[How to use the Pexpect Module](http://www.pythonforbeginners.com/systems-programming/how-to-use-the-pexpect-module-in-python/)

28 Jan 2013   | [System & OS](http://www.pythonforbeginners.com/category/systems-programming/)

This article is based on documentation from <http://www.noah.org/wiki/pexpect> and<http://pypi.python.org/pypi/pexpect/>

The reason I started to use Pexepect was because I was looking for a module that can take care of some of the automation needs I have (mostly with ssh and ftp).

You can use other modules such as [subprocess](http://docs.python.org/2/library/subprocess.html), but I find this module easier to use.

Note, this post is not for a Python beginner, but hey it’s always fun to learn new things

What is Pexpect?

Pexpect is a pure Python module that makes Python a better tool for controlling

and automating other programs.

Pexpect is basically a pattern matching system. It runs program and watches

output.

When output matches a given pattern Pexpect can respond as if a human were

typing responses.

What can Pexpect be used for?

Pexpect can be used for automation, testing, and screen scraping.

Pexpect can be used for automating interactive console applications such as

ssh, ftp, passwd, telnet, etc.

It can also be used to control web applications via `lynx`, `w3m`, or some

other text-based web browser.

Installing Pexpect

The latest version of Pexpect can be found [here](http://sourceforge.net/projects/pexpect/files/pexpect/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | wget http://pexpect.sourceforge.net/pexpect-2.3.tar.gz  tar xzf pexpect-2.3.tar.gz  cd pexpect-2.3  sudo python ./setup.py install    # If your systems support yum or apt-get, you might be able to use the  # commands below to install the pexpect package.    sudo yum install pexpect.noarch    # or    sudo apt-get install python-pexpect |

Pexpect Methods

There are two important methods in Pexpect: expect() and send() (or sendline()

which is like send() with a linefeed).

The expect() method

Waits for the child application to return a given strong.

The string you specify is a regular expression, so you can match complicated

patterns.

emember that any time you try to match a pattern that needs look-ahead that

you will always get a minimal match.

**The following will always return just one character:**

child.expect ('.+')

Specify correctly the text you expect back, you can add '.\*' to the beginning

or to the end of the text you're expecting to make sure you're catching

unexpected characters

**This example will match successfully, but will always return no characters:**

child.expect ('.\*')

Generally any star \* expression will match as little as possible.

The pattern given to expect() may also be a list of regular expressions,

this allows you to match multiple optional responses.

(example if you get various responses from the server)

The send() method

Writes a string to the child application.

From the child's point of view it looks just like someone typed the text from

a terminal.

The before and after properties

After each call to expect() the before and after properties will be set to the

text printed by child application.

**The before property** will contain all text up to the expected string pattern.

You can use child.before to print the output from the other side of the connection

**The after string** will contain the text that was matched by the expected pattern.

**The match property** is set to the re MatchObject.

Connect and download a file from a remote FTP Server

This connects to the openbsd ftp site and downloads the recursive directory

listing.

You can use this technique with any application.

This is especially handy if you are writing automated test tools.

Again, this example is copied from [here](http://www.noah.org/wiki/Pexpect)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | import pexpect  child = pexpect.spawn ('ftp ftp.openbsd.org')  child.expect ('Name .\*: ')  child.sendline ('anonymous')  child.expect ('Password:')  child.sendline ('noah@example.com')  child.expect ('ftp> ')  child.sendline ('cd pub')  child.expect('ftp> ')  child.sendline ('get ls-lR.gz')  child.expect('ftp> ')  child.sendline ('bye') |

In the second example, we can see how to get back the control from Pexpect

Connect to a remote FTP Server and get control

This example uses ftp to login to the OpenBSD site (just as above),

list files in a directory and then pass interactive control of the ftp session

to the human user.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | import pexpect  child = pexpect.spawn ('ftp ftp.openbsd.org')  child.expect ('Name .\*: ')  child.sendline ('anonymous')  child.expect ('Password:')  child.sendline ('noah@example.com')  child.expect ('ftp> ')  child.sendline ('ls /pub/OpenBSD/')  child.expect ('ftp> ')  print child.before    # Print the result of the ls command.  child.interact()       # Give control of the child to the user. |

EOF, Timeout and End Of Line

There are special patters to match the End Of File or a Timeout condition.

I will not write about it in this article, but refer to the official [documentation](http://www.noah.org/wiki/pexpect)

because it is good to know how it works.