

Lab-Report

Report No:02

Course code: ICT-3208

Course title: programing with python.

Date of Performance:15-01-21

Date of Submission:16-01-21

Submitted by

Name: Md.Abdullah Al Mamun

ID:IT-18040

3th year 2nd semester

Session: 2017-2018

Dept. of ICT

MBSTU.

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

4.1: Python function variables and modules.

Answer

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.

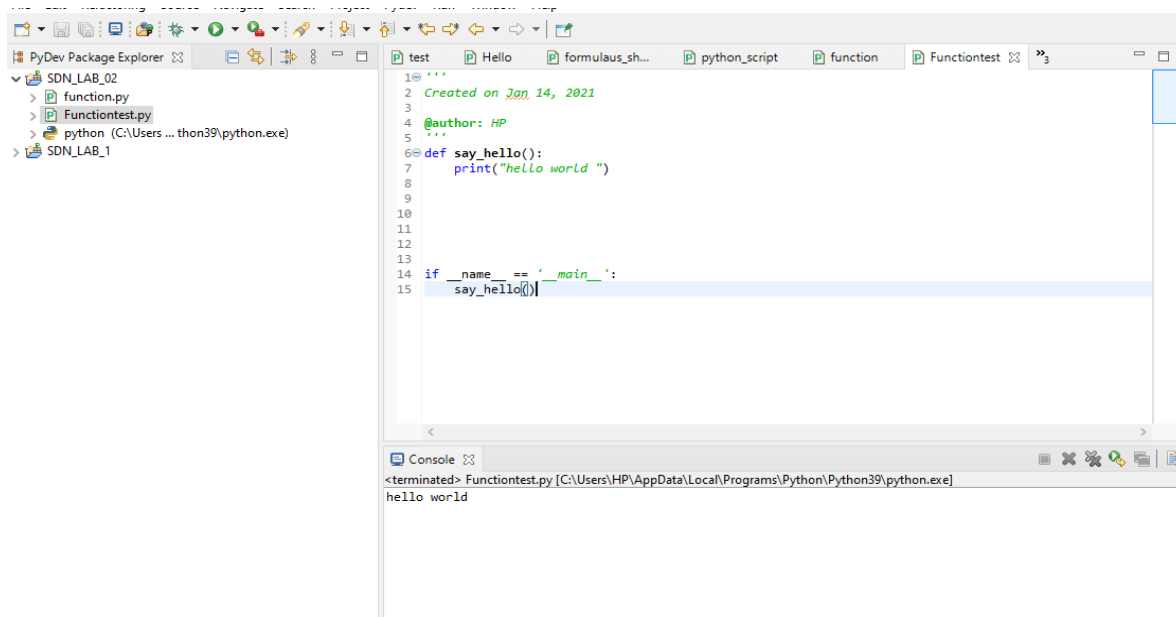
The global statement : Variables defined at the top level of the program are intended global. Global variables are intended to be used in any functions or classes). Global statement allows defining global variables inside functions as well.

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

4.1.1: Create a python project using with SDN_LAB.

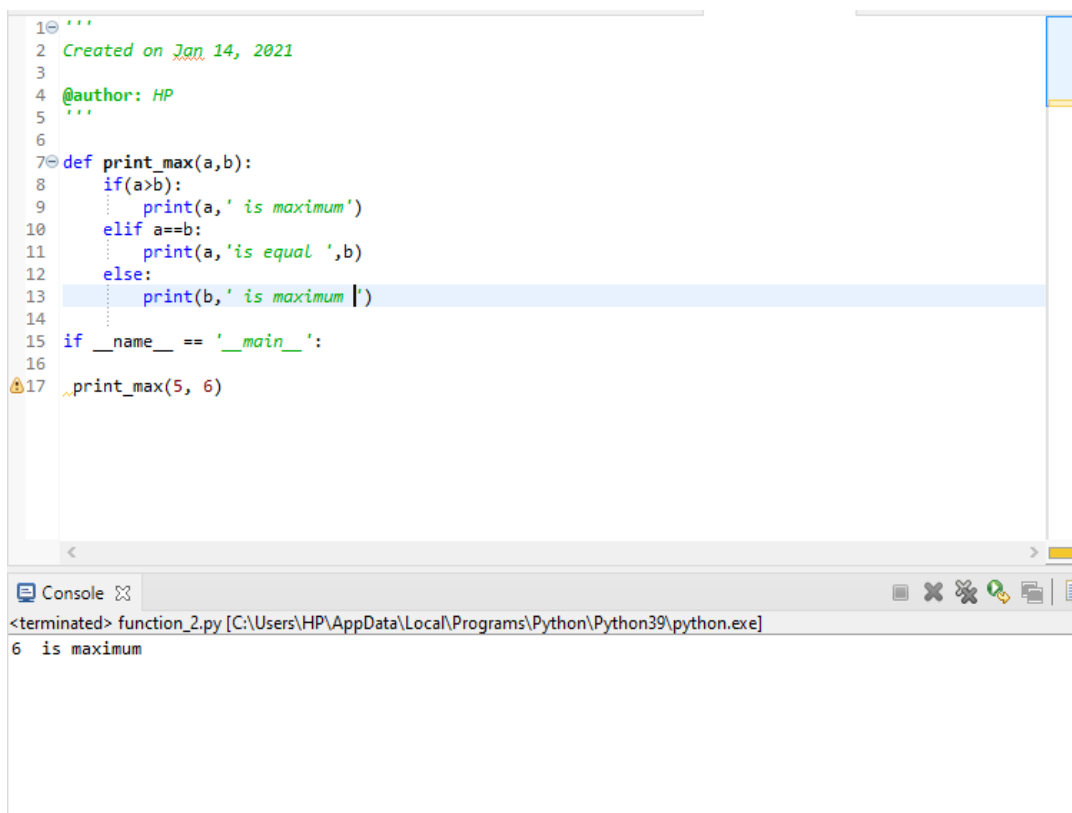
4.1.2: Python function (save as function.py).

Answer



4.1.3: Python function (save as function_2.py)

Answer:



4.1.4: Local variable (save as function_local.py)

Answer

```
1'''
2Created on Jan 14, 2021
3
4@author: HP
5'''
6
7x=50
8def fun(x):
9    print(' x is ',x)
10    x=2
11    print('change the value of x is ',x)
12
13
14
15
16
17
18if __name__ == '__main__':
19    pass
20    fun(x)
21    print('still x is ',x)
```

Console

```
<terminated> function_local.py [C:\Users\HP\AppData\Local\Programs\Python\Python39\python.exe]
x is 50
change the value of x is 2
still x is 50
```

Final value of is 50. The value of x not change because the value of x is change inside the function which is locally change. So finally the value of x is not change.

4.1.5: Global variable (save as function_global.py)

Answer

```
1 '''
2 Created on Jan 14, 2021
3
4 @author: HP
5 '''
6
7
8
9 x=50
10
11 def fun():
12     global x
13
14     print(' x is ',x)
15     x=2
16     print('changed the value of x is ',x)
17
18 if __name__ == '__main__':
19     pass
20 fun()
21 print('the value of x is ',x)
```

Console

```
<terminated> Function_Global.py [C:\Users\HP\AppData\Local\Programs\Python\Python39\python.exe]
x is 50
changed the value of x is 2
the value of x is 2
```

The final value of x is 2. This time the value of x is change because x declar globally in the function.

Exercise 4.1.6: Python modules

Answer

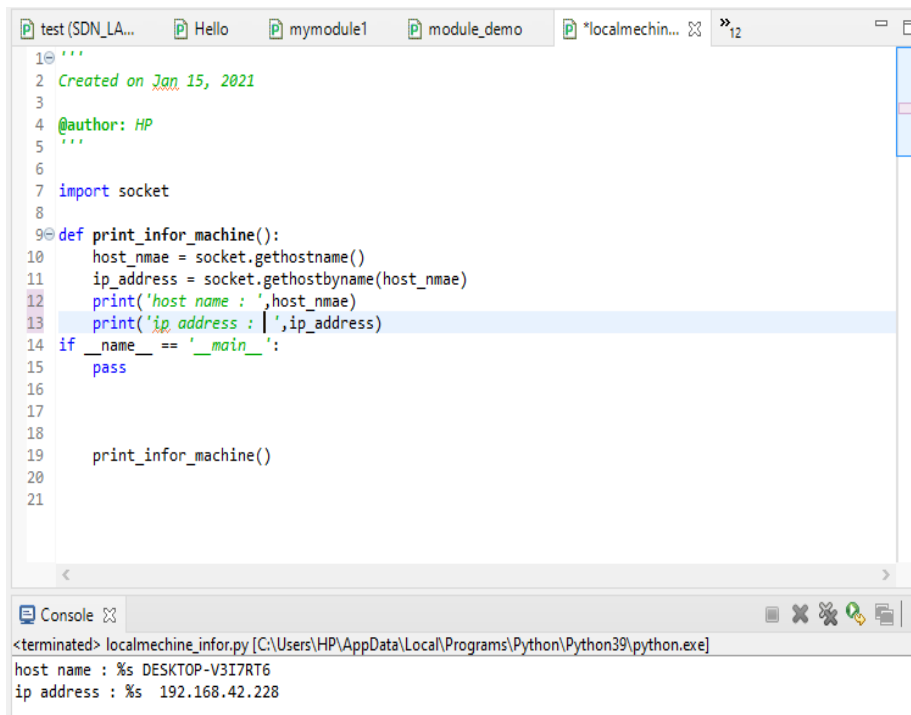
```
1 '''
2 Created on Jan 14, 2021
3
4 @author: HP
5 '''
6 import mymodule
7
8 def say_hi():
9
10     print('this is mymodule speaking ')
11     _version_='0.1'
12
13 if __name__ == '__main__':
14     pass
15 say_hi()
```

Console

```
<terminated> mymodule1.py [C:\Users\HP\AppData\Local\Programs\Python\Python39\python.exe]
this is mymodule speaking
```

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

Answer



The screenshot shows a Python IDE with a file named `localmachine_infor.py`. The code defines a function `print_infor_machine()` that uses the `socket` module to get the host name and IP address. The function is called at the bottom of the script. The console output shows the host name as `DESKTOP-V3I7RT6` and the IP address as `192.168.42.228`.

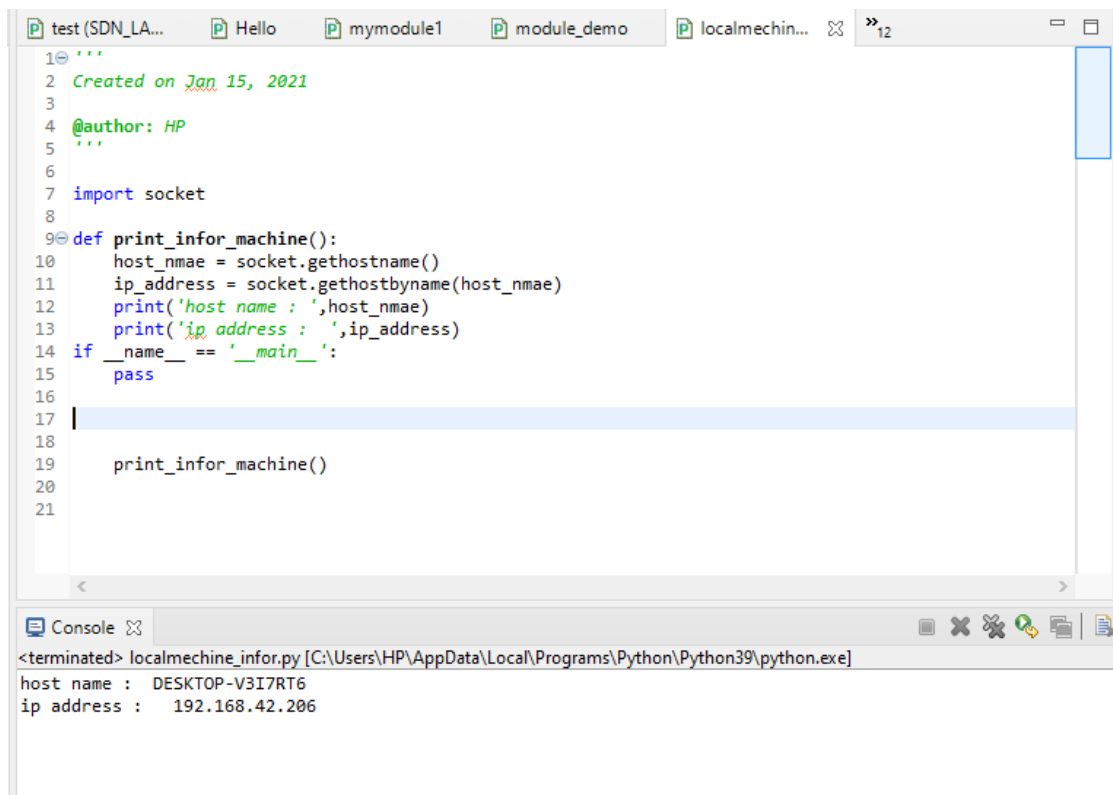
```
1'''  
2Created on Jan 15, 2021  
3  
4@author: HP  
5'''  
6  
7import socket  
8  
9def print_infor_machine():  
10    host_name = socket.gethostname()  
11    ip_address = socket.gethostbyname(host_name)  
12    print('host name : ',host_name)  
13    print('ip address : | ',ip_address)  
14if __name__ == '__main__':  
15    pass  
16  
17  
18  
19    print_infor_machine()  
20  
21
```

Console Output:

```
<terminated> localmachine_infor.py [C:\Users\HP\AppData\Local\Programs\Python\Python39\python.exe]  
host name : %s DESKTOP-V3I7RT6  
ip address : %s 192.168.42.228
```

4.2.2: Retrieving a remote machine's IP address .

Answer



The screenshot shows a Python IDE with a file named `localmachine_infor.py`. The script is as follows:

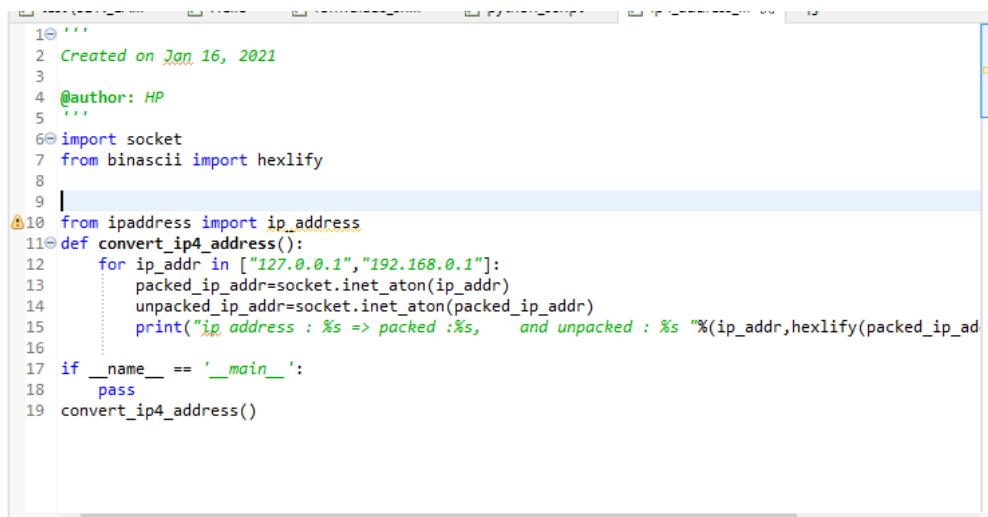
```
1'''
2Created on Jan 15, 2021
3
4@author: HP
5'''
6
7import socket
8
9def print_infor_machine():
10    host_nmae = socket.gethostname()
11    ip_address = socket.gethostbyname(host_nmae)
12    print('host name : ',host_nmae)
13    print('ip address : ',ip_address)
14if __name__ == '__main__':
15    pass
16
17
18
19    print_infor_machine()
20
21
```

The console output shows the execution results:

```
<terminated> localmachine_infor.py [C:\Users\HP\AppData\Local\Programs\Python\Python39\python.exe]
host name : DESKTOP-V3I7RT6
ip address : 192.168.42.206
```

Exercise 4.2.3: Converting an IPv4 address to different formats.

Answer



The screenshot shows a Python IDE with a script to convert IP addresses. The script is as follows:

```
1'''
2Created on Jan 16, 2021
3
4@author: HP
5'''
6
7import socket
8from binascii import hexlify
9
10from ipaddress import ip_address
11def convert_ip4_address():
12    for ip_addr in ["127.0.0.1", "192.168.0.1"]:
13        packed_ip_addr=socket.inet_aton(ip_addr)
14        unpacked_ip_addr=socket.inet_aton(packed_ip_addr)
15        print("ip address : %s => packed :%s, and unpacked : %s"%(ip_addr,hexlify(packed_ip_ad
16
17if __name__ == '__main__':
18    pass
19    convert_ip4_address()
```

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

Answer

```
1 '''
2 Created on Jan 16, 2021
3
4 @author: HP
5 '''
6 import socket
7
8 def find_service_name():
9     protocol_name='tcp'
10    for port in [80, 25]:
11
12    print("port : %s => service name : %s " %(port,socket.getservbyport(port,protocol_name)))
13
14    print("port : %s => service name : %s " %(53,socket.getservbyport(53,'udp')))
15
16
17 if __name__ == '__main__':
18     pass
19 find_service_name()
```

Console PyUnit

<terminated> ip4_address_conversion.py [C:\Users\HP\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.

Answer


```
1 '''  
2 Created on Jan 16, 2021  
3  
4 @author: HP  
5 '''  
6 import socket  
7 def test_socket_timeout():  
8     s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)  
9     print("default socket : %s" %s.gettimeout())  
10    s.settimeout(100)  
11    print("current timeout : %s" %s.gettimeout())  
12 if __name__ == '__main__':  
13     pass  
14 test_socket_timeout()  
15  
16
```

Console

```
<terminated> ip4_address_conversion.py [C:\Users\HP\AppData\Local\Programs\Python\Python39\python.exe]  
default socket : None  
current timeout : <built-in method gettimeout of socket object at 0x0000022C22473040>
```

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

Answer

```
test (SDN_LA...  Hello  formulaus_sh...  python_script  ip4_address_...  » 13

1 import socket
2 import sys
3 import argparse
4 import codecs
5 from codecs import encode, decode
6 host = 'localhost'
7 data_payload = 4096
8 backlog = 5
9 def echo_server(port):
10     """ A simple echo server """
11     # Create a TCP socket
12     sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
13     # Enable reuse address/port
14     sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
15
16     server_address = (host, port)
17     print ("Starting up echo server on %s port %s" %server_address)
18     sock.bind(server_address)
19
20     sock.listen(backlog)
21
22     while True:
23         print ("Waiting to receive message from client")
24         client, address = sock.accept()
25         data = client.recv(data_payload)
26
27         if data :
28
29             print ("Data: %s" %data)
```

```
22 while True:
23     print ("Waiting to receive message from client")
24     client, address = sock.accept()
25     data = client.recv(data_payload)
26
27     if data :
28
29         print ("Data: %s" %data)
30         client.send(data)
31         print ("sent %s bytes back to %s" % (data, address))
32 # end connection
33     client.close()
34
35 if __name__ == '__main__':
36
37     parser = argparse.ArgumentParser(description='Socket Server Example')
38     parser.add_argument("port", action="store", dest="port", type=int,
39 required=True)
40
41     given_args = parser.parse_args()
42     port = given_args.port
43     echo_server(9900)
```

5. Questions

Question 5.1: Explain in your own words which are the difference between functions and modules?

Answer:

This chapter is about *functions* and *modules* in Python. Functions are a handy way to isolate a particular part of your program's functionality and make it *reusable*. Modules are a way to collect a number of helpful functions in one file, which you can then use in multiple projects and share with other programmers.

We've already been using functions extensively in our programs; functions are one of the main building blocks of Python programming. Whenever you've typed something like `len(x)` or `type(y)` or even `random.choice([1, 2, 3])`, you've been using functions. It's just that these functions come pre-defined by Python. In this chapter, we're going to learn how to write our own functions.

Question 5.2: Explain in your own words when to use local and global variables?

Answer

Local variables can be used only by statements that are inside that function or block of code. Local variables are not known to functions on their own. Global variables are defined outside of all the functions, usually on top of the program. The global variables will hold their value throughout the lifetime of your program.

Question 5.3: Which is the role of sockets in computing networking? Are the sockets defined random or there is a rule?

Answer:

Socket in Computer Network

A **socket** is one endpoint of a **two way** communication link between two programs running on the network. The socket mechanism provides a means of inter-process communication (IPC) by establishing named contact points between which the communication take place. Like 'Pipe' is used to create pipes and sockets is created using '**socket**' system call. The socket provides bidirectional **FIFO** Communication facility over the network. A socket connecting to the network is created at each end of the communication. Each socket has a specific address. This address is composed of an IP address and a port number. Socket are generally employed in client server applications. The server creates a socket, attaches it to a network port addresses then waits for the client to contact it.

The client creates a socket and then attempts to connect to the server socket. When the connection is established, transfer of data takes place.

There is no socket defined to accept only the Europlug. Instead, the Europlug fits a range of sockets in common use in Europe. These sockets, including the CEE 7/1, CEE 7/3 (German/"Schuko"), CEE 7/5 (French), and most Israeli, Swiss, Danish and Italian sockets, were designed to accept pins of various diameters, mainly 4.8 mm, but also 4.0 mm and 4.5 mm, and are usually fed by final circuits with either 10 A or 16 A overcurrent protection devices

Question 5.4: Why is relevant to have the IPv4 address of remote server? Explain what is

Domain Name System (DNS)?

Answer

In Type the public name or IPv4 address used by clients to connect to the Remote Access server, enter the public name for the deployment (this name matches the subject name of the IP-HTTPS certificate, for example, edge1.contoso.com)

Domain Name System (DNS)

The Domain Name System (DNS) maps human-readable domain names (in URLs or in email address) to IP addresses. For example, DNS translates and maps the domain freecodecamp.org to the IP address 104.26.2.33.

To help you fully understand this description, this section details:

Question 5.5: Create a program that allows exchange messages increased by the user between client and server.

Answer:

Client-Server Model

The client-server model, or client-server architecture, is a distributed application framework dividing tasks between servers and clients, which either reside in the same system or communicate through a computer network or the Internet. The client relies on sending a request to another program in order to access a service made available by a server. The server runs one or more programs that share resources with and distribute work among clients.

The client server relationship communicates in a request–response messaging pattern and must adhere to a common communications protocol, which formally defines the rules, language, and dialog patterns to be used. Client-server communication typically adheres to the TCP/IP protocol suite.

TCP protocol maintains a connection until the client and server have completed the message exchange. TCP protocol determines the best way to distribute application data into packets that networks can deliver, transfers packets to and receives packets from the network, and manages flow control and retransmission of dropped or garbled packets. IP is a connectionless protocol in which each packet traveling through the Internet is an independent unit of data unrelated to any other data units.