Mawlana Bhashani Science and Technology University

Lab-Report

Report No: 02

Course code: ICT-3207

Course title: Computer network lab

Date of Performance:

Date of Submission:

Submitted by

NAME:Ruku Shikdeer &Tharima Akter

ID : IT18057 &IT18058

3rd year 1st semester

Session: 2017-2018

Dept. of ICT

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Lab Report No. : 02

Lab Report Name: Programming with python.

Theory:

Python functions: 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.

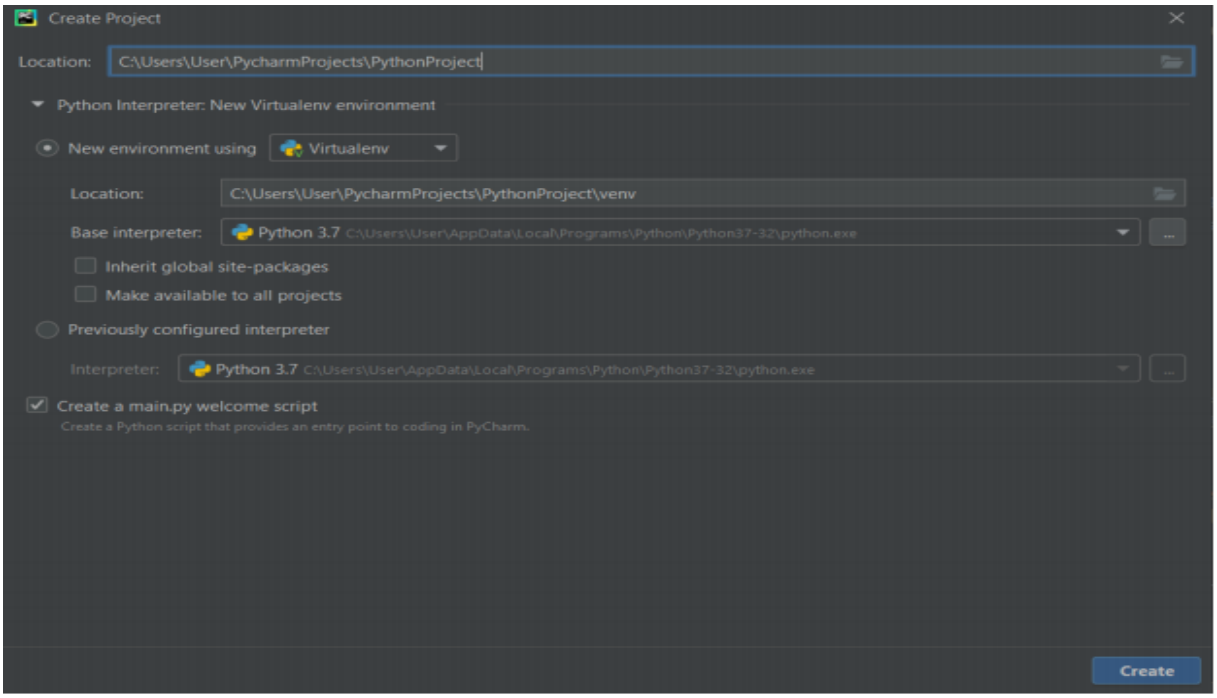
Local Variables: Variables declared inside a function definition are not related in any way to other variables with the same names used outside the function (variable names are local to the function). This is called the scope of the variable. All variables have the scope of the block they are declared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

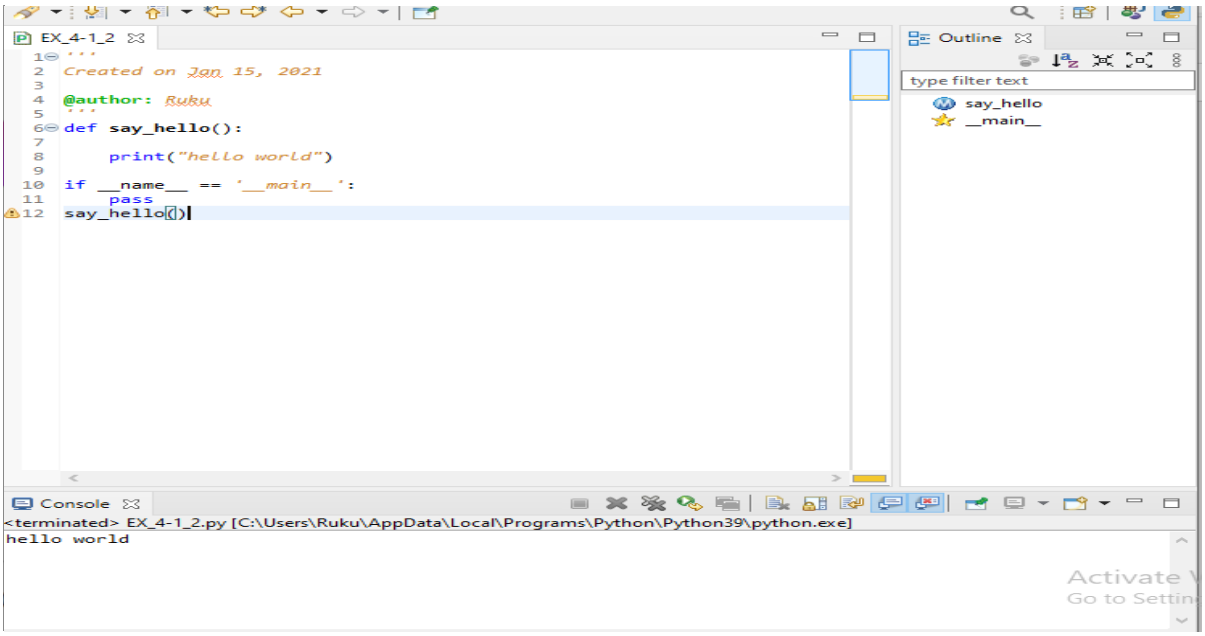
Modules: Modules allow reusing a number of functions in other programs.

Exercises:

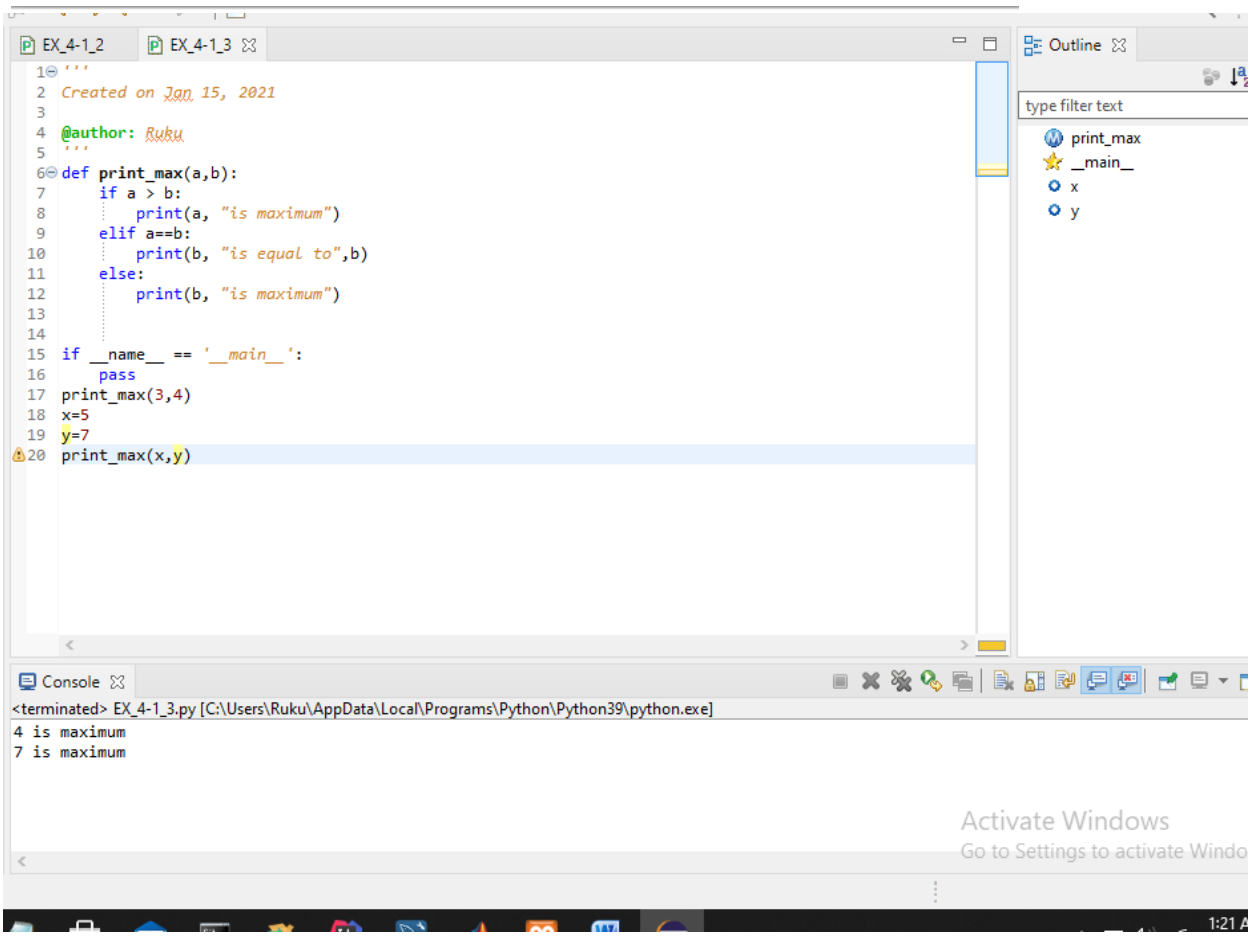
Exercise 4.1.1: Create a Python project with SDN-LAB.



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



Exercise 4.1.3: Python function (save as function_2.py)



Exercise 4.1.4: Local variable

```python
'''
Created on Jan 15, 2021

@author: Ruku
'''
x=50
def func(x):
    print("changed local x to",x)

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

```
Console
<terminated> EX_4-1_4.py [C:\Users\Ruku\AppData\Local\Programs\Python\Python39\python.exe]
changed local x to 50
x is still 50
```

Exercise 4.1.5: Global variable

```python

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()
```

```
Console
<terminated> finding_service_name.py [C:\Users\Ruku\AppData\Local\Programs\Python\Python39\python.exe]
Port: 80 => service name: http
Port: 25 => service name: smtp
Port: 53 => service name: domain
```

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

```python
import socket
hostname = socket.gethostname()
IPAddr = socket.gethostbyname(hostname)
print("Your Computer Name is:" + hostname)
print("Your Computer IP Address is:" + IPAddr)
```

Console

```
terminated> EX_4-2_1.py [C:\Users\Ruku\AppData\Local\Programs\Python\Python39\python.exe]
our Computer Name is:DESKTOP-VPK2KQC
our Computer IP Address is:192.168.0.114
```

Exercise 4.2.2: Retrieving a remote machine's IP address

```python
import socket
hostname = socket.gethostname()
IPAddr = socket.gethostbyname(hostname)
print("Your Computer Name is:" + hostname)
print("Your Computer IP Address is:" + IPAddr)
```

```
Console
terminated> EX_4-2_1.py [C:\Users\Ruku\AppData\Local\Programs\Python\Python39\python.exe]
our Computer Name is:DESKTOP-VPK2KQC
our Computer IP Address is:192.168.0.114
```

Exercise 4.2.3: Converting an IPv4 address to different formats.

```python
import socket
hostname = socket.gethostname()
IPAddr = socket.gethostbyname(hostname)
print("Your Computer Name is:" + hostname)
print("Your Computer IP Address is:" + IPAddr)
```

```
Console
terminated> EX_4-2_1.py [C:\Users\Ruku\AppData\Local\Programs\Python\Python39\python.exe]
our Computer Name is:DESKTOP-VPK2KQC
our Computer IP Address is:192.168.0.114
```

```python

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()
```

```
<terminated> finding_service_name.py [C:\Users\Ruku\AppData\Local\Programs\Python\Python39\python.exe]
Port: 80 => service name: http
Port: 25 => service name: smtp
Port: 53 => service name: domain
```

Exercise 4.2.5: Setting and getting the default socket timeout.

```python


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()

```

```
<terminated> sockettimeout.py [C:\Users\Ruku\AppData\Local\Programs\Python\Python39\python.exe]
Default socket timeout: None
Current socket timeout: 100.0
```

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

Server code:

```python
import socket
import sys
import argparse
import codecs
from codecs import encode, decode
host='localhost'
data_payload=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(f"Starting up echo server on {server_address} port {sock.bind(server_address)}")
    sock.listen(backlog)
    while True:
        print("Willing to receive message from client")
    client,address = sock.accept()
    data = client.recv(data_payload)
    if data:
        print(f"Data: {data}")
        client.send(data)
        print(f"sent {data} bytes back to {address}")
        client.close();

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)
```

Client code:

```python
import socket
import sys
import argparse
import codecs
from codecs import encode, decode
host='localhost'
data_payload=2048
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(f"Starting up echo server on {server_address} port {server_address}")
    sock.bind(server_address)
    sock.listen(backlog)
    while True:
        print("Willing to receive message from client")
    client,address = sock.accept()
    data = client.recv(data_payload)
    if data:
        print(f"Data: {data}")
        client.send(data)
        print(f"sent {data} bytes back to {address}")
        client.close();

parser = argparse.ArgumentParser(description="Computer Network lab")
parser.add_argument('--port',action="store",dest="port",type=int,required=True)
given_args = parser.parse_args()
port = given_args.port
echo_server(port)
```

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