MAWLANA BHASHANI SCIENCE AND TECHNOLOY UNIVERSITY

Santosh,Tangail – 1902

Course Title: Computer Networks Lab

Submitted by,

Name :NADIRA ISLAM

ID: IT-17051

Session: 2016-17

Dept. of ICT,MBSTU.

Submitted to,

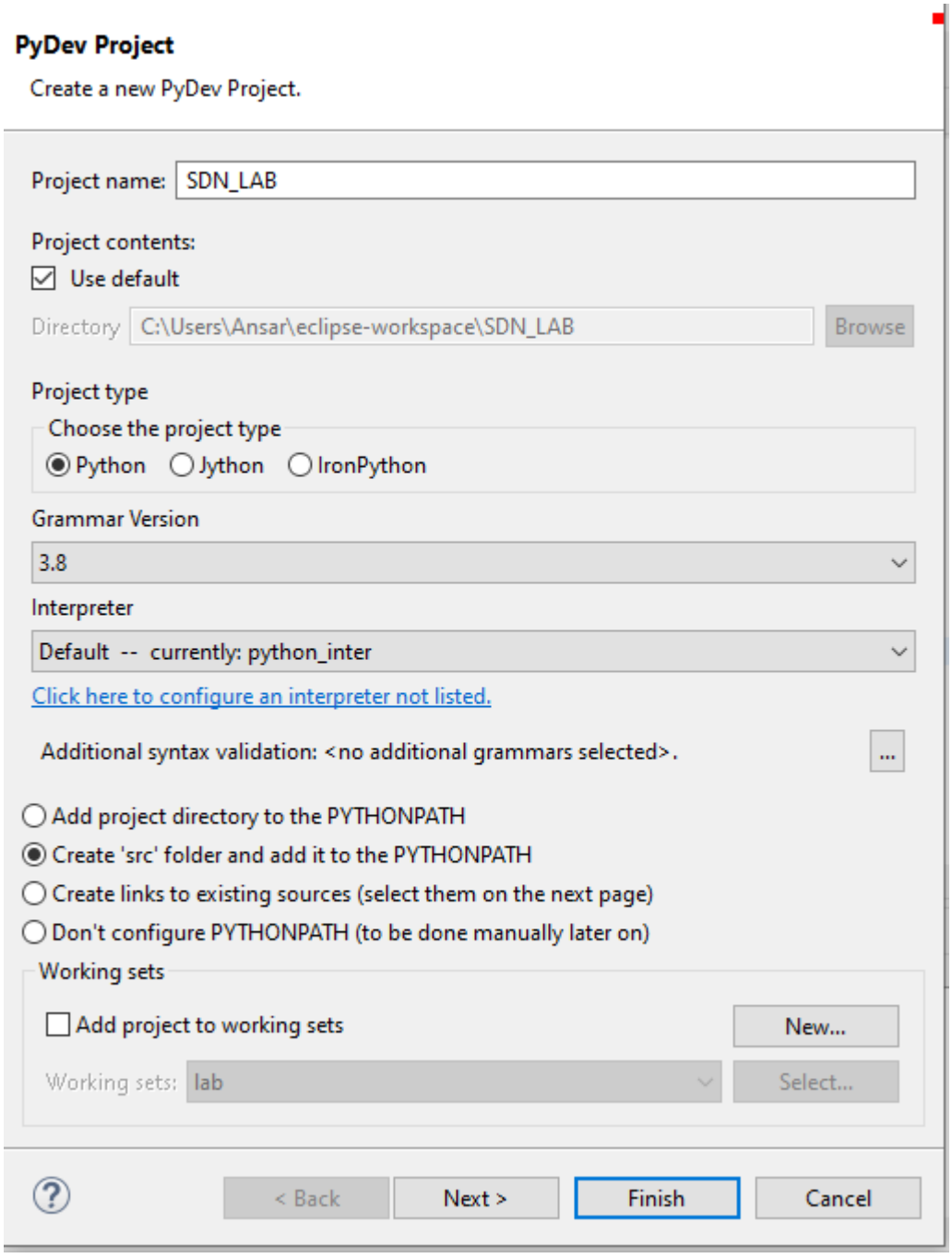
NAZRUL ISLAM

Assistant Professor

Dept. of ICT,MBSTU.

Exercise 4.1.1: Create a python project using with SDN_LAB

Ans:



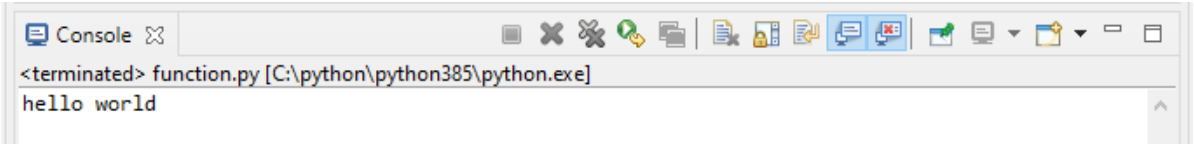
Exercise 4.1.2: Python function (save as function.py)

Create python scrip using the syntax provided below.

```
def say_hello():
print('hello world')
```

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

The output of this code is:

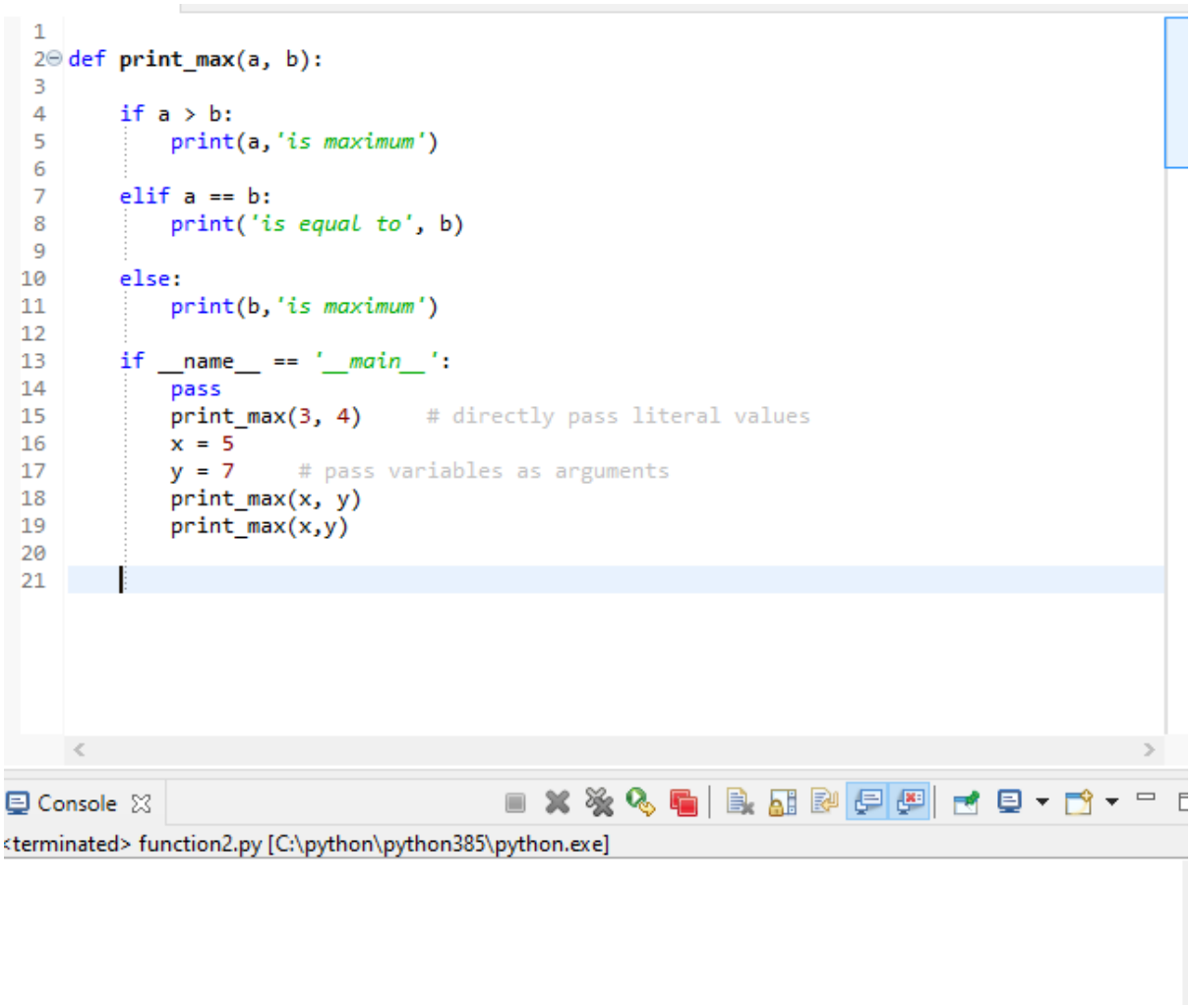


Exercise 4.1.3: Python function (save as function_2.py) Create python scrip using the syntax provided below.

```
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__':
pass
 print_max(3, 4)     # directly pass literal values
x = 5
y = 7     # pass variables as arguments
print_max(x, y)
```

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

Ans: The code does not show any output. May be there is some problem in this code.



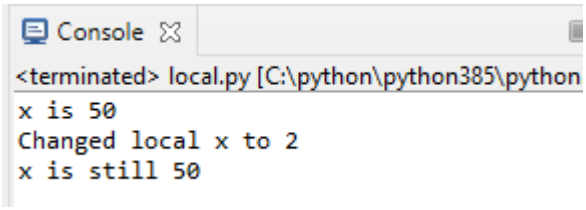
This function does not need any parameter .

Exercise 4.1.4: Local variable (save as function_local.py) Create python scrip using the syntax provided below.

```
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)
```

Which is the final value of variable x? Why variable x does not change to 2?

Ans: Output is:



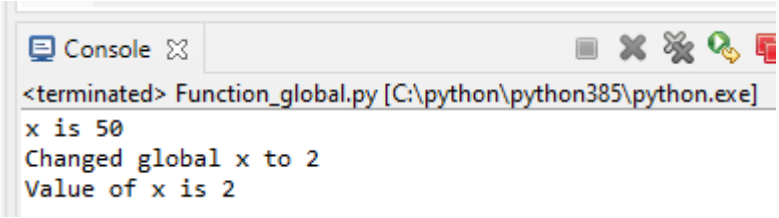
The final value of variable x is 50. It does not change because it is a global variable.

Exercise 4.1.5: Global variable (save as function_global.py) Create python scrip using the syntax provided below.

```
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)
```

Which is the final value of variable x? Why variable x change this time?

Ans: Output is:



This time variable x is declared as global inside the function .So the variable x is changed.

Exercise 4.1.6: Python modules Create python scrip using the syntax provided below (save as mymodule.py).

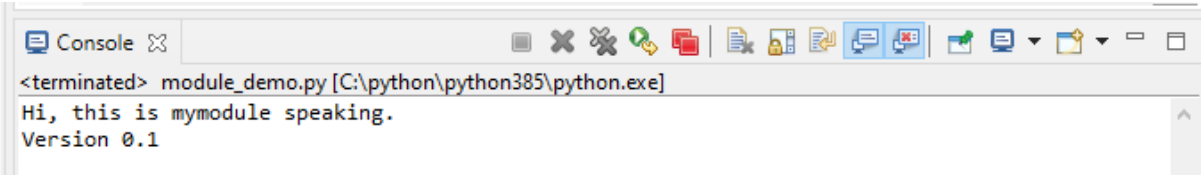
```
def say_hi():
print('Hi, this is mymodule speaking.')
__version__ = '0.1'
```

Create python scrip using the syntax provided below (save as module_demo.py).

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

Run the script, which is the role of import?

Ans: Output is:



Python modules can get access to code from another module by importing the file/function using import. The import statement is the most common way of invoking the import machinery, but it is not the only way. When import is used, it searches for the module initially in the local scope by calling __import__() function

Create python scrip using the syntax provided below (save as module_demo2.py).

from mymodule import say_hi, __version__

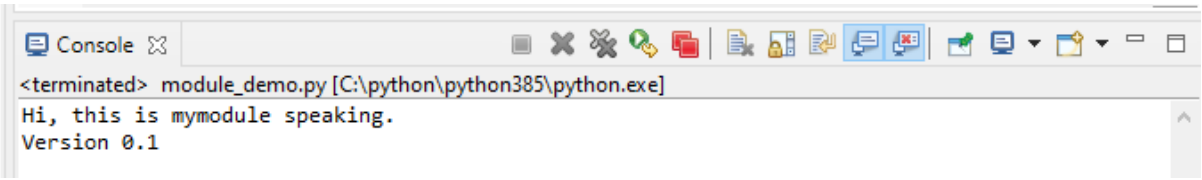
if __name__ == '__main__':

say_hi()

print('Version', __version__)

Run the script, which is the role of from, import?

Ans: Output is :



Using **'from'** we say the module name and then using **'import'** we say what we are importing from the module.

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): import socket

def print_machine_info():

host_name = socket.gethostname()

ip_address = socket.gethostbyname(host_name)

print (" Host name: %s" % host_name)

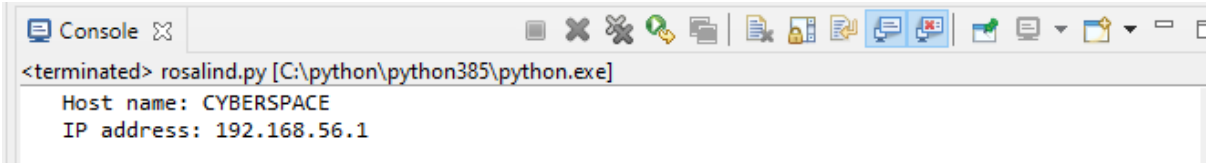
print (" IP address: %s" % ip_address)

if __name__ == '__main__':

print_machine_info()

Run the script, which module the program uses? Provide two additional functions of socket?

Ans:



The *type* argument specifies the socket type, which determines the semantics of communication over the socket. The following socket types are defined; implementations may specify additional socket types:

SOCK_STREAM
: Provides sequenced, reliable, bidirectional, connection-mode byte streams, and may provide a transmission mechanism for out-of-band data.

SOCK_DGRAM
: Provides datagrams, which are connectionless-mode, unreliable messages of fixed maximum length.

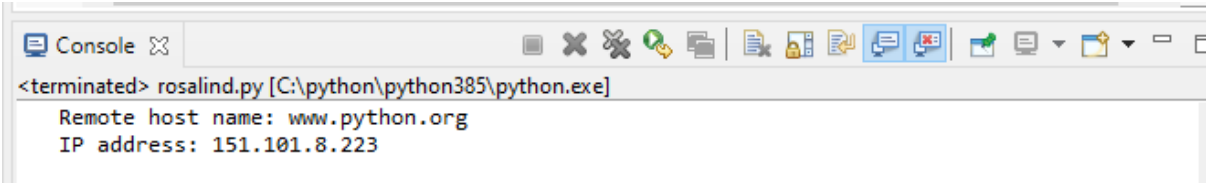
SOCK_SEQPACKET
: Provides sequenced, reliable, bidirectional, connection-mode transmission paths for records. A record can be sent using one or more output operations and received using one or more input operations, but a single operation never transfers part of more than one record. Record boundaries are visible to the receiver via the MSG_EOR flag.

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):

```
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()
```

Run the script, which is the output? Modify the code for getting the RMIT website info.

Ans:



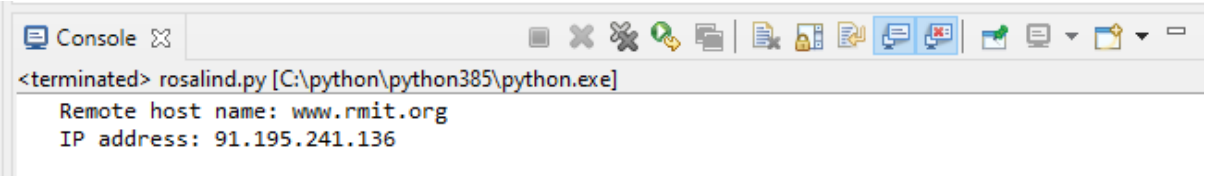
RMIT website info:

Code:

```
import socket
```

```
def get_remote_machine_info():
    remote_host = 'www.rmit.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()
```

Output:



Exercise 4.2.3: Converting an IPv4 address to different formats Create python scrip using the syntax below (save as ip4_address_conversion.py):

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)

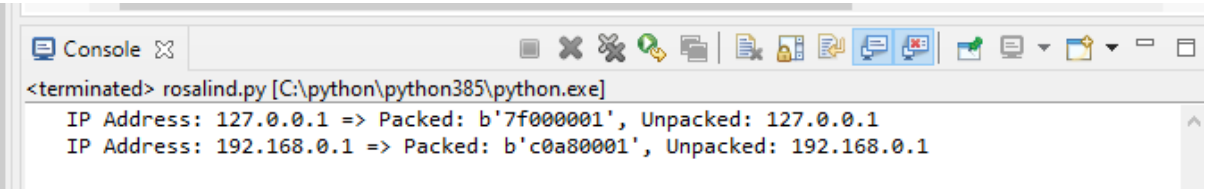
print (" IP Address: %s => Packed: %s, Unpacked: %s" %(ip_addr, hexlify(packed_ip_addr), unpacked_ip_addr))

if __name__ == '__main__':

convert_ip4_address()

Run the script, which is the output? How binascii works?

Ans:



Binascii:

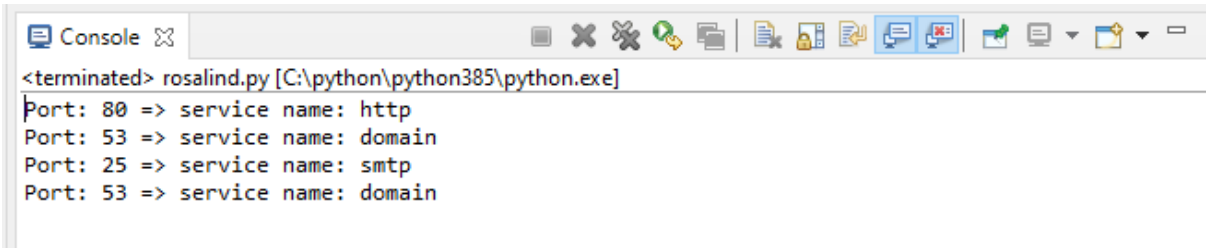
The **binascii** module contains a number of methods to convert between binary and various ASCII-encoded binary representations. ... Convert binary data to a line of ASCII characters, the return value is the converted line, including a newline char. The length of data should be at most 45.

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

```
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()
```

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

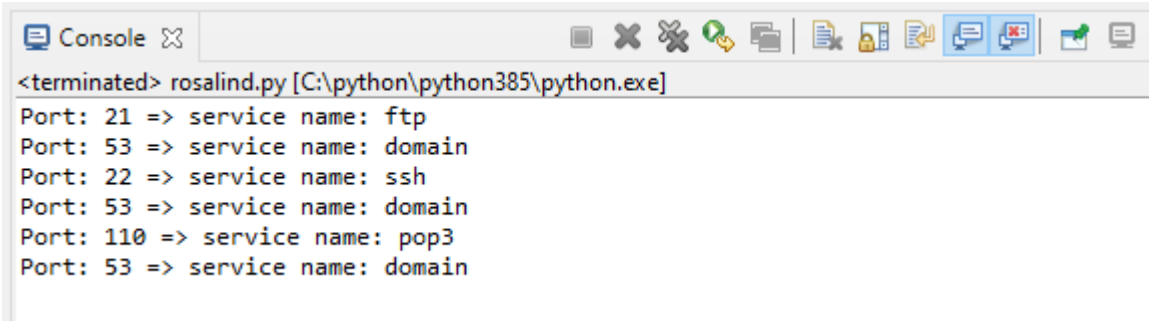
Output:



For the given port the code will be:

```
import socket
def find_service_name():
    protocolname = 'tcp'
    for port in [21,22,110]:
        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()
```

Output:

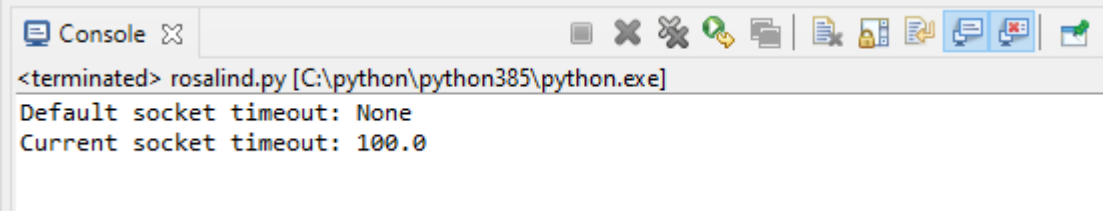


Exercise 4.2.5: Setting and getting the default socket timeout

```
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()
```

Run the script, which is the role of socket timeout in real applications?

Output:



A **socket timeout** implementation should allow for setting the timeout at ... For **example**, this is how we connect to a local HTTP server on port 80 ... It can be implemented as a method that we add to IO::Socket::INET class, possibly by using a **Role**. ... The **real** version handles EINTR and other corner cases.

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

Ans:

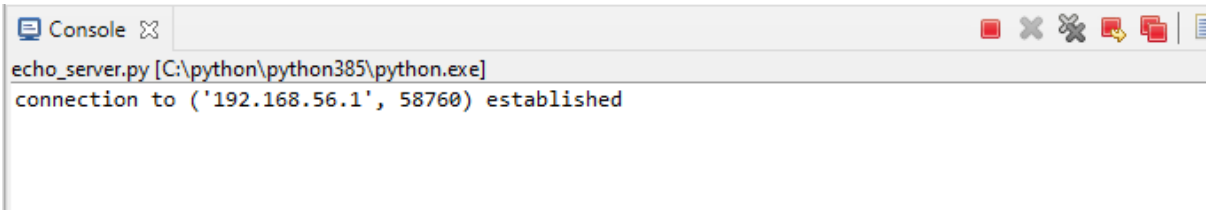
Server:

```
import socket

s= socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.bind((socket.gethostname(),1022))
s.listen(5)
while True:
    clt,adr = s.accept()
    print(f"connection to {adr} established")
    clt.send(bytes("Socket programming in python","utf-8"))
```

Client:

```
import socket
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((socket.gethostname(),1022))
msg = s.recv(1022)
print(msg.decode("utf-8"))
```

Output:



We have to run the server program first ,then client program.