GeeksforGeeks
A computer science portal for geeks

Custom Searcl

Practice | GATE CS | Placements | Videos | Contribute

Login/Register

Quick Links for Python
Recent Articles
MCQ / Quizzes
Practice Problems

Basics
Introduction
New Generation Language
Keywords, Set 1 Set 2
Explore More...
Variables
Variables,Expressions & Functions
Global and Local Variables
Type Conversion
Explore More...
Operators
Increment and Decrement Operator

Socket Programming in Python

2.6

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.
They are the real backbones behind web browsing. In simpler terms there is a server and a client.

Socket programming is started by importing the socket library and making a simple socket.

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Here we made a socket instance and passed it two parameters. The first parameter is **AF_INET** and the second one is **SOCK_STREAM**. AF_INET refers to the address family ipv4. The SOCK_STREAM means connection oriented TCP protocol.

Now we can connect to a server using this socket.

Connecting to a server:
Note that if any error occurs during the creation of a socket then a socket.error is thrown and we can only connect to a server by knowing it's ip. You can find the ip of the server by using this :

```
$ ping www.google.com
```

You can also find the ip using python:

```
import socket

ip = socket.gethostbyname('www.google.com')
```

```
print ip
```

Here is an example of a script for connecting to Google

```
# An example script to connect to Google using socket
# programming in Python
import socket # for socket
import sys

try:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print "Socket successfully created"
except socket.error as err:
    print "socket creation failed with error %s" %(err)

# default port for socket
port = 80

try:
    host_ip = socket.gethostbyname('www.google.com')
except socket.gaierror:

    # this means could not resolve the host
    print "there was an error resolving the host"
    sys.exit()

# connecting to the server
s.connect((host_ip, port))

print "the socket has successfully connected to google \
on port == %s" %(host_ip)
```

Run on IDE

Output :

```
Socket successfully created
the socket has successfully connected to google
on port == 173.194.40.19
```

- First of all we made a socket.
- Then we resolved google's ip and lastly we connected to google.
- Now we need to know how can we send some data through a socket.
- For sending data the socket library has a *sendall* function. This function allows you to send data to a server to which the socket is connected and server can also send data to the client using this function.

A simple server-client program :

Server :

A server has a bind() method which binds it to a specific ip and port so that it can listen to incoming requests on that ip and port.A server has a listen() method which puts the server into listen mode. This allows the server to listen to incoming connections. And last a server has an accept() and close() method. The accept method initiates a connection with the client and the close method closes the connection with the client.

```
# first of all import the socket library
import socket

# next create a socket object
s = socket.socket()
print "Socket successfully created"

# reserve a port on your computer in our
# case it is 12345 but it can be anything
port = 12345

# Next bind to the port
# we have not typed any ip in the ip field
# instead we have inputted an empty string
# this makes the server listen to requests
# coming from other computers on the network
s.bind(('', port))
print "socket binded to %s" %(port)

# put the socket into listening mode
s.listen(5)
print "socket is listening"

# a forever loop until we interrupt it or
# an error occurs
while True:

   # Establish connection with client.
   c, addr = s.accept()
   print 'Got connection from', addr

   # send a thank you message to the client.
   c.send('Thank you for connecting')

   # Close the connection with the client
   c.close()
```

Run on IDE

- First of all we import socket which is necessary.
- Then we made a socket object and reserved a port on our pc.
- After that we binded our server to the specified port. Passing an empty string means that the server can listen to incoming connections from other computers as well. If we would have passed 127.0.0.1 then it would have listened to only those calls made within the local computer.
- After that we put the server into listen mode.5 here means that 5 connections are kept waiting if the server is busy and if a 6th socket trys to connect then the connection is refused.
- At last we make a while loop and start to accept all incoming connections and

close those connections after a thank you message to all connected sockets.

Client :

Now we need something with which a server can interact. We could tenet to the server like this just to know that our server is working. Type these commands in the terminal:

```
# start the server
$ python server.py
```

# keep the above terminal open

# now open another terminal and type:

```
$ telnet localhost 12345
```

Output :

```
# in the server.py terminal you will see
# this output:
Socket successfully created
socket binded to 12345
socket is listening
Got connection from ('127.0.0.1', 52617)
```

```
# In the telnet terminal you will get this:
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Thank you for connectingConnection closed by foreign host.
```

This output shows that our server is working.

Now for the client side:

```python
# Import socket module
import socket

# Create a socket object
s = socket.socket()

# Define the port on which you want to connect
port = 12345

# connect to the server on local computer
s.connect(('127.0.0.1', port))

# receive data from the server
print s.recv(1024)
```

```
# close the connection
s.close()
```

Run on IDE

- First of all we make a socket object.
- Then we connect to localhost on port 12345 (the port on which our server runs) and lastly we receive data from the server and close the connection.
- Now save this file as client.py and run it from the terminal after starting the server script.

```
# start the server:
$ python server.py
Socket successfully created
socket binded to 12345
socket is listening
Got connection from ('127.0.0.1', 52617)
```

```
# start the client:
$ python client.py
Thank you for connecting
```

Reference : Python Socket Programming

This article is contributed by **Kishlay Verma**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GATE CS Corner **Company Wise Coding Practice**

Computer Networks Python Technical Scripter Login to Improve this Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

Simple Chat Room using Python
Socket Programming with Multi-threading in Python
Explicitly assigning port number to client in Socket
try and except in Python
Socket Programming in Java

Extracting MAC address using Python
Weak RSA decryption with Chinese-remainder theorem
Computer Network | Packet flow in different network
Hash Functions in System Security
Message Authentication Codes

(Login to Rate)

2.6 Average Difficulty : **2.6/5.0**
Based on **5** vote(s)

Basic Easy Medium Hard Expert

☐ Add to TODO List
☐ Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Share this post!

1 Comment **GeeksforGeeks** **Login**

Recommend 6 **Share** **Sort by Newest**

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Surendra Lalwani • 2 months ago
In the Server side program, the program is working fine but with the people using python3 or python version greater than 35 then in c.send('Thank you for connecting') you will get error TypeError: a bytes-like object is required, not 'str' because in Python 3, strings are Unicode, but when transmitting on the network, the data needs to be bytes strings instead. HENCE use c.send(b'Thank you for connecting').
• Reply • Share ›

Subscribe **Add Disqus to your siteAdd DisqusAdd** **Privacy**

Privacy Policy