Overview

This article will show how you can use FTP in Python with the help of the

ftplib module.

Ftplib

The ftplib module in Python allows you to write Python programs that perform a

variety of automated FTP jobs. You can easily connect to a FTP server to retrieve

files and process them locally.

To use the ftplib module in Python, you first have to import it into your script.

Open a Connection

To "open" a connection to the FTP Server, you have to create the object.

Once the connection is made (opened), you can use the methods in the ftplib

module.

Several methods are available in two flavors: one for handling text files and

another for binary files.

You can easily navigate the directory structure, manage and download files.

How do I use it?

This program will first connect to a FTP server (ftp.cwi.nl) and then list the

files and directories in the FTP server root directory using the LIST() method.

**from ftplib import FTP**

**ftp = FTP('ftp.cwi.nl') *# connect to host, default port***

**ftp.login() *# user anonymous, passwd anonymous@***

**ftp.retrlines('LIST') *# list directory contents***

Our second program opens a connection to 'ftp.sunet.se' as the user 'anonymous'

with an email address of 'anonymous@sunet.se

'

It then lists the files and directories on the FTP server by using the dir()

method.

The output is saved to the 'files' variable.

I then use print to see the files on screen.

If I want I to change directory I would just use ftp.cwd(path) to do so.

To close the FTP connection, use the quit() method.

**import ftplib**

**ftp = ftplib.FTP('ftp.sunet.se', 'anonymous', 'anonymous@sunet.se**

**')**

**print "File List: "**

**files = ftp.dir()**

**print files**

**ftp.cwd("/pub/unix") *#changing to /pub/unix***

Common FTP Methods

FTP.connect(host[, port[, timeout]])

Connect to the given host and port.

The default port number is 21, as specified by the FTP protocol specification.

It is rarely needed to specify a different port number.

This function should be called only once for each instance

It should not be called at all if a host was given when the instance was created.

All other methods can only be used after a connection  
has been made.

The optional timeout parameter specifies a timeout in seconds for the connection

attempt.

If no timeout is passed, the global default timeout setting will be used.

FTP.getwelcome()

Return the welcome message sent by the server in reply to the initial connection.

This message sometimes contains disclaimers or help information that may be

relevant to the user

FTP.login([user[, passwd[, acct]]])

Log in as the given user.

The passwd and acct parameters are optional and default to the empty string.

If no user is specified, it defaults to 'anonymous'.

If user is 'anonymous', the default passwd is 'anonymous@'.

This function should be called only once for each instance, after a connection

has been established.

It should not be called at all if a host and user were given when the instance  
was created.

Most FTP commands are only allowed after the client has logged in.

The acct parameter supplies “accounting information”; few systems implement this.

FTP.retrbinary(command, callback[, maxblocksize[, rest]])

Retrieve a file in binary transfer mode.

Command should be an appropriate RETR command: 'RETR filename'.

The callback function is called for each block of data received, with a single

string argument giving the data block.

The optional maxblocksize argument specifies the maximum chunk size to read on

the low-level socket object created to do the actual transfer.

A reasonable default is chosen. rest means the same thing as in the transfercmd()

method.

FTP.retrlines(command[, callback])

Retrieve a file or directory listing in ASCII transfer mode.

Command should be an appropriate RETR command or a command such as LIST, NLST or

MLSD.

LIST retrieves a list of files and information about those files.

NLST retrieves a list of file names.

On some servers, MLSD retrieves a machine readable list of files and information

about those files.

The callback function is called for each line with a string argument containing

the line with the trailing CRLF stripped.

The default callback prints the line to sys.stdout.

FTP.dir(argument[, ...])

Produce a directory listing as returned by the LIST command, printing it to

standard output.

The optional argument is a directory to list (default is the current server

directory).

Multiple arguments can be used to pass non-standard options to the LIST command.

If the last argument is a function, it is used as a callback function as for

retrlines(); the default prints to sys.stdout.

This method returns None.

FTP.delete(filename)

Remove the file named filename from the server.

If successful, returns the text of the response, otherwise raises error\_perm on

permission errors or error\_reply on other errors.

FTP.cwd(pathname)

Set the current directory on the server.

FTP.mkd(pathname)

Create a new directory on the server.

FTP.pwd()

Return the pathname of the current directory on the server.

FTP.quit()

Send a QUIT command to the server and close the connection.

This is the “polite” way to close a connection, but it may raise an exception if

the server responds with an error to the QUIT command.

This implies a call to the close() method which renders the FTP instance useless

for subsequent calls.

FTP.close()

Close the connection unilaterally.

This should not be applied to an already closed connection such as after a

successful call to quit().

After this call the FTP instance should not be used any more.

After a call to close() or quit() you cannot reopen the connection by issuing

another login() method).

For more information, please see the official [Python Documentation](http://docs.python.org/2/library/ftplib.html)