There is a library named ftplib that you can use to program automatic file uploads to a web site. This lesson will help you learn how to use ftplib. After you complete the lesson, you may also want to consult the official Python reference for ftplib:
https://docs.python.org/3.6/library/ftplib.html

Perhaps the best way to start learning is by using the Python shell. As you enter each line of code at the shell prompt (>>>) you can see the results. Depending on the line, the ftp server may send back a text message indicating an action taken. You won't see these messages if you run the same lines in a Python program.

Below is a copy of a Python shell session I did, connecting to the GEAR web site. You can try the same lines at the shell prompt, but don't include my comments starting with the hash (#) character. Also, the first three lines below are printed on the screen when you start up the shell. Notice we are using version 3.5.3 of Python.

Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.

```
>>> from ftplib import FTP_TLS
#here we import FTP_TLS from ftplib.  FTP_TLS is a secure FTP method (user name and password are encrypted), which is required by the Yahoo web hosting service that hosts the GEAR web site

>>> s = FTP_TLS('put host name here')  # ask Mr. La Favre if you don't know GEAR's host name
#the above line creates an ftp object named s, which connects to the server hosting the domain name.

>>> s.login('put user name here', 'put password here') #ask Mr. La Favre for user name and password
#when you connect to a domain with FTP_TLS, it is a secure connection.  Now you must supply your user name and password before you can do anything else.

'230 OK. Current directory is /'
# the line above was returned by the FTP server.  Code 230 OK indicates user name and password were OK.  Also note that server indicates we are currently at the root directory (/)

>>> s.prot_p()
#if you are concerned that someone might intercept sensitive data in a file you are transferring, then you should add this line.  Then all data transported will be encrypted.
```

'200 Data protection level set to "private"'
#the above line is the message sent back by server, indicating that data will be encrypted

>>> s.dir()
#remember that we created an FTP object named s. s.dir() returns a list of subdirectories and files for the current directory, which in this case is the root directory. Below you will see a partial list that I got (I removed many of the listings). The first three listings below are subdirectories named Aidan, Alyssa and Bryn. We know this by looking at the permissions code, drwxrwxr-x, which starts with the letter d. The next entry is a text file named FTPTEST.txt. It has a permissions code of -rw-rw-r--. The letter r represents read and the letter w represents write. There are three sets of permissions: those for the Owner of the file, those for Group, those for Public. You are the owner of the file since you have accessed the site using secure FTP with the only GEAR account available (for files uploaded by other GEAR members, you would be a member of Group if you accessed site with different user names and passwords, but for GEAR there is only one user name and password available). For this particular file you have read and write privileges. Notice that the public have only read privileges (for example they can use a web browser to view the file, however they are not allowed to edit the file or delete the file). Notice that Group privileges are the same as the owner. If we had a user name and password for each GEAR member, then you could protect your files from changes made by other GEAR members if you changed the Group privileges, but we will keep it simple with just one account for log on.

Now look at the last entry for the web page file test2.html. This file was uploaded to the web site using Python ftplib. Notice that it has a different set of permissions: -rwxrwxr-x. The x stands for executable and is meant for files that are actually programs. This is perhaps a minor problem because a web page should not be designated as an executable file. I am not sure if this is a bug or if I did not use the proper code in uploading the file. We will fix this next.

```
drwxrwxr-x    2 1911913    cgi03            4096 Jan  9 10:12 Aidan

drwxrwxr-x    2 1911913    cgi03            4096 Jan  9 10:10 Alyssa

drwxrwxr-x    2 1911913    cgi03            4096 Jan  3 17:26 Bryn

-rw-rw-r--    1 1911913    cgi03              26 Jan  8 08:11 FTPTEST.txt

-rw-rw-r--    1 1911913    cgi03             976 Dec 21 07:23 gogle_chart.html

-rw-rw-r--    1 1911913    cgi03          259309 Jan  9 20:04 header_graphic.jpg
```

```
-rwxrwxr-x    1 1911913    cgi03            211 Jan 10 03:35 test2.html
```

>>> s.sendcmd('chmod 0664 test2.html')
#sendcmd is short for **send command** and here we send the command **chmod**, which is used to set file permissions (chmod works if the sever is Unix or Linux os, don't think this will work for servers running Microsoft software as operating system). chmod 0664 will set permissions to -rw-rw-r--

'200 Permissions changed on test2.html'
#the message above returned by server indicates we were successful in changing permissions, but we will also check by doing another directory list (I list only the line for the file):

>>> s.dir()
#below we see that we were successful in changing the file permissions

```
-rw-rw-r--    1 1911913    cgi03            211 Jan 10 03:35 test2.html
```

Now let us look at various commands we can issue while we have an open connection to the FTP server.

>>> s.getwelcome()
#this command returns the Welcome message put in place by Yahoo for their FTP server. The message is below. Notice that there are no anonymous logins allowed (i.e., you must have a user name and password). Also notice that you will be disconnected if you don't issue any commands over a period of 5 minutes. If you are disconnected, you will have to start over with importing FTP_TLS, etc.

'220-Welcome to the Yahoo! Web Hosting FTP server\n220-Need help? Get all details at:\n220-http://help.luminate.com/help/us/webhosting/gftp/\n220- \n220-No anonymous logins accepted.\n220-Yahoo!\n220-Local time is now 08:52. Server port: 21.\n220-This is a private system - No anonymous login\n220-IPv6 connections are also welcome on this server.\n220 You will be disconnected after 5 minutes of inactivity.'

Now we will set up for the upload of a file to the server.

>>> file = '/home/pi/LaFavre_web_pages/test2.html'
#here we create a variable named **file** which points to the file on the Raspberry Pi we want to upload

>>> ftppath = ''

#now we create a variable **ftppath**, which is the path on the server where the file will be uploaded. In this case there is nothing between the quotes, so the upload will be to the root directory

>>> filename = "test2.html"

filename will be the variable that holds the name to be applied to the file that is uploaded to the server. If we use a name other than the file name on the Raspberry Pi, then the file will have a different name on the server.

>>> s.cwd(ftppath)

cwd is used to set the current directory on the sever, and in this case will set it to the root directory

'250 OK. Current directory is /'

#message above returned from server confirms that root is set as the current directory

>>> f = open(file, 'rb')

#now we open the file on the Raspberry Pi to read its contents. We must read the file in binary format 'rb'. Contents of the file will be assigned to the variable named **f**

>>> s.storlines('STOR ' + filename, f)

#if we wish to upload a text file (those with file extensions .txt, .htm, .html) then we must use **storlines**. For other files, like .jpg images, you will need to use **storbinary**.

'226-File successfully transferred\n226 0.107 seconds (measured here), 1.93 Kbytes per second'

#the message returned from FTP server indicates that file transfer was successful

>>> s.dir()

#now when we issue a directory call, we see the file has the wrong permissions again

```
-rwxrwxr-x    1 1911913    cgi03            211 Jan 10 09:06 test2.html
```

Ok, the line below will fix the permissions

>>> s.sendcmd('chmod 0664 test2.html')

'200 Permissions changed on test2.html'

#we see from the server message the permissions were changed

```
>>> s.dir()
```
#issuing a directory call again confirms the change in permissions for test2.html

```
-rw-rw-r--    1 1911913    cgi03             211 Jan 10 09:06 test2.html
```

What you have learned so far can be very powerful as part of a Python program to upload data to the GEAR web site. When FTP is included in a Python program, the uploading can be automated. For example, if your Raspberry Pi is periodically taking temperature measurements and storing them in a web page, those web pages can be automatically uploaded to the web site.

OK, let us go through some other commands you can issue

```
>>> s.pwd()
```
#if you want to know the directory that is currently set, issue the pwd command

```
'/'
```
#in this case server returns message above indicating current directory is the root

```
>>> s.cwd("Grant")
```
#on our server there is a subdirectory of root named Grant. The cwd command sets the current directory on the server.

```
'250 OK. Current directory is /Grant'
```
#above is message returned by server

```
>>> s.pwd()
```
#again we ask for the path to current directory

```
'/Grant'
```
#server returns message indicating path to current directory

```
>>> s.mkd('/TEST')
```
mkd is the command to make a subdirectory of the root directory on the server. In this case the subdirectory will be named TEST.

```
'/TEST'
```
#server returns the above message

```
>>> s.delete('test2.html')
```

#the file test2.html is located at the root directory and the current directory is also the root directory. The above will delete the file test2.html on the server. Don't expect a warning message when you do this, it will just delete the file. Make sure you know what you are doing!

```
'250 Deleted test2.html'
```

#message returned by server – too late if that is not what you really wanted to do!!!

```
>>> s.rmd('TEST')
```

#use this to delete a directory (directory named TEST in this case)

```
'250 The directory was successfully removed'
```

#message from server indicates directory was removed

OK, we are almost finished. We need to close the f object and quit the FTP session. This is how it is done.

```
>>> f.close()
```

```
>>> s.quit()
```

```
'221-Goodbye. You uploaded 1 and downloaded 0 kbytes.\n221 Logout.'
```

#message returned by server when you quit the session

(see next page for file permission codes)

File permission codes - The access level is defined in numbers and complete code is three digits: owner, group, public:

0 - no access to the file

1 - execute only

2 - write only

3 - write and execute

4 - read only

5 - read and execute

6 - read and write

7 - read, write and execute (full permissions)

Here are some common codes used on servers

400	r--------	files (won't let you accidentally erase) – only owner has access and only to read, must change permission to delete or edit
444	r--r--r--	files (lets everyone read)
600	rw-------	files (no one else can read or see files) – only owner has access, can read and write
644	rw-r--r--	files owner can read and write, others can only read
664	rw-rw-r--	files owner and group can read and write, public can only read
666	rw-rw-rw-	files everybody can read and write – probably should not be used in most cases
700	rwx------	programs and directories only owner can read write and execute
750	rwxr-x---	programs and directories owner full access, group read and execute, public no access
755	rwxr-xr-x	programs and directories owner full access, group and public read and execute only
777	rwxrwxrwx	programs and directories everybody has full access – probably should not be used in most cases