



Mawlana Bhashani Science and Technology University

Lab-Report

Report No:02

Course code: ICT-3208

Course title: Network Planning and Designing Lab

Date of Performance:14.01.2021

Date of Submission:16.01.2021

Submitted By

Name:Kallol Devnath

ID:IT-18029

Name:Nandon Kumar Pal

ID:IT-18022

3rd year 2nd semester

Session: 2017-2018

Dept. of ICT

MBSTU.

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

1. Objectives

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

2. 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.

Networking background for sockets

What is a socket and how use it? 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.

1.On the server-side: The server just waits, listening to the socket for a client to make a connection request.

2.On the client-side: The client knows the hostname of the machine on which the server is running and the port number on which the server is listening. To make a connection request, the client tries to rendezvous with the server on the server's machine and port. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection. This is usually assigned by the system

TCP: 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. The protocol builds up a connection prior to data transfer using a system called a three-way handshake. This is a way for the two ends of the communication to acknowledge the request and agree upon a method of ensuring data reliability. After the data has been sent, the connection is torn down using a similar four-way handshake. TCP is the protocol of choice for many of the most popular uses for the internet, including WWW, FTP, SSH, and email. It is safe to say that the internet we know today would not be here without TCP.

UDP: UDP stands for user datagram protocol. It is a popular companion protocol to TCP and is also implemented in the transport layer. The fundamental difference between UDP and TCP is that UDP offers unreliable data transfer. It does not verify that data has been received on the other end of the connection. This might sound like a bad thing, and for many purposes, it is. However, it is also extremely important for

some functions. Because it is not required to wait for confirmation that the data was received and forced to resend data, UDP is much faster than TCP. It does not establish a connection with the remote host, it simply fires off the data to that host and doesn't care if it is accepted or not. Because it is a simple transaction, it is useful for simple communications like querying for network resources. It also doesn't maintain a state, which makes it great for transmitting data from one Page | 16 SDN-Labs machine to many real-time clients. This makes it ideal for VOIP, games, and other applications that cannot afford delays.

3. Methodology

Defining functions:

Functions are defined using the def keyword. After this keyword comes an identifier name for the function, followed by a pair of parentheses which may enclose some names of variables, and by the final colon that ends the line

```
def XX_YY(variable1, variable2):
```

```
# block belonging to the function
```

```
# End of function
```

Defining local and global variables:

Local and global variables can be defined using:

```
x = 50 #Local global x
```

Defining modules:

There are various methods of writing modules, but the simplest way is to create a file with a .py extension that contains functions and variables.

```
def xx_yy(): aa
```

Using modules:

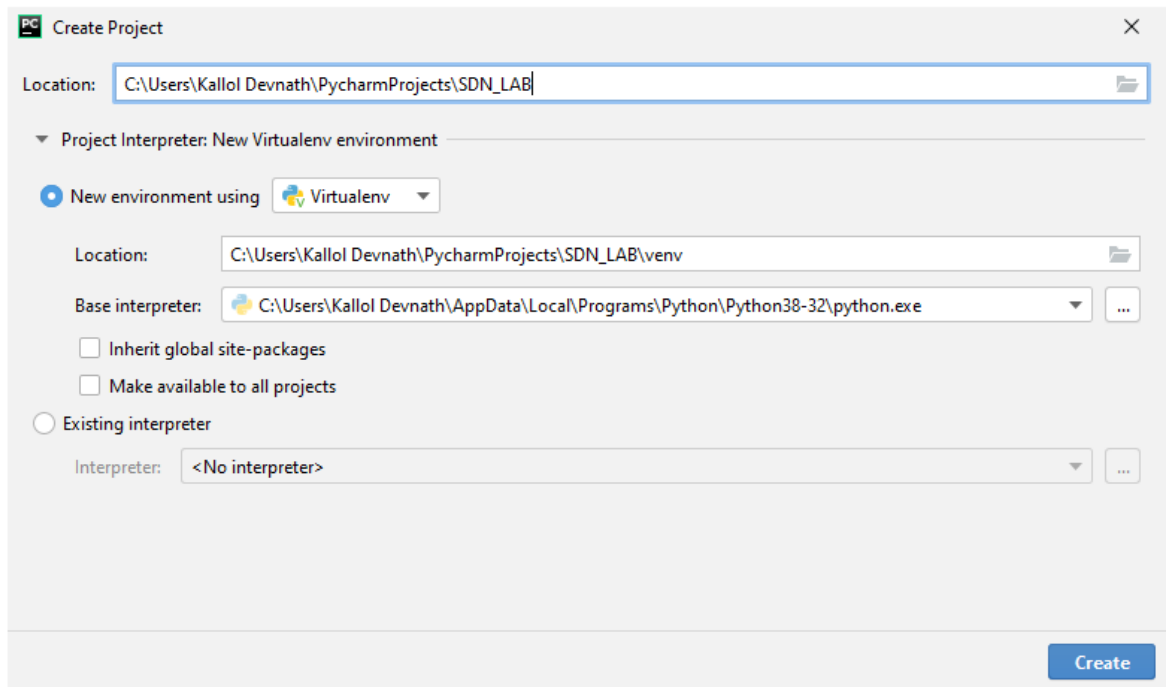
A module can be imported by another program to make use of its functionality. This is how we can use the Python standard library as well. `import xx_yy`

4. Exercises

Exercise 4.1.1: Create a python project using with SDN_LAB

Output:

.



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

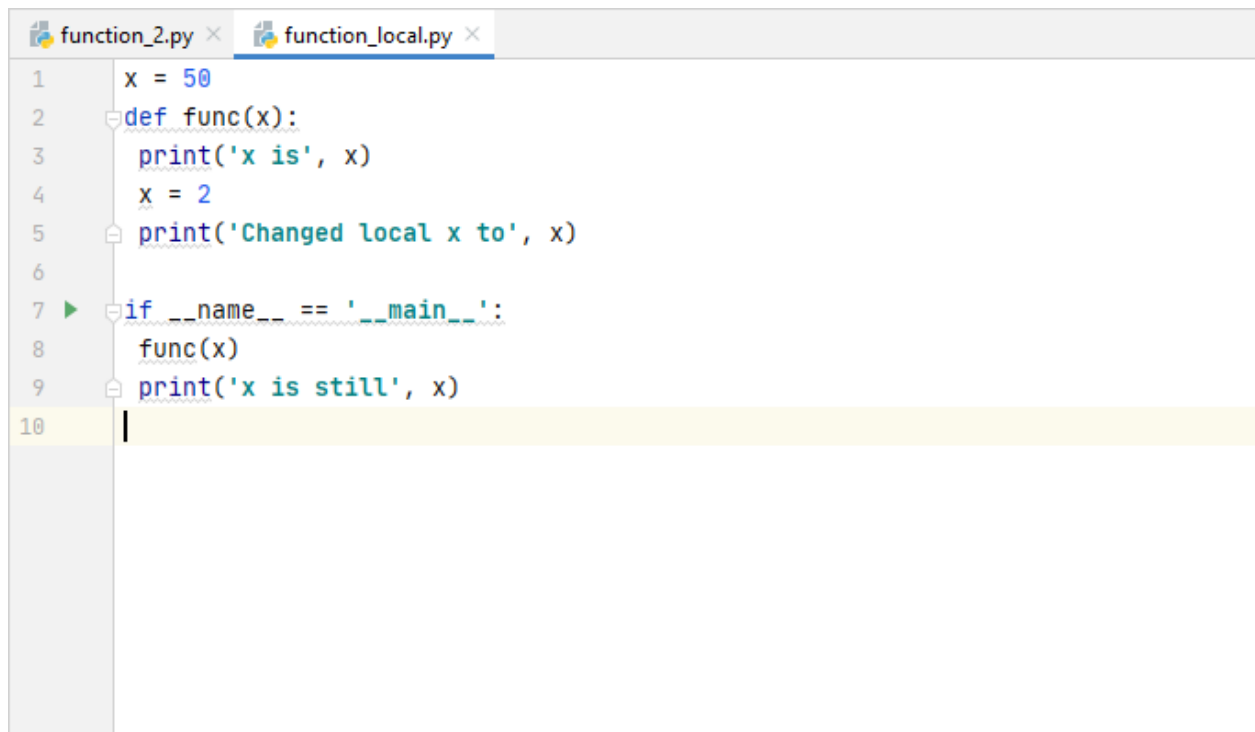
```
function.py x
1 def say_hello():
2     # block belonging to the function
3     print('hello world')
4     # End of function
5 i __name__ == '__main__':
6     say_hello() # call the function
```

Output:

```
Run: function x
" C:\Users\Kallol Devnath\PycharmProjects\SDN_LA
hello world

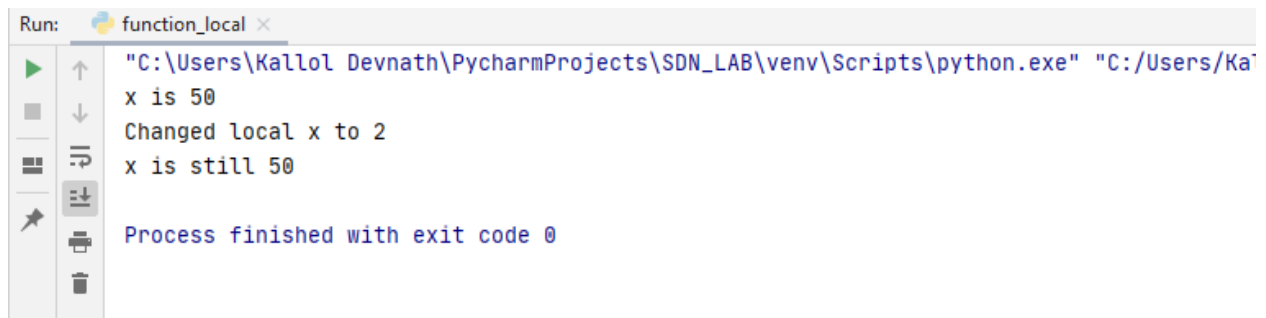
Process finished with exit code 0
```

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



```
1 x = 50
2 def func(x):
3     print('x is', x)
4     x = 2
5     print('Changed local x to', x)
6
7 if __name__ == '__main__':
8     func(x)
9     print('x is still', x)
10
```

Output:



```
Run: function_local x
"C:\Users\Kallol Devnath\PycharmProjects\SDN_LAB\venv\Scripts\python.exe" "C:/Users/Ka
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)

```
as function_global.py x
1 x = 50
2 def func():
3     global x
4     print('x is', x)
5     x = 2
6     print('Changed global x to', x)
7 if __name__ == '__main__':
8     func()
9     print('Value of x is', x)
```

Output:

```
Run: as function_global x
"C:\Users\Kallol Devnath\PycharmProjects\SDN_LAB\venv\Scripts\python.exe" "C:/Users/Kal
x is 50
Changed global x to 2
Value of x is 2

Process finished with exit code 0
```