

Read Me

This document contains the format termed as “Indicator Script” using which the teams indicate the pickup and delivery of Packages.

Step 1: Download **Package.zip** from the resource section of e-Yantra portal.

Package.zip contains two folders:

1. **Package_server**
2. **Package_client**

The folder “Package_server” contains two files:

1. **package_server.py**
2. **setup.py**

The folder “Package_client” contains two files:

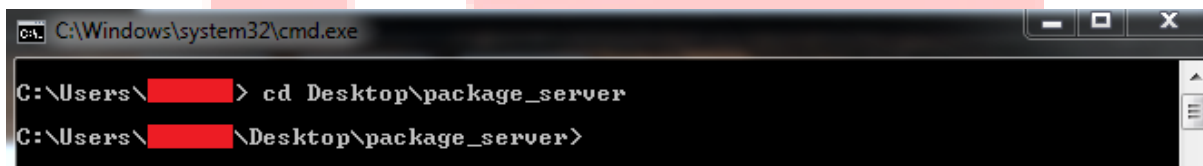
1. **package_client.py**
2. **setup.py**

Copy (i) the folder “Package_server” to the PC/Laptop used to login to the Raspberry Pi and (ii) the folder “Package_client” to the Raspberry Pi using any storage device.

The PC/Laptop will act as a **server** and the Raspberry Pi will act as a **client**.

Steps to be followed on the PC/Laptop:

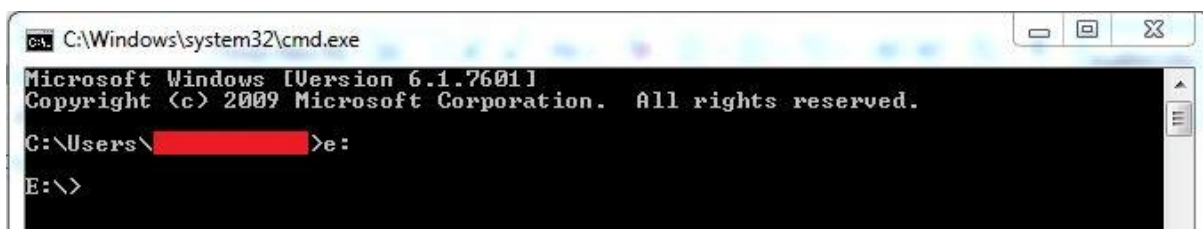
Step 2: Open command prompt on the PC/Laptop and change directory to the folder containing package_server.py using the **cd** (change directory) command.



```
C:\Windows\system32\cmd.exe
C:\Users\[redacted]> cd Desktop\package_server
C:\Users\[redacted]\Desktop\package_server>
```

Figure 1: Changing directory in PC/Laptop

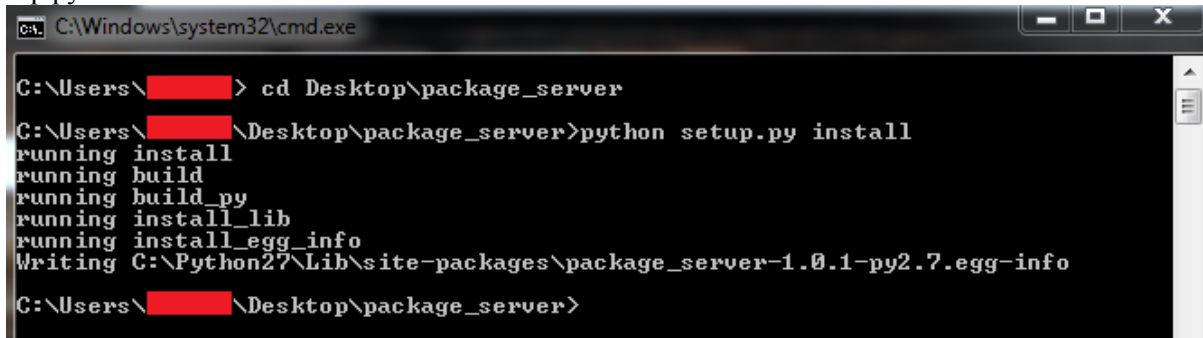
Note: It may happen that you choose to store the folder in a directory other than the C drive. To change the drive from C:\ to any other drive (Example, E:\) you simply need to write “drive_name:”.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\[redacted]>e:
E:\>
```

Figure 2: Changing drive

Step 3: Installing package_server.py. This is done using the script setup.py using the command “python setup.py install”.



```
C:\Windows\system32\cmd.exe

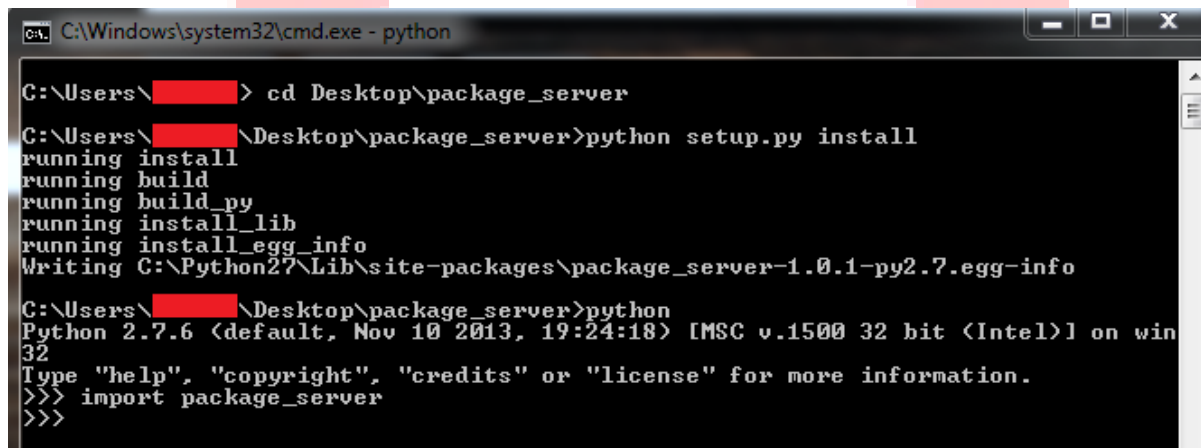
C:\Users\> cd Desktop\package_server

C:\Users\Desktop\package_server>python setup.py install
running install
running build
running build_py
running install_lib
running install_egg_info
Writing C:\Python27\Lib\site-packages\package_server-1.0.1-py2.7.egg-info

C:\Users\Desktop\package_server>
```

Figure 3: Installing package_server.py on PC/Laptop

Step 4: Check whether the installation is complete by opening “python” in cmd. Next write the command “import package_server”. Figure 4 shows the command line output in case of successful installation.



```
C:\Windows\system32\cmd.exe - python

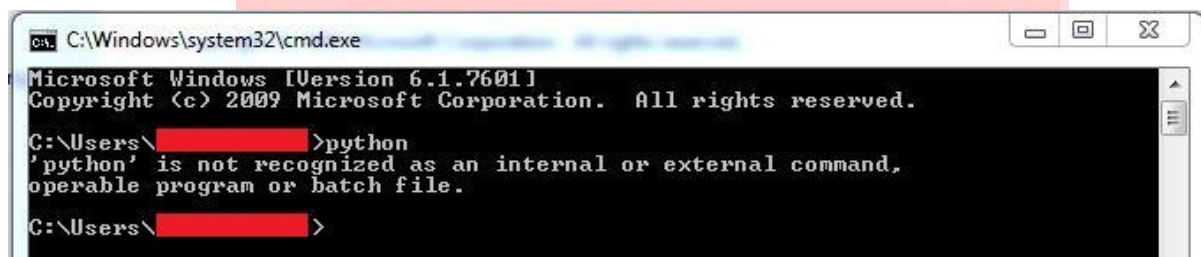
C:\Users\> cd Desktop\package_server

C:\Users\Desktop\package_server>python setup.py install
running install
running build
running build_py
running install_lib
running install_egg_info
Writing C:\Python27\Lib\site-packages\package_server-1.0.1-py2.7.egg-info

C:\Users\Desktop\package_server>python
Python 2.7.6 (default, Nov 10 2013, 19:24:18) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import package_server
>>>
```

Figure 4: Successful installation of package_server.py

Note: In some cases teams may get an error as shown in Figure 5.



```
C:\Windows\system32\cmd.exe

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\>python
'python' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\>
```

Figure 5: python not recognized as a command

This is because your environmental variable is not set, i.e., the PC/Laptop has no reference to python. Thus we need to add paths from where the PC/Laptop can call python. Teams that get this error need to follow **Steps 5 to 8**. Teams that do not get this error may directly move to **Step 9**.

Step 5: Right click on “Computer” and click on **Properties**.

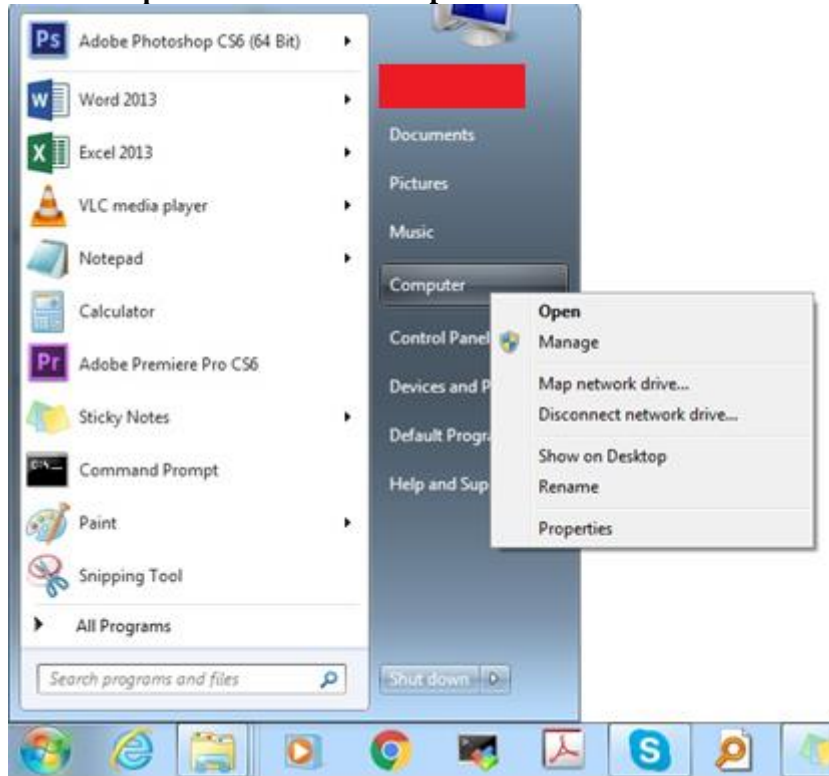


Figure 6: Opening Computer Properties

Step 6: This will open “Computer Properties” as shown in Figure 7. Now click on “Advanced system settings”. This will open a **System Properties** window as shown in Figure 8.

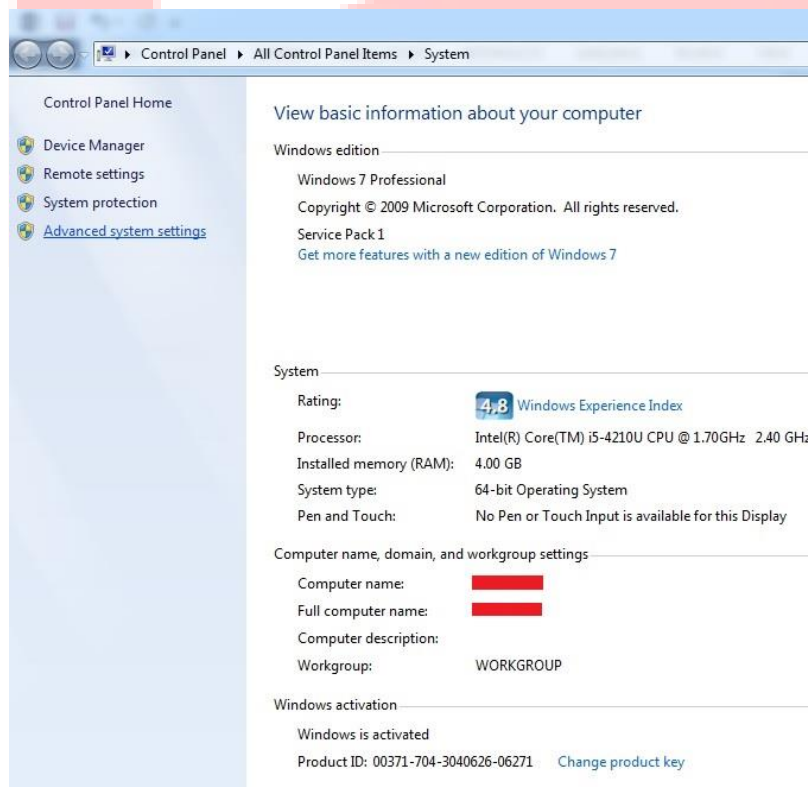


Figure 7: Computer Properties

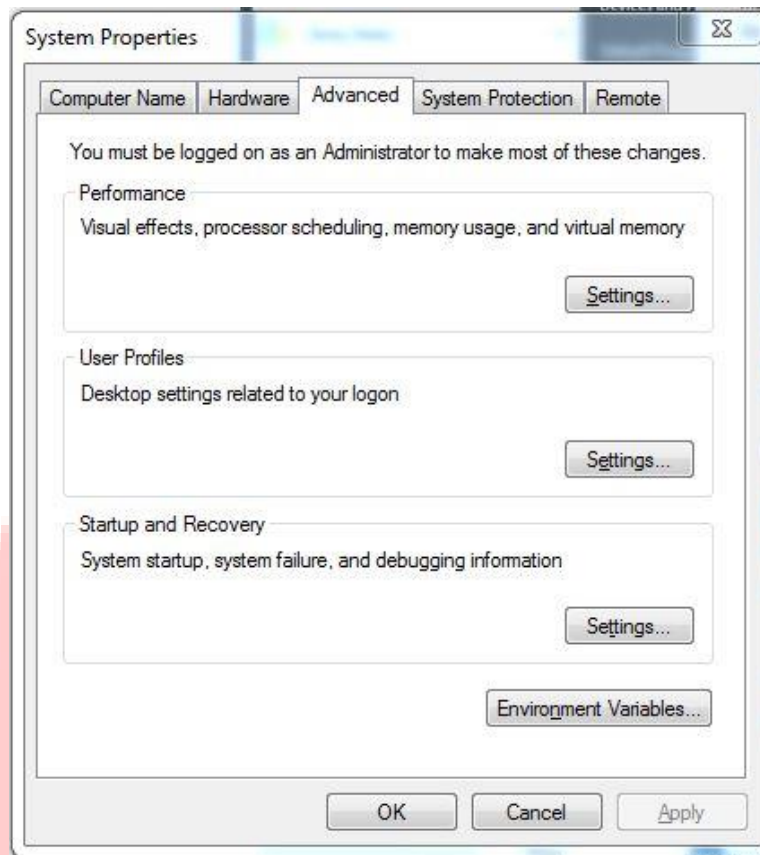


Figure 8: System Properties

Step 7: Click on **Environment Variables**. This will open an Environment Variables window as shown in Figure 9. Scroll down in the **System variables** tab till you see a variable called **Path**.

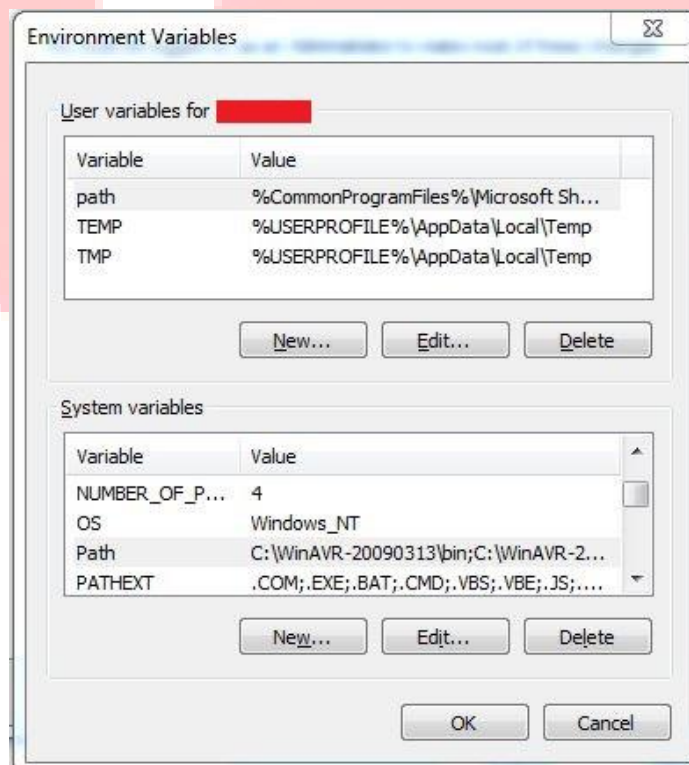


Figure 9: Environment Variables

Step 8: We need to add python paths to this variable so that the PC/Laptop gets a reference and then we can use the python command in cmd. Click on Path and click on **Edit**. Add the following paths separated by a **semi-colon (;)** as shown in Figure 10:

1. C:\Python27
2. C:\Python27\Scripts

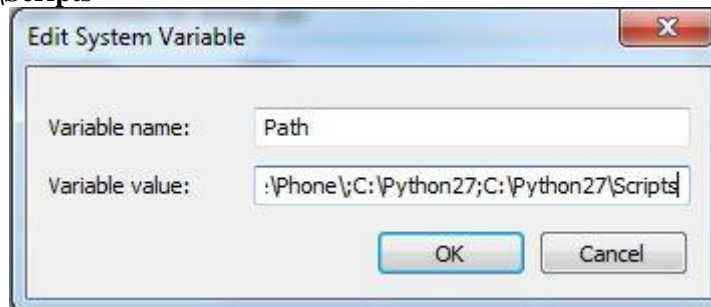


Figure 10: Adding python paths

Note: Change the path if you have installed Python in another directory.

Step 9: Open IDLE and write “from package_server import pickup_table”.

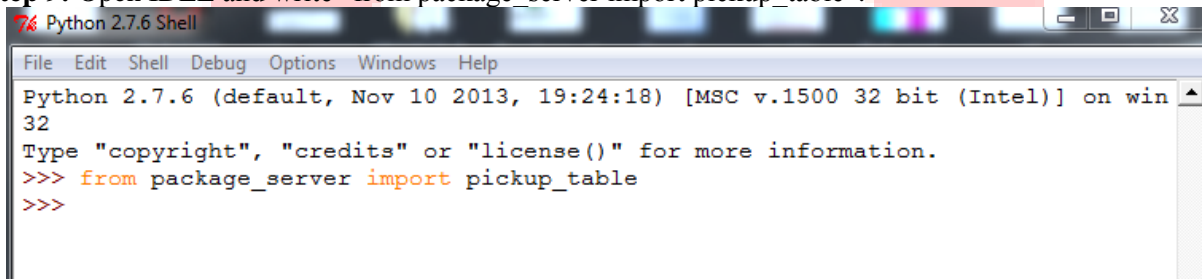


Figure 11: Importing pickup_table

Step 10: Create an instant of class pickup_table by writing the command:
pt = pickup_table() // pt can be any variable

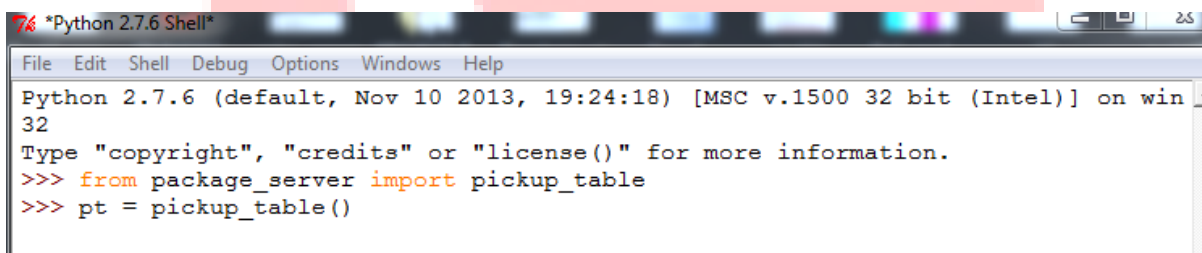
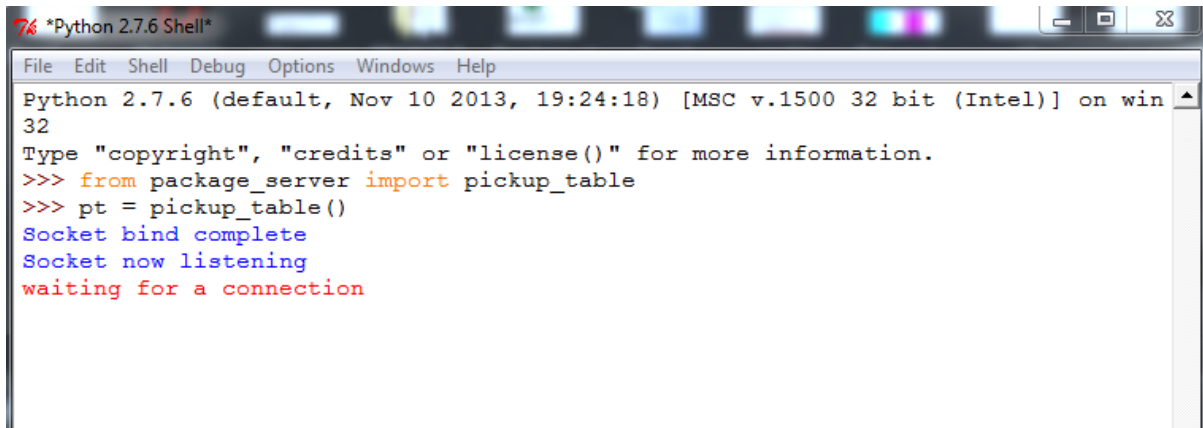


Figure 12: Creating an instance of pickup_table

This will open a new window named “Packages on the Robot” as shown in Figure 13.



Figure 13: Packages on the Robot window



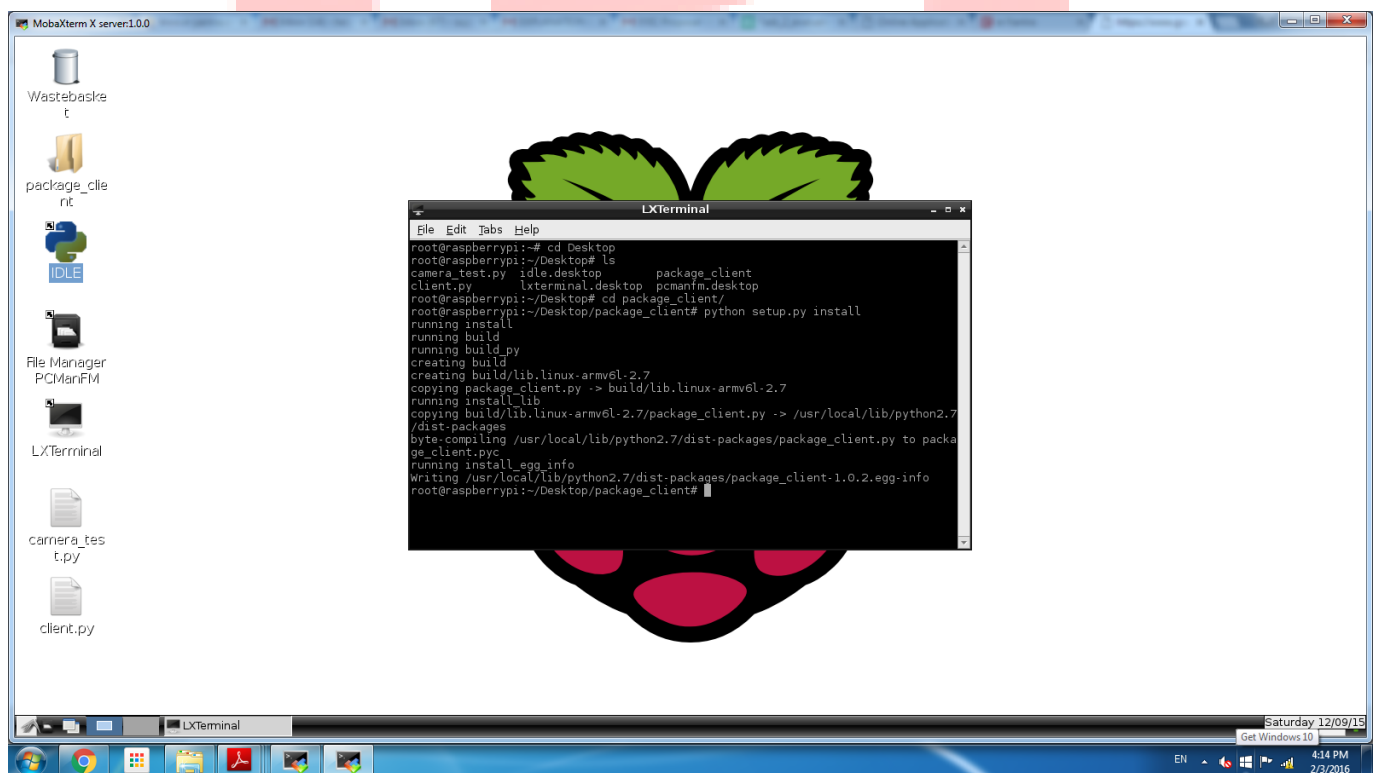
```
Python 2.7.6 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.6 (default, Nov 10 2013, 19:24:18) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> from package_server import pickup_table
>>> pt = pickup_table()
Socket bind complete
Socket now listening
waiting for a connection
```

Figure 14: Message displayed at server (PC/Laptop) shell

Steps to be followed on the Raspberry Pi:

Step 11: Copy the folder Package_client in a USB device and connect the USB device to RPi. Now copy Package_client to Desktop of Raspberry Pi.

Step 12: Open Terminal and change directory to the folder containing package_client.py and install it using the procedure mentioned in Steps 3 and 4 for **package_client**.

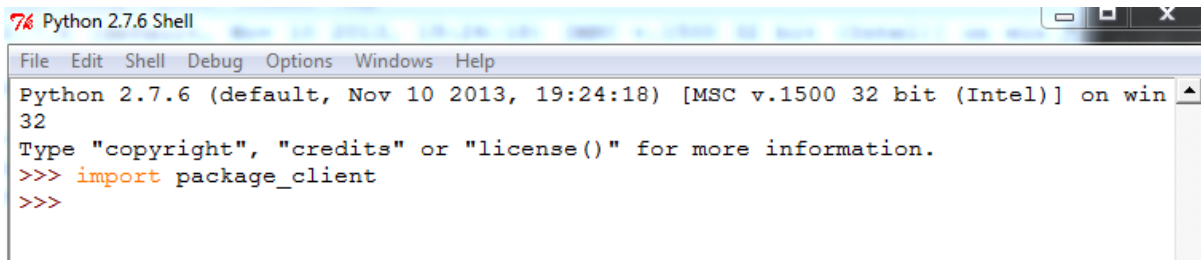


```
MobaXterm X server 1.0.0
Wastebasket
package_client
idle
File Manager
PCManFM
LXTerminal
camera_test.py
client.py

LXTerminal
root@raspberrypi:~# cd Desktop
root@raspberrypi:~/Desktop# ls
camera_test.py  idle.desktop  package_client
client.py       lxterminal.desktop  pcmanfm.desktop
root@raspberrypi:~/Desktop# cd package_client/
root@raspberrypi:~/Desktop/package_client# python setup.py install
running install
running build
running build_py
creating build
creating build/lib.linux-armv6l-2.7
copying package_client.py -> build/lib.linux-armv6l-2.7
running install_lib
copying build/lib.linux-armv6l-2.7/package_client.py -> /usr/local/lib/python2.7
/dist-packages
byte-compiling /usr/local/lib/python2.7/dist-packages/package_client.py to packa
ge_client.pyc
running install_egg_info
writing /usr/local/lib/python2.7/dist-packages/package_client-1.0.2.egg-info
root@raspberrypi:~/Desktop/package_client#
```

Figure 15: Installing package_client.py

Step 13: Open IDLE in Raspberry Pi and write “import package_client”.



```
Python 2.7.6 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.6 (default, Nov 10 2013, 19:24:18) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> import package_client
>>>
```

Figure 16: Importing package_client

Steps to be followed on the PC/Laptop:

Step 14: Go to “Wireless Network Connection Status” in the PC/Laptop.

Click “Details” and note down IPv4 Address from the “Network Connection Details” window.

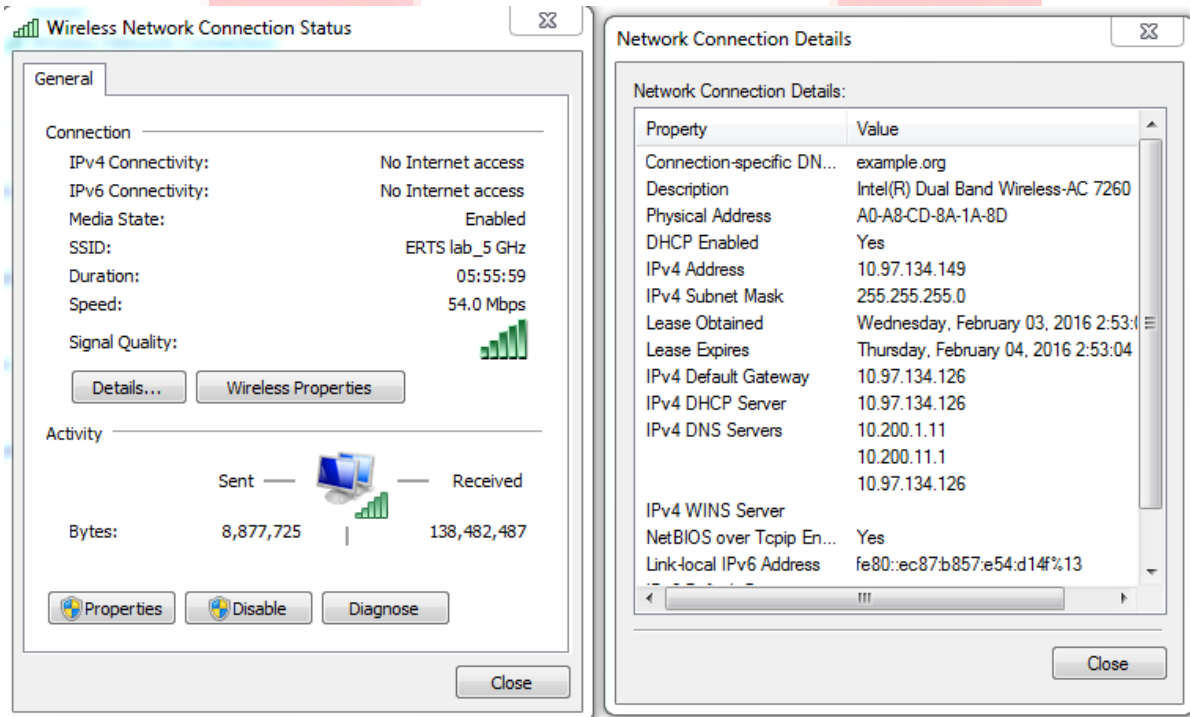


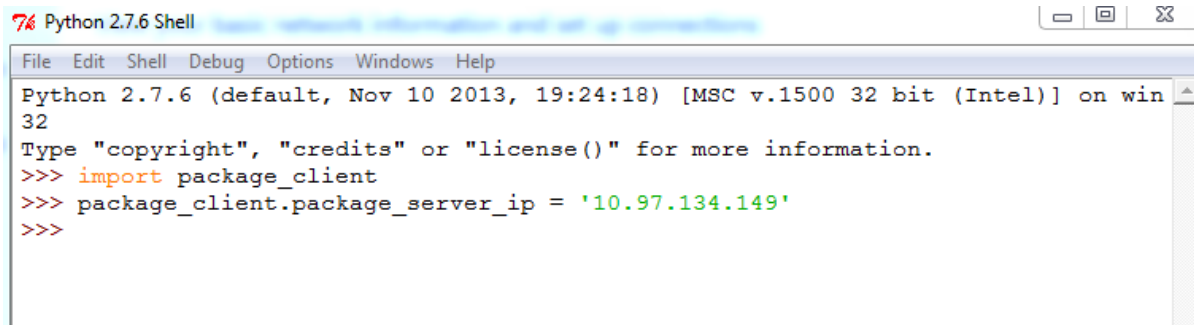
Figure 17: Noting down the IPv4 address

Steps to be followed on the Raspberry Pi:

Step15: Now in the shell of Raspberry Pi write the following command:

package_client.package_server_ip = 'IPv4 Address'

In our case it will be, package_client.package_server_ip = '10.97.134.149'



```
Python 2.7.6 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.6 (default, Nov 10 2013, 19:24:18) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> import package_client
>>> package_client.package_server_ip = '10.97.134.149'
>>>
```

Figure 18: Setting the IPv4 address of server (PC/Laptop) at client's (RPi's) shell

Step 16: Call the function `package_client.Message('PC1')`

Here, the first letter ('P') in the argument ('PC1') of the function call represents the colour of the package and it can have any of the following values:

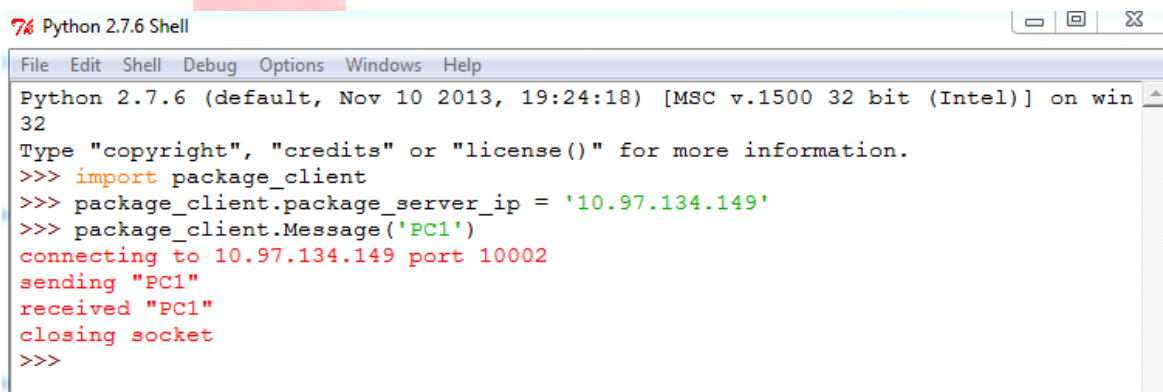
- P for Pink
- G for Green
- O for Orange
- B for Cyan

The second letter ('C') in the argument ('PC1') of the function call represents the shape of the package and it can have following values:

- C for Circle
- S for Square
- T for Triangle

The third letter ('1') in the argument ('PC1') of the function call represents the functional state of the package and it can have following values:

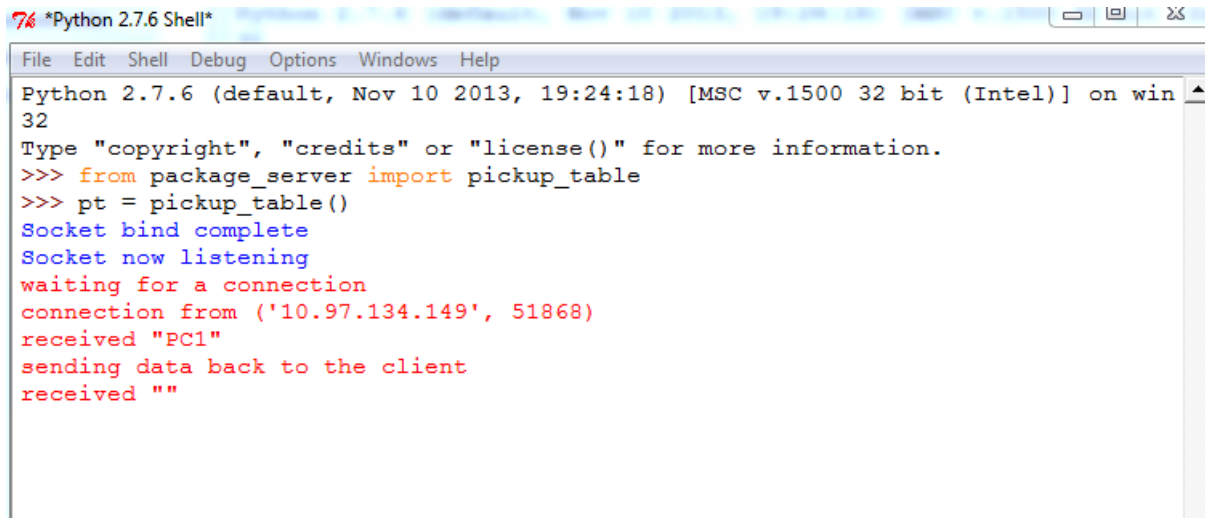
- 1 for Picking-up
- 0 for Delivering



```
Python 2.7.6 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.6 (default, Nov 10 2013, 19:24:18) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> import package_client
>>> package_client.package_server_ip = '10.97.134.149'
>>> package_client.Message('PC1')
connecting to 10.97.134.149 port 10002
sending "PC1"
received "PC1"
closing socket
>>>
```

Figure 19: Sending Message ('PC1') from client (RPi) to server (PC/Laptop)

On the sever shell we get the text shown in Figure 20 and a Pink Circle in “Packages on the Robot” window as shown in Figure 21.



```
Python 2.7.6 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.6 (default, Nov 10 2013, 19:24:18) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> from package_server import pickup_table
>>> pt = pickup_table()
Socket bind complete
Socket now listening
waiting for a connection
connection from ('10.97.134.149', 51868)
received "PC1"
sending data back to the client
received ""
```

Figure 20: Messages shown at server (PC/Laptop) shell

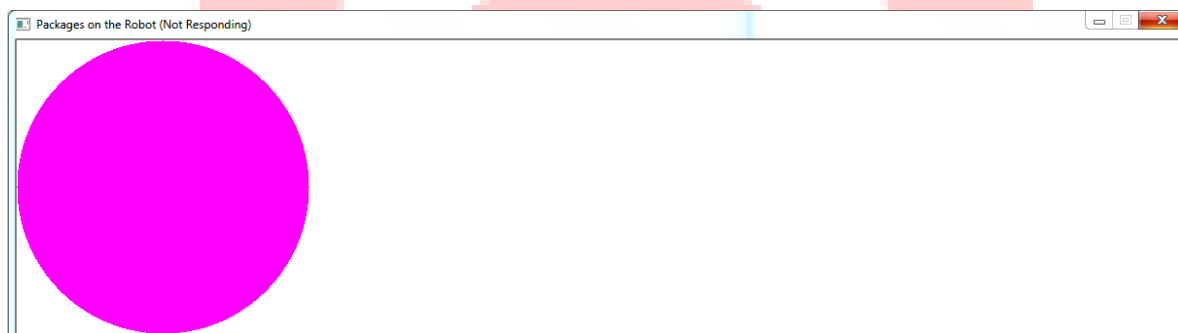


Figure 21: Pink Circle shown at server (PC/Laptop) shell

Example:

Suppose we pick up packages in the following order:

1. Pink Circle
2. Orange Circle
3. Green Triangle
4. Sky Blue Square

For Picking up Packages in the above order call `package_client.Message(' ')` at the client shell.

```
Python 2.7.6 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.6 (default, Nov 10 2013, 19:24:18) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> import package_client
>>> package_client.package_server_ip = '10.97.134.149'
>>> package_client.Message('PC1')
connecting to 10.97.134.149 port 10002
sending "PC1"
received "PC1"
closing socket
>>> package_client.Message('OC1')
connecting to 10.97.134.149 port 10002
sending "OC1"
received "OC1"
closing socket
>>> package_client.Message('GT1')
connecting to 10.97.134.149 port 10002
sending "GT1"
received "GT1"
closing socket
>>> package_client.Message('BS1')
connecting to 10.97.134.149 port 10002
sending "BS1"
received "BS1"
closing socket
```

Figure 22: Message function calls to generate output

The function **Message()** call will generate the output as shown in Figure 23:

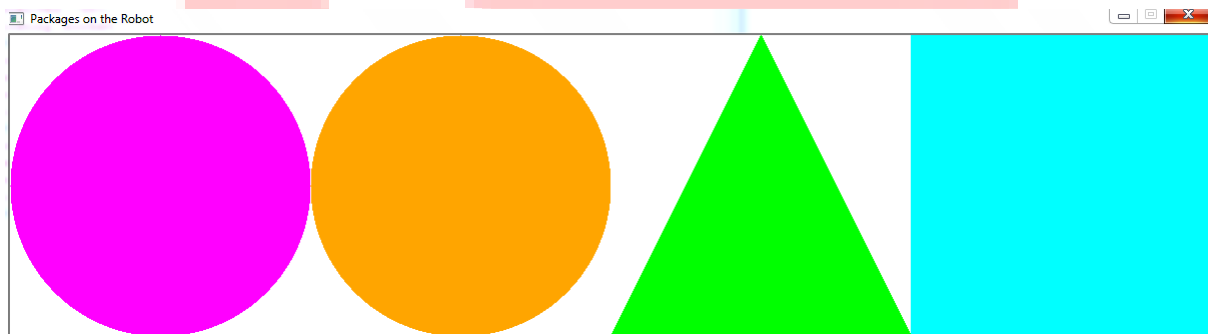
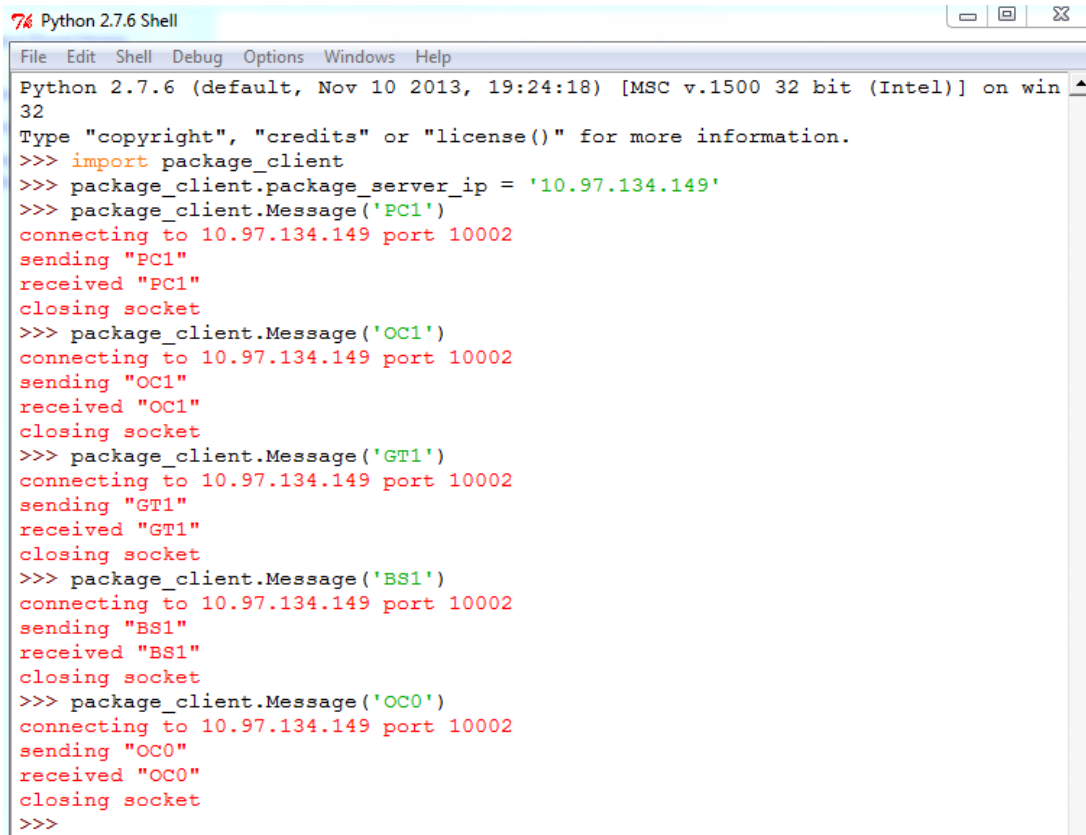


Figure 23: Packages picked up by the Robot

Now let's assume the robot delivers Orange Circle. To deliver the package we need to call `package_client.Message('OC0')` as shown in Figure 24.



```
Python 2.7.6 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.6 (default, Nov 10 2013, 19:24:18) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> import package_client
>>> package_client.package_server_ip = '10.97.134.149'
>>> package_client.Message('PC1')
connecting to 10.97.134.149 port 10002
sending "PC1"
received "PC1"
closing socket
>>> package_client.Message('OC1')
connecting to 10.97.134.149 port 10002
sending "OC1"
received "OC1"
closing socket
>>> package_client.Message('GT1')
connecting to 10.97.134.149 port 10002
sending "GT1"
received "GT1"
closing socket
>>> package_client.Message('BS1')
connecting to 10.97.134.149 port 10002
sending "BS1"
received "BS1"
closing socket
>>> package_client.Message('OC0')
connecting to 10.97.134.149 port 10002
sending "OC0"
received "OC0"
closing socket
>>>
```

Figure 24: Dropping the package Orange Circle

The output on the 'Packages on the Robot' window will change as shown in Figure 25 at the server's (PC/Laptop) end.

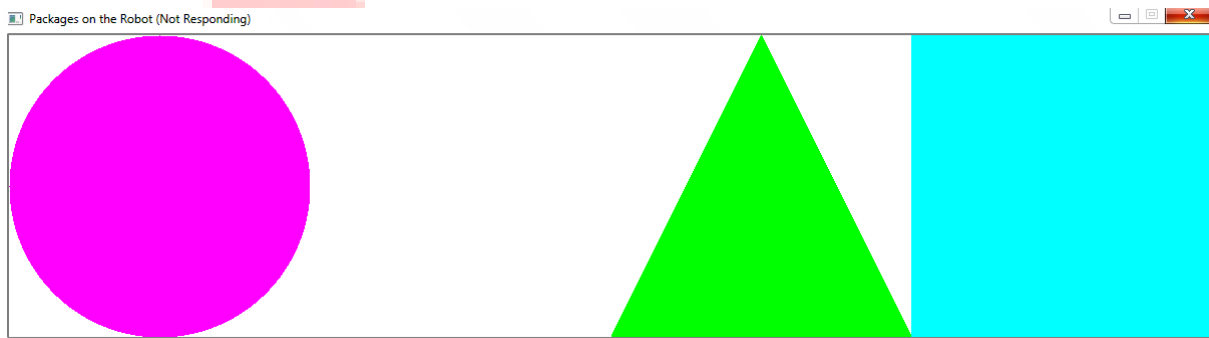


Figure 25: Delivery reflected on Packages on the Robot window

Similarly we can deliver remaining Packages by calling Message function in the Following order:

1. `package_client.Message('BS0')`
2. `package_client.Message('GT0')`
3. `package_client.Message('PC0')`

After delivering all the Packages the "Packages on the Robot" window will look like as shown in Figure 26.

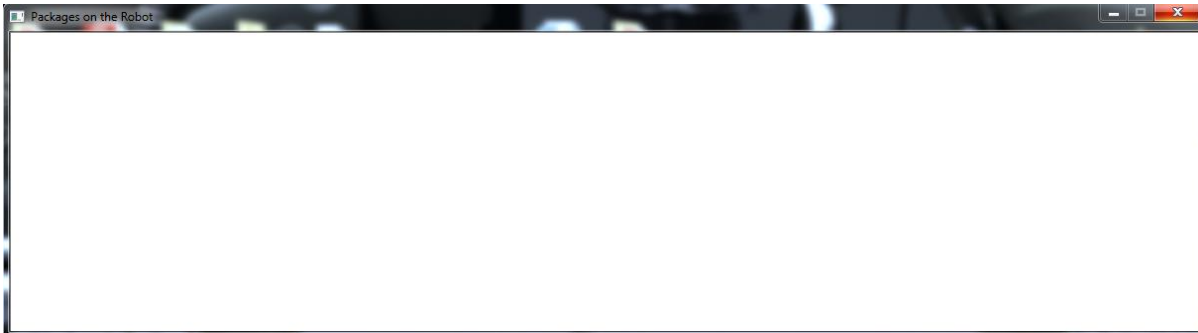
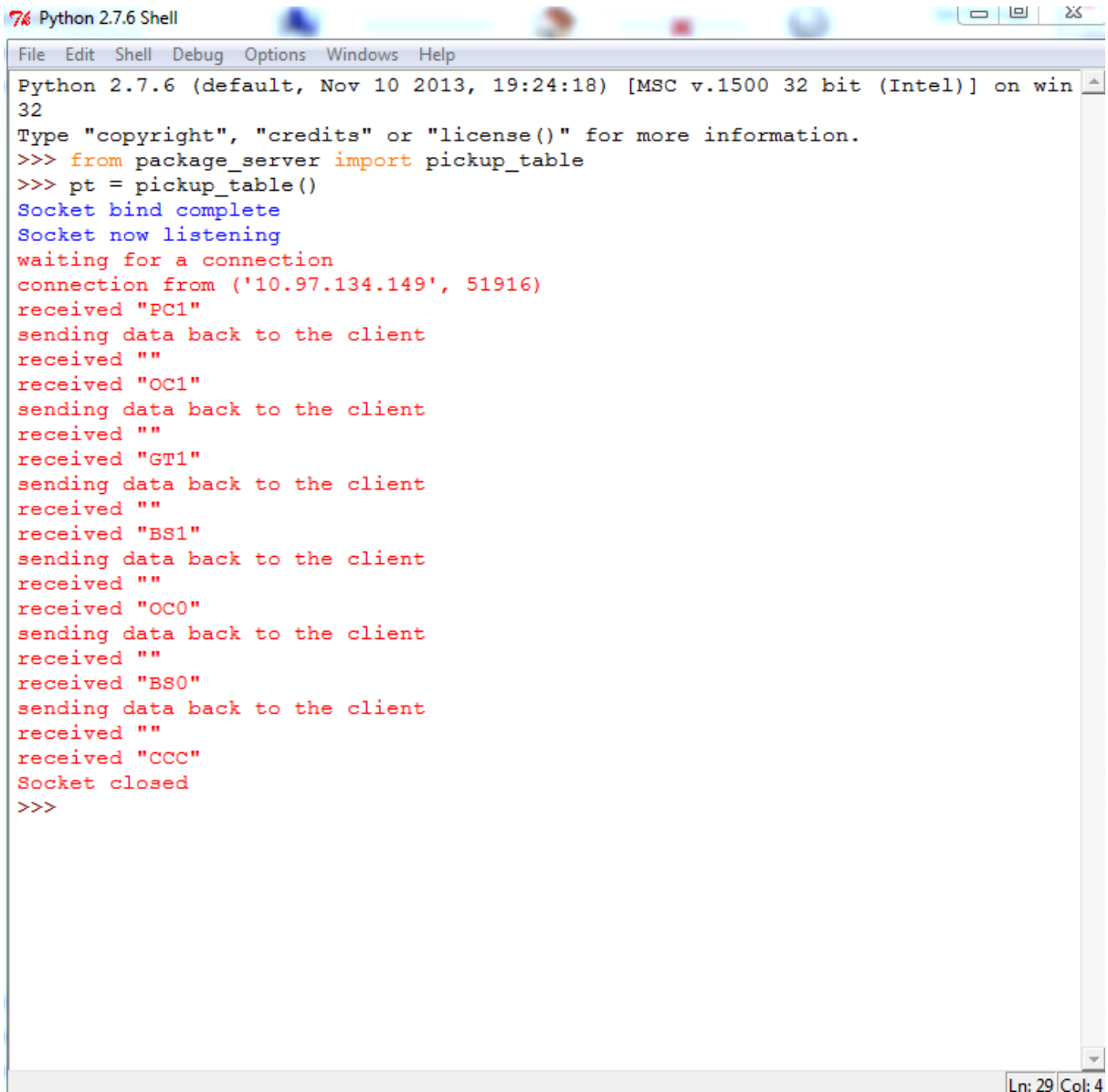


Figure 26: After delivering all the Packages

When the argument 'CCC' is sent, i.e., `package_client.Message('CCC')` from the server end the Socket closes.

```
Python 2.7.6 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.6 (default, Nov 10 2013, 19:24:18) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> import package_client
>>> package_client.package_server_ip = '10.97.134.149'
>>> package_client.Message('PC1')
connecting to 10.97.134.149 port 10002
sending "PC1"
received "PC1"
closing socket
>>> package_client.Message('OC1')
connecting to 10.97.134.149 port 10002
sending "OC1"
received "OC1"
closing socket
>>> package_client.Message('GT1')
connecting to 10.97.134.149 port 10002
sending "GT1"
received "GT1"
closing socket
>>> package_client.Message('BS1')
connecting to 10.97.134.149 port 10002
sending "BS1"
received "BS1"
closing socket
>>> package_client.Message('OC0')
connecting to 10.97.134.149 port 10002
sending "OC0"
received "OC0"
closing socket
>>> package_client.Message('BS0')
connecting to 10.97.134.149 port 10002
sending "BS0"
received "BS0"
closing socket
>>> package_client.Message('CCC')
connecting to 10.97.134.149 port 10002
sending "CCC"
closing socket
>>>
```

Figure 27: Closing socket at client (RPi) shell



```
Python 2.7.6 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.6 (default, Nov 10 2013, 19:24:18) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> from package_server import pickup_table
>>> pt = pickup_table()
Socket bind complete
Socket now listening
waiting for a connection
connection from ('10.97.134.149', 51916)
received "PC1"
sending data back to the client
received ""
received "OC1"
sending data back to the client
received ""
received "GT1"
sending data back to the client
received ""
received "BS1"
sending data back to the client
received ""
received "OC0"
sending data back to the client
received ""
received "BS0"
sending data back to the client
received ""
received "CCC"
Socket closed
>>>
```

Figure 28: Closing socket at server (PC/Laptop) shell

The End!