MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY

LAB-REPORT

Report no : 05

Report Name: Programming with Python Lab

Course code : ICT-3208

Course title : Network Planning and Designing Lab

Date of Submission : 15-Sept-2020

Submitted By

Name: Md. Nayan Ali

ID: IT-16062

3rd year 2nd semester

Session: 2015-2016

Dept. of ICT

MBSTU.

Submitted To

Mr. Nazrul Islam

Assistant Professor,

Dept. of ICT, MBSTU.

Objectives:

i. Understand how python function works
ii. Understand the use of global and local variables
iii. Understand how python modules works
iv. Learning the basis of networking programing with python

Programming with Python Lab

Exercises:

1. Python function (save as function.py).

Source Code:

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

Output:

```
hello world
```

2. Python function (save as function_2.py)

Source 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)
# directly pass literal values
x = 5
y = 7
# pass variables as arguments
print_max(x, y)
```

Output:

```
4 is maximum
7 is maximum
```

3. Local variable (save as function_local.py).

Source Code:

```
def 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:

```
x is 50
Changed global x to 2
Value of x is 2
```

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

Source Code:

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

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

Source Code:

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

Output:

```
Hi, this is mymodule speaking.
Version 0.1
```

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

Source Code:

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

Output:

```
Hi, this is mymodule speaking.
Version 0.1
```

7. Printing your machine's name and IPv4 address. Create python scrip using the syntax provided below (save as local_machine_info.py):

Source 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:

```
Host name: DESKTOP-8GD1EFD
IP address: 192.168.56.1
```

8. Retrieving a remote machine's IP address. Create python scrip using the syntax provided below (save as remote_machine_info.py):

Source Code:

```
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()
```

Output:

```
Remote host name: www.python.org
IP address: 151.101.8.223
```

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

Source Code:

```
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()
```

Output:

```
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
```

10. Finding a service name, given the port and protocol. Create python scrip using the syntax below (save as finding_service_name.py):

Source 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:

```
Port: 80 => service name: http
Port: 25 => service name: smtp
Port: 53 => service name: domain
```

11. Setting and getting the default socket timeout. Create python scrip using the syntax below (save as socket_timeout.py):

Source 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(100)
print ("Current socket timeout: %s" %s.gettimeout())
if __name__ == '__main__':
test_socket_timeout()
```

Output:

```
Default socket timeout: None
Current socket timeout: 100.0
```

12. Writing a simple echo client/server application (Tip: Use port 9900). Create python scrip using the syntax below (save as echo_server.py):

Source Code:

```
import socket
import sys
import argparse
import codecs
from codecs import encode, decode
host = 'localhost'
data_payload = 8088
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
connecti
ons
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()
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()
port = given_args.port
echo_server(port)
```

Output: I can'nt fix the error.

13. Create python scrip using the syntax below (save as echo_client.py):

Source Code:

```
#!/usr/bin/env python
import socket
import sys
import argparse
import codecs
from codecs import encode, decode
host = 'localhost'
def echo_client(port):
""" A simple echo client """
# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Connect the socket to the server
server_address = (host, port)
print ("Connecting to %s port %s" % server_address)
sock.connect(server_address)
# Send data
try:
# Send data
message = "Test message: SDN course examples"
print ("Sending %s" % message)
sock.sendall(message.encode('utf_8'))
# Look for the response
amount_received = 0
amount_expected = len(message)
while amount_received < amount_expected:
data = sock.recv(16)
amount_received += len(data)
```

```
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 Example')
parser.add_argument('--
port', action="store", dest="port", type=int,required=True)
given_args = parser.parse_args()
port = given_args.port
echo_client(port)
```

Output: I can'nt fix the error.

Conclusion: From this lab, I've learned that the functionality of python. Also learn the use of Module that can be reused a number of functions in other programs. Networking background for socket and TCP, UDP with python are the most important part of the Lab.