[Python Secure FTP module](http://www.pythonforbeginners.com/python-on-the-web/python-secure-ftp-module/)

17 Jun 2013   | [Web & Internet](http://www.pythonforbeginners.com/category/modules-in-python/)

Tags: [pysftp](http://www.pythonforbeginners.com/tag/pysftp/)

Overview

In the previous post we covered the ftplib module in Python, which you can read

more about [here](http://www.pythonforbeginners.com/code-snippets-source-code/how-to-use-ftp-in-python/). In this post we will cover the pysftp module.

SFTP (Secure File Transfer Protocol) is used for securely exchanging files

over the Internet.

What is it?

pysftp is an easy to use sftp module that utilizes paramiko and pycrypto.

It provides a simple interface to sftp.

**Some of the features are:**

Gracefully handles both RSA and DSS private key files automatically

Supports encrypted private key files.

Logging can now be enabled/disabled

Why should I use it?

When you want to securely exchange files over the Internet.

How do I install it?

pysftp is listed on PyPi and can be installed using pip.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | # Search for pysftp  pip search pysftp    pysftp                    # - A friendly face on SFTP    #Install pysftp  pip install pysftp |

How do I use it?

Using pysftp is easy and we will show some examples on how you can use it

List a remote directory

To connect to our FTP server, we first have to import the pysftp module and

specify (if applicable) server, username and password credentials.

After running this program, you should see all the files and directories of

the current directory of your FTP server.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | import pysftp    srv = pysftp.Connection(host="your\_FTP\_server", username="your\_username",  password="your\_password")    # Get the directory and file listing  data = srv.listdir()    # Closes the connection  srv.close()    # Prints out the directories and files, line by line  for i in data:      print i |

Connection parameters

Arguments that are not given are guessed from the environment.

**Parameter** **Description**

host The Hostname of the remote machine.

username Your username at the remote machine.(None)

private\_key Your private key file.(None)

password Your password at the remote machine.(None)

port The SSH port of the remote machine.(22)

private\_key\_pass password to use if your private\_key is encrypted(None)

log log connection/handshake details (False)

Download / Upload a remote file

As in the previous example we first import the pysftp module and specify

(if applicable) server, username and password credentials.

We also import the sys module, since we want the user to specify the file to

download / upload.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | import pysftp  import sys    # Defines the name of the file for download / upload  remote\_file = sys.argv[1]    srv = pysftp.Connection(host="your\_FTP\_server", username="your\_username",  password="your\_password")    # Download the file from the remote server  srv.get(remote\_file)    # To upload the file, simple replace get with put.  srv.put(remote\_file)    # Closes the connection  srv.close() |

What’s the next step?

Play around with the script, change things and see what happens.

Try add error handling to it. What happens if no argument is passed?

Add some interaction to the program by prompting for input.

Sources

<https://code.google.com/p/pysftp/>

<http://en.wikipedia.org/wiki/SSH_File_Transfer_Protocol>