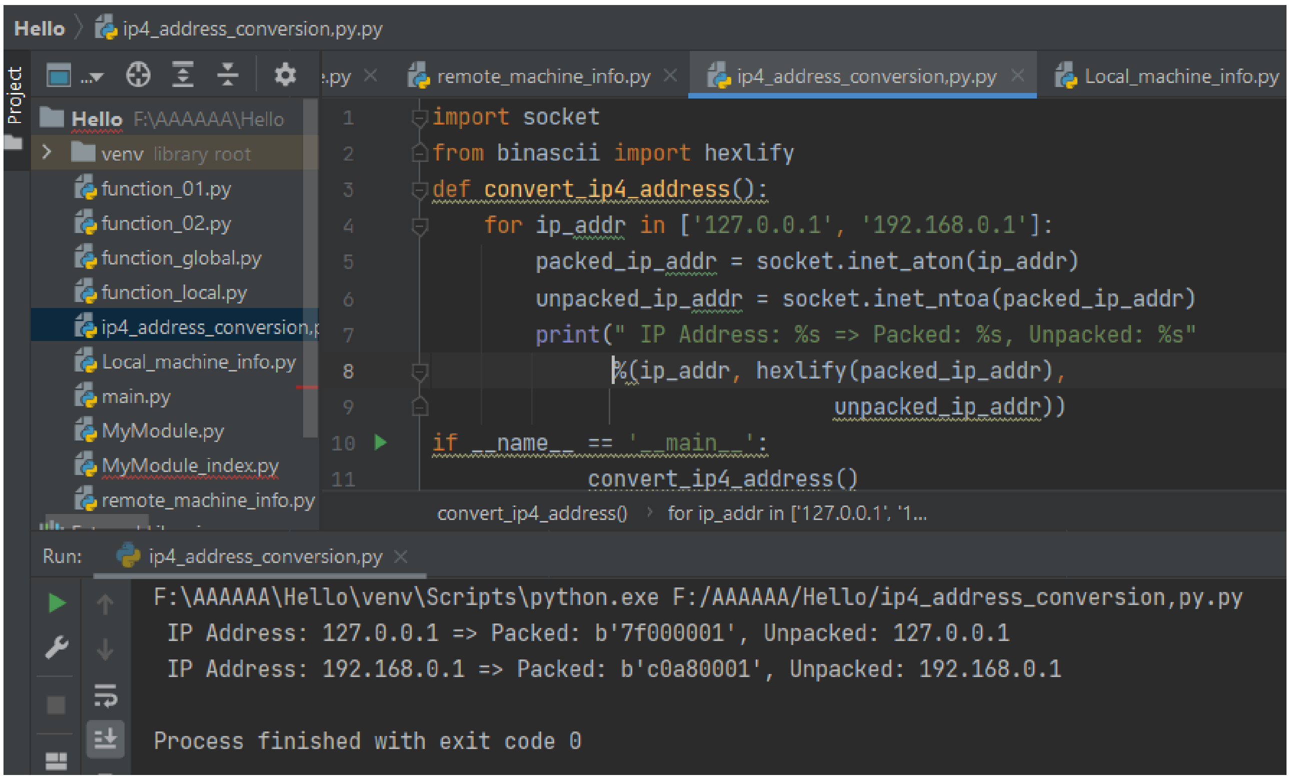


```python
import socket
def get_remote_machine_info():
    remote_host = 'www.python.org'
    try:
        print("  Remote host name: %s" % remote_host)
        print(" IP address is: %s" % socket.gethostbyname(remote_host))
    except socket.error as err_msg:
        print("Error accesing %s : error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

```
F:\AAAAAA\Hello\venv\Scripts\python.exe F:/AAAAAA/Hello/remote_machine_info.py
  Remote host name: www.python.org
Error accesing www.python.org : error number and detail [Errno 11001] getaddrinfo failed

Process finished with exit code 0
```

Exercise 4.2.3: Converting an IPv4 address to different formats

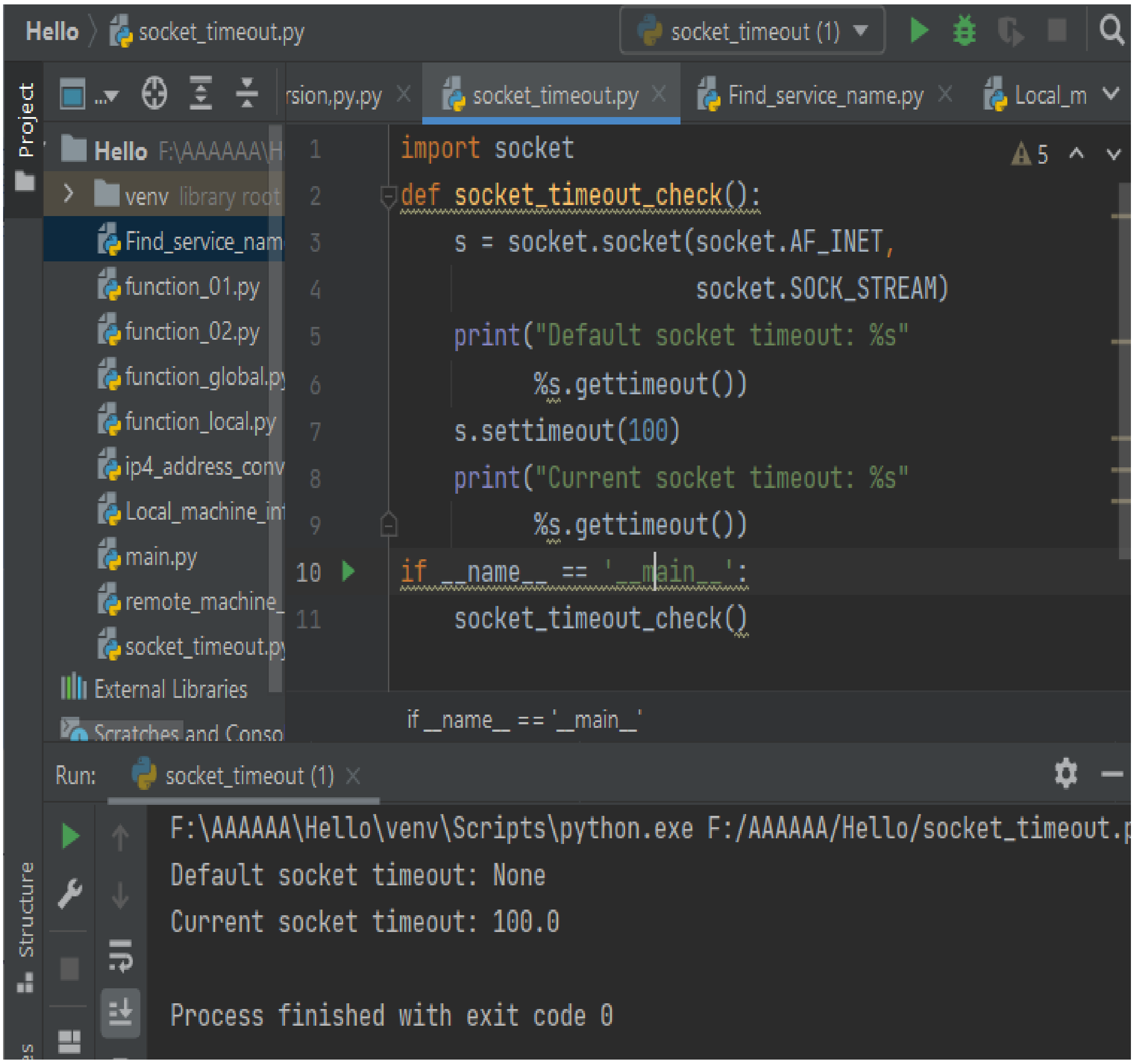


```python
import socket
from binascii import hexlify
def convert_ip4_address():
    for ip_addr in ['127.0.0.1', '192.168.0.1']:
        packed_ip_addr = socket.inet_aton(ip_addr)
        unpacked_ip_addr = socket.inet_ntoa(packed_ip_addr)
        print(" IP Address: %s => Packed: %s, Unpacked: %s"
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

```
F:\AAAAAA\Hello\venv\Scripts\python.exe F:/AAAAAA/Hello/ip4_address_conversion,py.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: Setting and getting the default socket timeout



```
import socket
def socket_timeout_check():
    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__':
    socket_timeout_check()
```

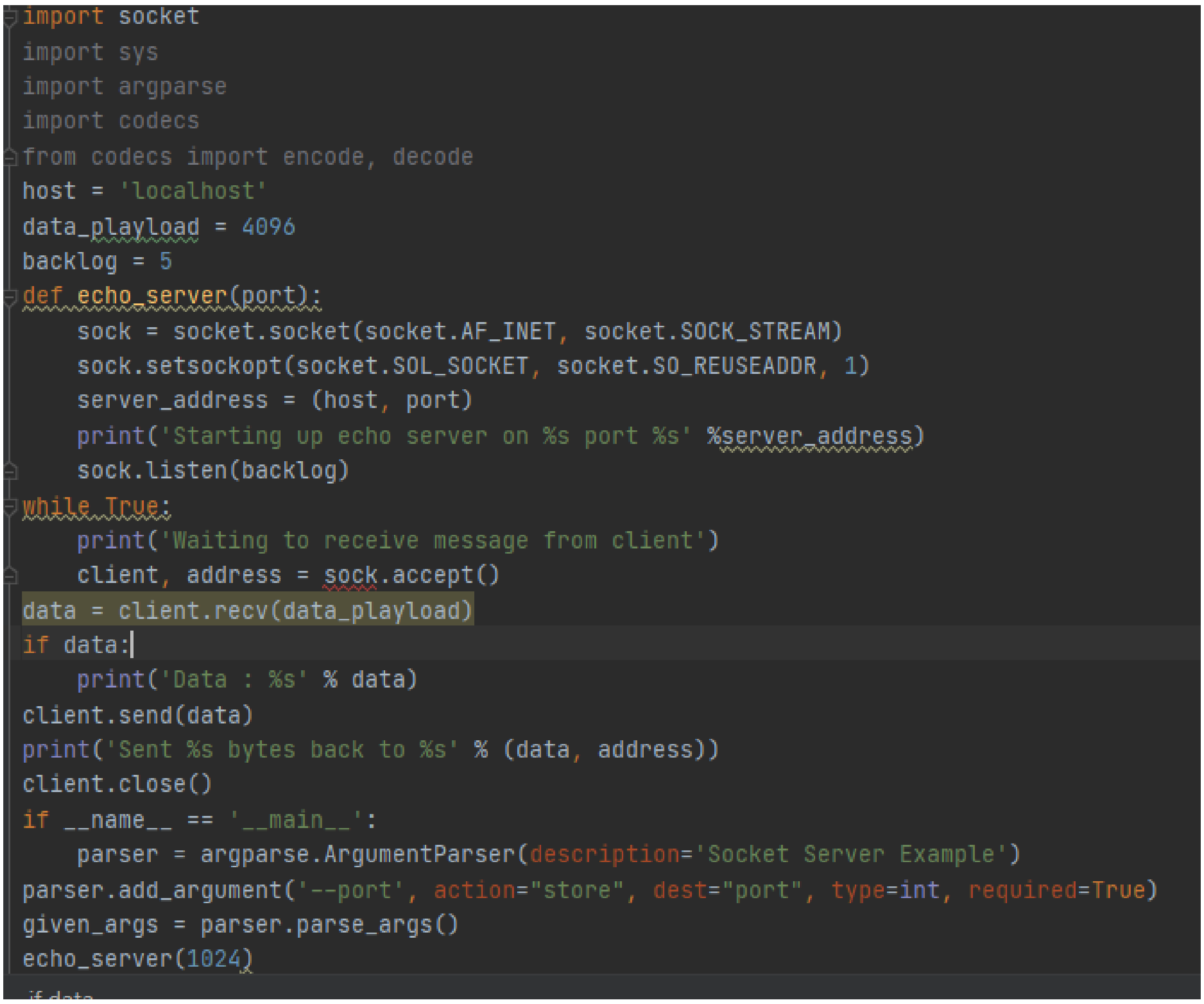
if __name__ == '__main__'

```
F:\AAAAAA\Hello\venv\Scripts\python.exe F:/AAAAAA/Hello/socket_timeout.p
Default socket timeout: None
Current socket timeout: 100.0

Process finished with exit code 0
```

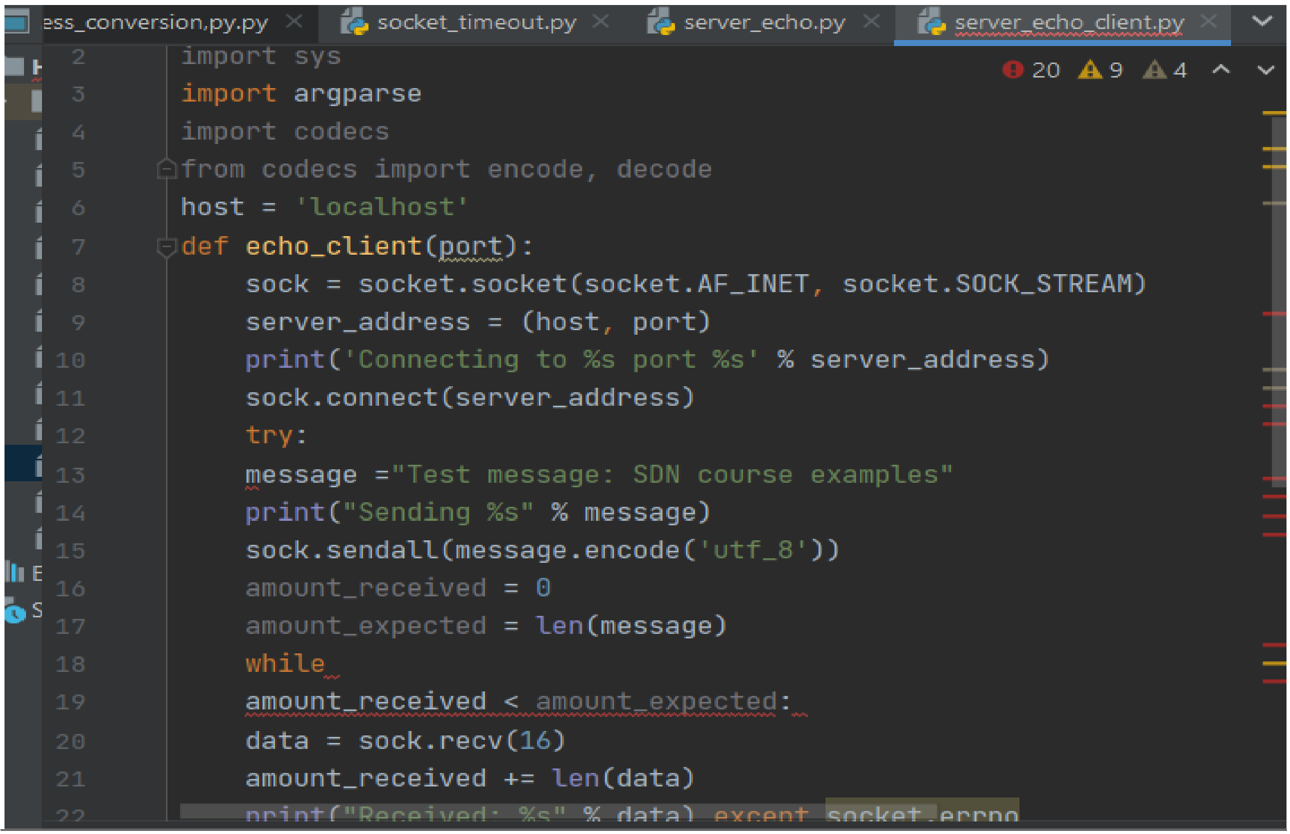
Exercise 4.2.4: Writing a simple echo client/server application (Tip: Use port 9900)

<u>Server Code:</u>

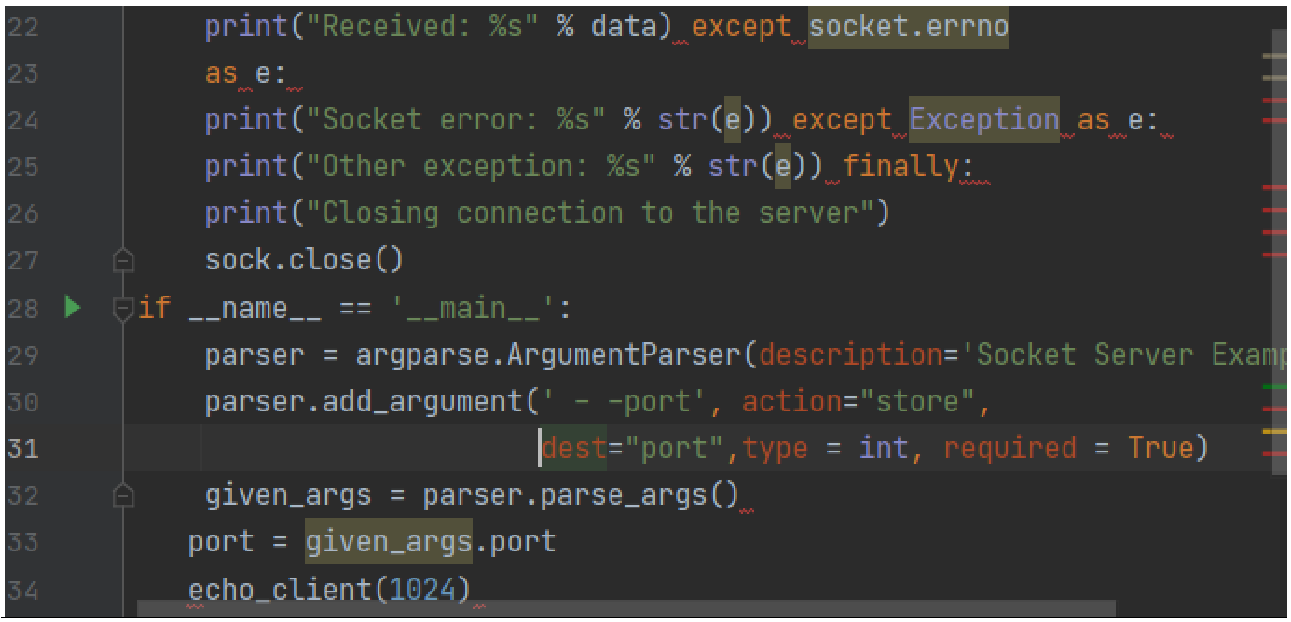


```python
import socket
import sys
import argparse
import codecs
from codecs import encode, decode
host = 'localhost'
data_playload = 4096
backlog = 5
def echo_server(port):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_address = (host, port)
    print('Starting up echo server on %s port %s' %server_address)
    sock.listen(backlog)
while True:
    print('Waiting to receive message from client')
    client, address = sock.accept()
data = client.recv(data_playload)
if data:
    print('Data : %s' % data)
client.send(data)
print('Sent %s bytes back to %s' % (data, address))
client.close()
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Socket Server Example')
parser.add_argument('--port', action="store", dest="port", type=int, required=True)
given_args = parser.parse_args()
echo_server(1024)
```

Client Code:



```python
import sys
import argparse
import codecs
from codecs import encode, decode
host = 'localhost'
def echo_client(port):
    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 ="Test 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)
```



```python
    print("Received: %s" % data) except socket.errno
    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__':
    parser = argparse.ArgumentParser(description='Socket Server Examp
    parser.add_argument(' - -port', action="store",
                        dest="port",type = int, required = True)
    given_args = parser.parse_args()
   port = given_args.port
   echo_client(1024)
```

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) etc.