

**MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY,  
Santosh, Tangail -1902**



**Lab Report No** : 05  
**Lab Report Name** : Programing with Python  
**Course Name** : Computer Networks Lab

**Submitted by,**

**Name** : Sabrin Afroz

**ID** : IT-17007

**Session** : 2016-17

Dept. of ICT, MBSTU.

**Submitted to,**

Nazrul Islam

Assistant Professor

Dept. of ICT, MBSTU.

**Objective :**

The objective of this lab is to:

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

**Theory :**

Functions are reusable pieces of programs. They allow us to give a name to a block of statements, allowing us to run that block using the specified name anywhere in the program and any number of times.

Local variables declared inside a function definition are not related in any way to other variables with the same names used outside the function.

Global variables defined at the top level of the program are intended global. Global variables are intended to be used in any functions or classes.

A socket is one endpoint of a two-way communication link between two programs running on the network or PC. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to. Endpoint: An endpoint is a combination of an IP address and a port number. Server and Client: Normally, a server runs on a specific computer and has a socket that is bound to a specific port number.

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.

User Datagram Protocol (UDP) – a communications protocol that facilitates the exchange of messages between computing devices in a network.

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

##### Code :

```
def say_hello():  
    print("hello world")  
if __name__ == '__main__':  
    say_hello()
```

##### output :

A terminal window titled 'function' with a close button. It shows the command 'C:\ProgramData\Anaconda3\python.exe E:/Python/ComputerNetwork/function.py' being executed. The output is 'hello world' on a new line. At the bottom, it says 'Process finished with exit code 0'.

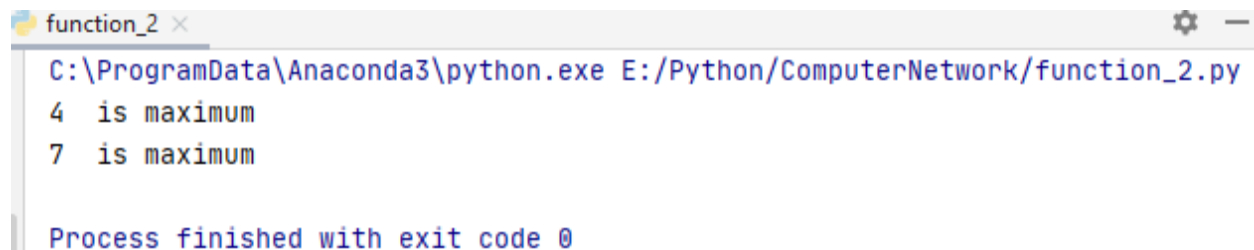
```
function ×  
C:\ProgramData\Anaconda3\python.exe E:/Python/ComputerNetwork/function.py  
hello world  
  
Process finished with exit code 0
```

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

##### Code :

```
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)  
    x = 5  
    y = 7  
    print_max(x,y)
```

##### output :

A terminal window titled 'function\_2' with a close button, a settings gear icon, and a minus icon. It shows the command 'C:\ProgramData\Anaconda3\python.exe E:/Python/ComputerNetwork/function\_2.py' being executed. The output is '4 is maximum' on one line and '7 is maximum' on the next line. At the bottom, it says 'Process finished with exit code 0'.

```
function_2 × ⚙️ —  
C:\ProgramData\Anaconda3\python.exe E:/Python/ComputerNetwork/function_2.py  
4 is maximum  
7 is maximum  
  
Process finished with exit code 0
```

## Section 4.1: Python function variables and modules

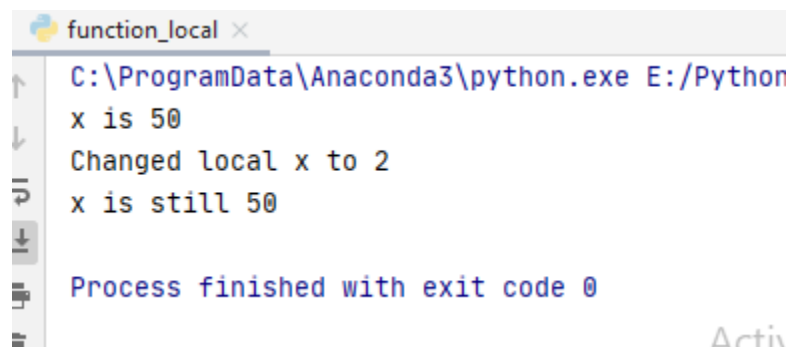
### Exercise 4.1.4: Local variable (save as function\_local.py)

#### Code:

```
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)
```

#### output :



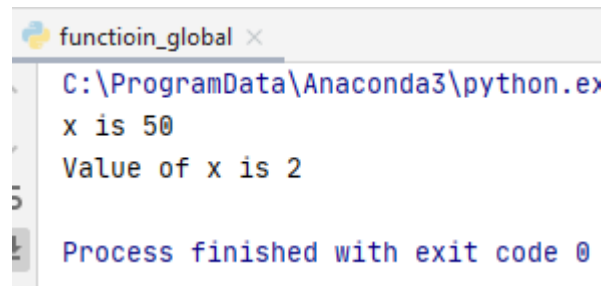
```
function_local x
C:\ProgramData\Anaconda3\python.exe E:/Python
x is 50
Changed local x to 2
x is still 50
Process finished with exit code 0
```

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

#### Code :

```
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)
```

**output :**



```
function_global ×
C:\ProgramData\Anaconda3\python.exe
x is 50
Value of x is 2
Process finished with exit code 0
```

**Exercise 4.1.6:** Python modules.

mymodule.py

**code :**

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

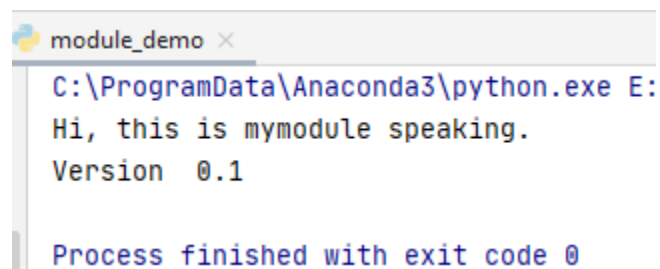
module\_demo.py

code:

```
import mymodule
```

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

**Output :**



```
module_demo ×
C:\ProgramData\Anaconda3\python.exe E:
Hi, this is mymodule speaking.
Version 0.1
Process finished with exit code 0
```

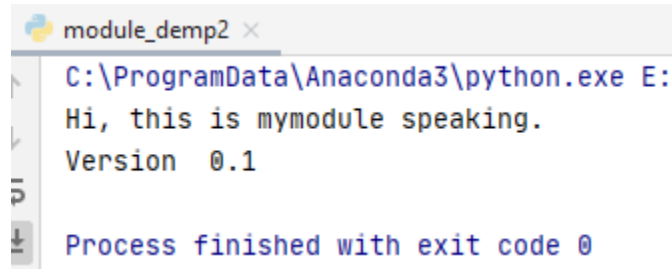
module\_demp2.py

**code :**

```
from mymodule import say_hi,__version__

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

**output :**



```
module_demp2 x
C:\ProgramData\Anaconda3\python.exe E:
Hi, this is mymodule speaking.
Version 0.1
Process finished with exit code 0
```

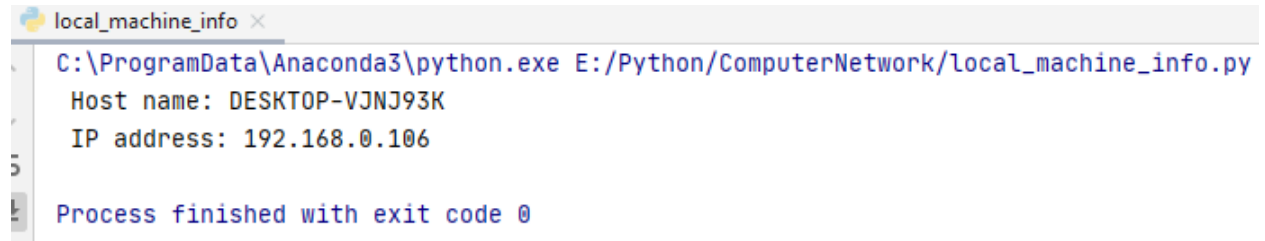
## Section 4.2: Sockets, IPv4, and Simple Client/Server Programming

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

**Code :**

```
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()
```

**Output :**



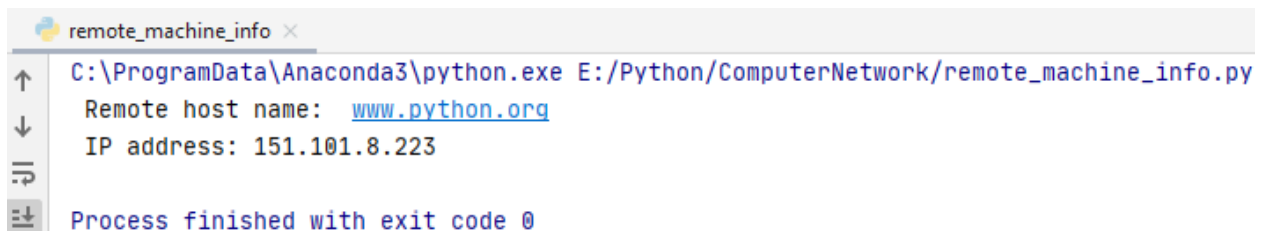
```
local_machine_info x
C:\ProgramData\Anaconda3\python.exe E:/Python/ComputerNetwork/local_machine_info.py
Host name: DESKTOP-VJNJ93K
IP address: 192.168.0.106
Process finished with exit code 0
```

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

#### Code :

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

#### Output :



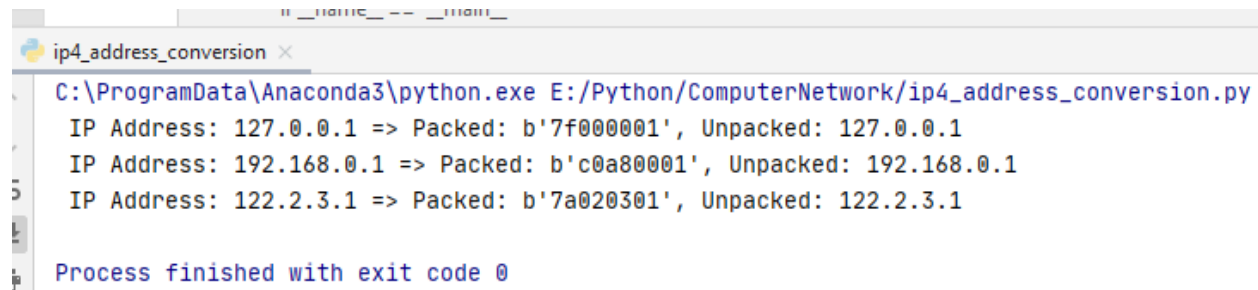
```
remote_machine_info x
C:\ProgramData\Anaconda3\python.exe E:/Python/ComputerNetwork/remote_machine_info.py
Remote host name: www.python.org
IP address: 151.101.8.223
Process finished with exit code 0
```

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

#### Code :

```
import socket
from binascii import hexlify
def convert_ip4_address():
    for ip_addr in ['127.0.0.1', '192.168.0.1', '122.2.3.1']:
        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()
```

## Output :



```
ip4_address_conversion ×
C:\ProgramData\Anaconda3\python.exe E:/Python/ComputerNetwork/ip4_address_conversion.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
IP Address: 122.2.3.1 => Packed: b'7a020301', Unpacked: 122.2.3.1

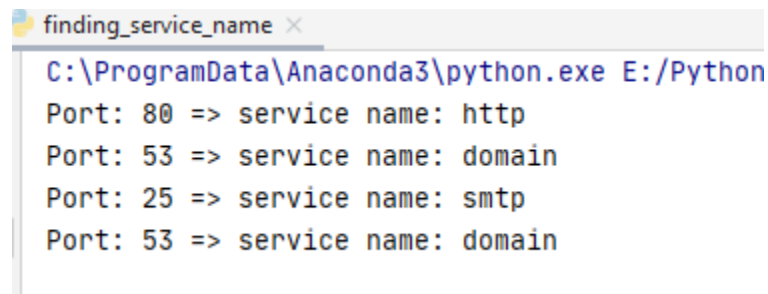
Process finished with exit code 0
```

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

### Code :

```
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()
```

### Output:



```
finding_service_name ×
C:\ProgramData\Anaconda3\python.exe E:/Python
Port: 80 => service name: http
Port: 53 => service name: domain
Port: 25 => service name: smtp
Port: 53 => service name: domain
```

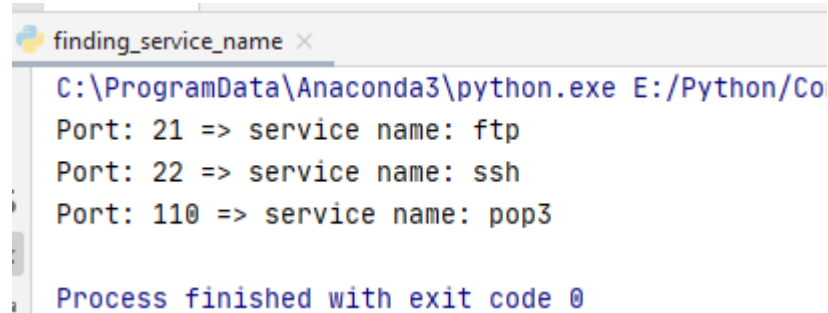
### Code :

```
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:



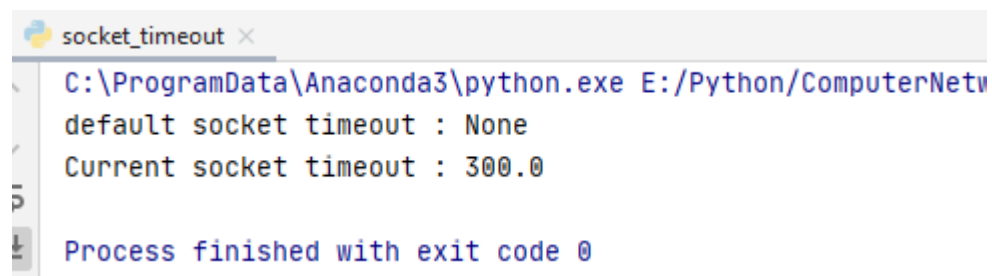
```
finding_service_name x  
C:\ProgramData\Anaconda3\python.exe E:/Python/Co  
Port: 21 => service name: ftp  
Port: 22 => service name: ssh  
Port: 110 => service name: pop3  
Process finished with exit code 0
```

### Exercise 4.2.5: Setting and getting the default socket timeout

#### Code :

```
import socket  
  
def test_socket_timeout():  
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
    print("default socket timeout : %s" % s.gettimeout())  
    s.settimeout(300)  
    print("Current socket timeout : %s" % s.gettimeout())  
  
if __name__ == '__main__':  
    test_socket_timeout()
```

### Output :



```
socket_timeout x  
C:\ProgramData\Anaconda3\python.exe E:/Python/ComputerNetv  
default socket timeout : None  
Current socket timeout : 300.0  
Process finished with exit code 0
```

#### Exercise 4.2.6: Writing a simple echo client/server application

##### Code :

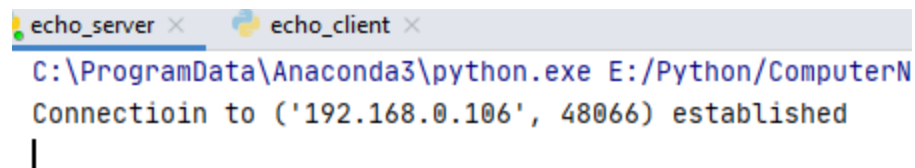
Echo\_server.py

```
import socket
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.bind((socket.gethostname(),1025))
s.listen(5)
while True:
    clt,adr = s.accept()
    print(f"Connectioin to {adr} established")
    clt.send(bytes("Socket Programmin in Python","utf-8"))
```

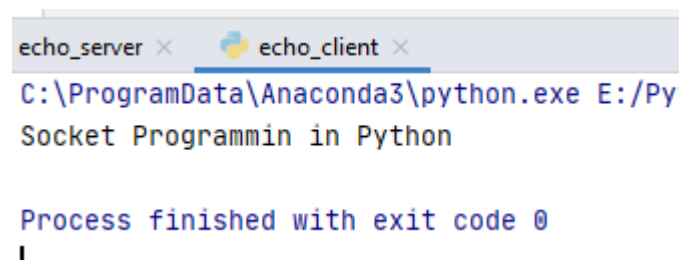
echo\_client

```
import socket
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((socket.gethostname(),1025))
msg = s.recv(1025)
print(msg.decode("utf-8"))
```

##### Output :



```
echo_server x echo_client x
C:\ProgramData\Anaconda3\python.exe E:/Python/ComputerN
Connectioin to ('192.168.0.106', 48066) established
|
```



```
echo_server x echo_client x
C:\ProgramData\Anaconda3\python.exe E:/Py
Socket Programmin in Python

Process finished with exit code 0
|
```

**Conclusion :** In this lab we have learned the basic concept of python programming and socket programming in python. In Python we have learned how to define a function, local and global variables. Using the Python program, we learned how to recover our machine's hostname, IP address, and how to recover the IP address of a remote machine. We know how to connect client-side and server-side using the Python program.

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++.

Sockets and the socket API are used to send messages across a network. They provide a form of inter-process communication (IPC). The network can be a logical, local network to the computer, or one that's physically connected to an external network, with its own connections to other networks.

Python's socket module provides an interface to the Berkeley sockets API. This is the module that we'll use and discuss in this tutorial.

The primary socket API functions and methods in this module are:

- `socket()`
- `bind()`
- `listen()`
- `accept()`
- `connect()`
- `connect_ex()`
- `send()`
- `recv()`
- `close()`

Network devices (for example, routers and switches), have finite bandwidth available and their own inherent system limitations. They have CPUs, memory, buses, and interface packet buffers, just like our clients and servers. TCP relieves us from having to worry about packet loss, data arriving out-of-order, and many other things that invariably happen when you're communicating across a network.