Python

Python and IRC
Contributed by Peyton McCullough
2005−02−16
[*Send Me Similar Content When Posted*]

[*Add Developer Shed Headlines To Your Site*]

DISCUSS NEWS SEND PRINT PDF

Article Index:

IRC is becoming an increasingly popular medium for communication. In this article, Peyton McCullough explains how to make Python and IRC work together.

Introduction

I'm sure you've all heard of it – the modern miracle known as Internet Relay Chat, or IRC. It allows geeks, such as myself, to converse with other people from around the globe. While you can connect to it with a vanilla client, you can also connect to it with another miracle – Python.

Python can connect to a channel and act as anything you like – a calculator, a weatherman, a scribe or a silent occupant. In addition, it is fairly simple to make Python and IRC get along, contrary to what you might be thinking right now, and this article will explain exactly how to do it. By the end of this article, you should have a basic understanding of the IRC protocol and how to use it in your Python scripts.

To understand this article, you will need an understanding of the Python language and an understanding of sockets. You should also be familiar with IRC.

Our first task is to connect to an IRC network. To do this, we must first create a socket. Then, we must connect to the network, and, finally, we must complete a few short steps to become eligible to interact with other users.

```
import socket

network = 'irc.insert.a.network.here'
port = 6667
irc = socket.socket ( socket.AF_INET,
socket.SOCK_STREAM )
```

```
irc.connect ( ( network, port ) )
irc.send ( 'NICK PyIRC\r\n' )
irc.send ( 'USER PyIRC PyIRC PyIRC :Python IRC\r\n'
)
irc.send ( 'QUIT\r\n' )
irc.close()
```

Although the code demonstrates the absolute basics, it doesn't do anything special. In fact, the server you connect to might not even acknowledge the data you send it until after a few seconds – which we do not allow for in the script. Don't worry though, we will soon take a look at a fully functional script after we tackle the very basics.

The first piece of data we send to it sets our nickname. Notice how we suffix each outgoing message with a carriage return and line feed ("\r\n"). Take note of this because it is very important. We then specify our username ("PyIRC"), host name ("PyIRC"), server name ("PyIRC") and real name ("Python IRC") in the next outgoing line. Finally, we issue the "QUIT" command and close the connection.

You should also take note of the colon before "Python IRC." Colons tell the server that the attribute will possibly be made up of multiple words.

We will use and develop this skeleton throughout the remainder of the article, so it is important that you understand it.

Now that we know how to connect, our next task is joining a channel and sending a message to its occupants. This is suprisingly easy.

```
import socket

network = 'irc.insert.a.network.here'
port = 6667
irc = socket.socket ( socket.AF_INET,
socket.SOCK_STREAM )
irc.connect ( ( network, port ) )
irc.send ( 'NICK PyIRC\r\n' )
irc.send ( 'USER PyIRC PyIRC PyIRC :Python IRC\r\n'
)
irc.send ( 'JOIN #pyirc\r\n' )
irc.send ( 'PRIVMSG #pyirc :Can you hear me?\r\n' )
irc.send ( 'PART #pyirc\r\n' )
irc.send ( 'QUIT\r\n' )
irc.close()
```

Note that, again, the code will probably not perform the instructions we gave it, for the same reason as last time. However, the above code would ideally join the channel #pyirc and say "Can you hear me?" Let's break the code down to see how it works. The first few lines should already look familiar, and the last few lines should look familiar as well. We wrote them in the previous section. However, the "JOIN," "PRIVMGS," and "PART" lines are new. They simply join the channel, send a message to the channel and leave the channel, respectively.

To join multiple channels, just issue the "JOIN" command again. When sending a message, be sure to specify the name of the channel in the "PRIVMSG" command. You can also message another user with the "PRIVMSG" command.

```
import socket

network = 'irc.insert.a.network.here'
port = 6667
irc = socket.socket ( socket.AF_INET,
socket.SOCK_STREAM )
irc.connect ( ( network, port ) )
irc.send ( 'NICK PyIRC\r\n' )
irc.send ( 'USER PyIRC PyIRC PyIRC :Python IRC\r\n'
)
irc.send ( 'PRIVMSG Jimbo :Can you hear me?\r\n' )
irc.send ( 'QUIT\r\n' )
irc.close()
```

So far, we can connect to an IRC network and send a message to a given channel. We can also send a message to another user. Now we will accept messages from both the server and from other users.

Every once in a while, the server will send us a "PING" command to check on us. To stay connected, we must send the server a "PONG" command. Let's build a script that does just that.

```
import socket

network = 'irc.insert.a.network.here'
port = 6667
irc = socket.socket ( socket.AF_INET,
socket.SOCK_STREAM )
irc.connect ( ( network, port ) )
print irc.recv ( 4096 )
irc.send ( 'NICK PyIRC\r\n' )
irc.send ( 'USER PyIRC PyIRC PyIRC :Python IRC\r\n'
)
irc.send ( 'JOIN #pyirc\r\n' )
irc.send ( 'PRIVMSG #pyirc :Hello.\r\n' )
while True:
   data = irc.recv ( 4096 )
   if data.find ( 'PING' ) != -1:
      irc.send ( 'PONG ' + data.split() [ 1 ] +
'\r\n' )
   print data
```

As you can see, we've changed our skeleton a bit in this script. We have replaced the bottom section with an infinite loop. The loop receives data, and if a "PING" command is present, it replies with a "PONG" command. Feel free to test the script out.

Let's modify our script to accept messages. Messages come in a form similar to this:

```
:Nick!user@host PRIVMSG destination :Message
```

Here's an example:

```
:Peyton!~peyton@python.org PRIVMSG #pyrc :Hey!
```

In our new script, we will break down the above form of data.

```
import socket

network = 'irc.insert.a.network.here'
port = 6667
irc = socket.socket ( socket.AF_INET,
socket.SOCK_STREAM )
irc.connect ( ( network, port ) )
trash = irc.recv ( 4096 )
irc.send ( 'NICK PyIRC\r\n' )
irc.send ( 'USER PyIRC PyIRC PyIRC :Python IRC\r\n'
)
irc.send ( 'JOIN #pyirc\r\n' )
irc.send ( 'PRIVMSG #pyirc :Hello.\r\n' )
while True:
   data = irc.recv ( 4096 )
   if data.find ( 'PING' ) != -1:
      irc.send ( 'PONG ' + data.split() [ 1 ] +
'\r\n' )
   elif data.find ( 'PRIVMSG' ) != -1:
      nick = data.split ( '!' ) [ 0 ].replace (
':', '' )
      message = ':'.join ( data.split ( ':' ) [ 2:
] )
      print nick + ':', message
```

We've added a few lines to the script. If the "PRIVMSG" command is found inside the line of data, we pull the line apart to get the nickname of the person who sent it and the message by using the split() function. Run the script, join #pyirc on the specified network and test out the script.

Now we need our script to discriminate between messages in different channels and private messages. This can be done easily by extracting the destination from the "PRIVMSG" command.

```
import socket

network = 'irc.insert.a.network.here'
port = 6667
irc = socket.socket ( socket.AF_INET,
socket.SOCK_STREAM )
irc.connect ( ( network, port ) )
irc.recv ( 4096 )
irc.send ( 'NICK PyIRC\r\n' )
```

```
irc.send ( 'USER PyIRC PyIRC PyIRC :Python IRC\r\n'
)
irc.send ( 'JOIN #pyirc\r\n' )
irc.send ( 'PRIVMSG #pyirc :Hello.\r\n' )
while True:
   data = irc.recv ( 4096 )
   if data.find ( 'PING' ) != -1:
      irc.send ( 'PONG ' + data.split() [ 1 ] +
'\r\n' )
   elif data.find ( 'PRIVMSG' ) != -1:
      nick = data.split ( '!' ) [ 0 ].replace (
':', '' )
      message = ':'.join ( data.split ( ':' ) [ 2:
] )
      destination = ''.join ( data.split ( ':' ) [
:2 ] ).split ( ' ' ) [ -2 ]
      if destination == 'PyIRC':
         destination = 'PRIVATE'
      print '(', destination, ')', nick + ':',
message
```

Test out the script like before and see the result. You should see something similar to this:

```
( #pyirc ) Peyton: Test

( PRIVATE ) Peyton: This is a private message.
```

Let's apply our knowledge to something useful – a Python-powered IRC bot that performs basic mathematical functions. Let's make the bot perform arithmetic calculations and a few trigonometric functions: sine, cosine and tangent.

We will first create a file to perform the calculations for us. Create a file named ircMath.py and insert the following code.

```
import math

def arithmatic ( args ):

   args [ 0 ] = args [ 0 ].replace ( '\r\n', '' )
   for letter in 'abcdefghijklmnopqrstuvwxyz':
      args [ 0 ] = args [ 0 ].replace ( letter, ''
)
   solution = str ( eval ( args [ 0 ], {
'__builtins__' : {} } ) )
   return solution

def sine ( args ):

   solution = str ( math.sin ( float ( args [ 0 ] )
* ( 2 * math.pi ) / 360 ) )
   return solution
```

```
def cosine ( args ):

   solution = str ( math.cos ( float ( args [ 0 ] )
* ( 2 * math.pi ) / 360 ) )
   return solution

def tangent ( args ):

   solution = str ( math.tan ( float ( args [ 0 ] )
* ( 2 * math.pi ) / 360 ) )
   return solution
```

The guts of ircMath.py aren't too important, and the code should be pretty straightforward, so I won't get into detail.

We will now create the bot. Its code will be a modified version of the last section's script. We will search the incoming message for the string "%PyIRC" to discriminate between messages we do and do not need. We will then split up the message into a function and arguments. Finally, we will call the appropriate function.

```
import ircMath
import socket

network = 'irc.insert.a.network.here'
port = 6667
irc = socket.socket ( socket.AF_INET,
socket.SOCK_STREAM )
irc.connect ( ( network, port ) )
irc.recv ( 4096 )
irc.send ( 'NICK PyIRC\r\n' )
irc.send ( 'USER PyIRC PyIRC PyIRC :Python IRC\r\n'
)
irc.send ( 'JOIN #pyirc\r\n' )
irc.send ( 'PRIVMSG #pyirc :Hello.\r\n' )
while True:
   data = irc.recv ( 4096 )
   if data.find ( 'PING' ) != -1:
      irc.send ( 'PONG ' + data.split() [ 1 ] +
'\r\n' )
   elif data.find ( 'PRIVMSG' ) != -1:
      message = ':'.join ( data.split ( ':' ) [ 2:
] )
      if message.lower().find ( '%pyirc' ) == 0:
         nick = data.split ( '!' ) [ 0 ].replace (
':', '' )
         destination = ''.join ( data.split ( ':' )
[ :2 ] ).split ( ' ' ) [ -2 ]
         function = message.split ( ' ' ) [ 1 ]
         if function == 'calc':
            try:
               args = message.split ( ' ' ) [ 2: ]
```

```
                solution = ircMath.arithmatic ( args
)
                irc.send ( 'PRIVMSG ' +  destination
+ ' :' + nick + ': ' + solution + '\r\n' )
             except:
                pass
          if function == 'sin':
             try:
                args = message.split ( ' ' ) [ 2: ]
                solution = ircMath.sine ( args )
                irc.send ( 'PRIVMSG ' +  destination
+ ' :' + nick + ': ' + solution + '\r\n' )
             except:
                pass
          if function == 'cos':
             try:
                args = message.split ( ' ' ) [ 2: ]
                solution = ircMath.cosine ( args )
                irc.send ( 'PRIVMSG ' +  destination
+ ' :' + nick + ': ' + solution + '\r\n' )
             except:
                pass
          if function == 'tan':
             try:
                args = message.split ( ' ' ) [ 2: ]
                solution = ircMath.tangent ( args )
                irc.send ( 'PRIVMSG ' +  destination
+ ' :' + nick + ': ' + solution + '\r\n' )
             except:
                pass
```

Start up the script and enter the specified channel on the specified network. Try saying these lines:

```
%PyIRC calc 2+2
%PyIRC calc 8+10
%PyIRC sin 30
%PyIRC cos 45
%PyIRC tan 27
```

Our bot should give us the answer to each of the problems. Pretty neat, huh?

Conclusion

You should now know the basics of Python and IRC connectivity. Try to expand on the examples provided in this article to create your own unique scripts. If you would like to learn more about the IRC protocol, the protocol is documented here:

http://www.irchelp.org/irchelp/rfc