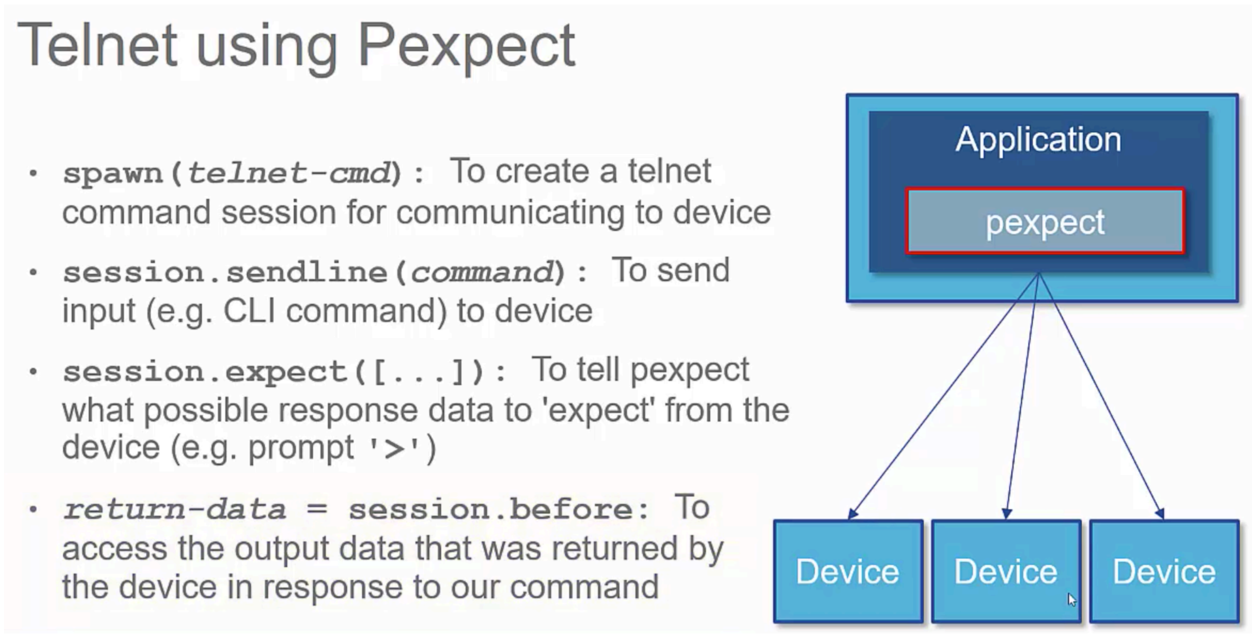


Telnet using Pexpect

- **spawn(*telnet-cmd*):** To create a telnet command session for communicating to device
- **session.sendline(*command*):** To send input (e.g. CLI command) to device
- **session.expect([...]):** To tell pexpect what possible response data to 'expect' from the device (e.g. prompt '>')
- ***return-data* = session.before:** To access the output data that was returned by the device in response to our command

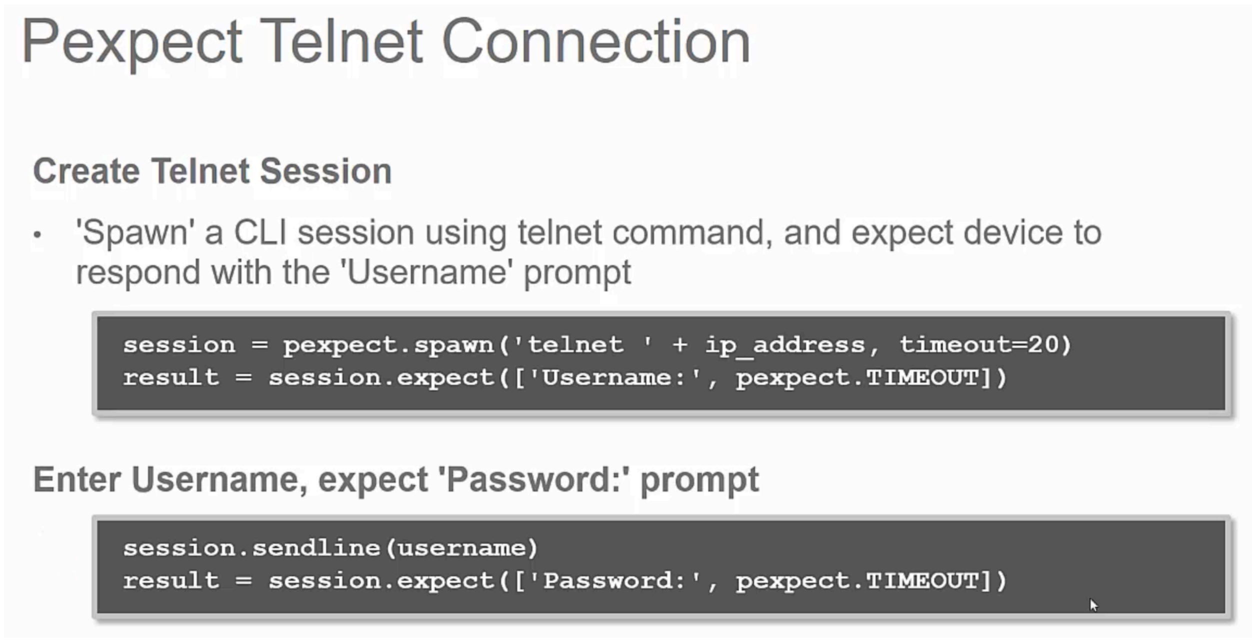
Import Pexpect

Pexpect is not one of the built-in functions for Python, so you will have to 'import' it into your program.

```
import pexpect
```

The import command tells Python that your code may be using the functions, objects, and methods that come with the Pexpect library. Your code will be connecting to a network device at a specific IP address, using a specific username and password. In the Pexpect examples that follow, this information will be stored in variables:

```
ip_address = '10.30.30.1'
username = 'cisco'
password = 'cisco'
```



Pexpect Telnet Connection

Create Telnet Session

- 'Spawn' a CLI session using telnet command, and expect device to respond with the 'Username' prompt

```
session = pexpect.spawn('telnet ' + ip_address, timeout=20)
result = session.expect(['Username:', pexpect.TIMEOUT])
```

Enter Username, expect 'Password:' prompt

```
session.sendline(username)
result = session.expect(['Password:', pexpect.TIMEOUT])
```

Connect Via Telnet Session

You will create ('spawn') a Pexpect session using the `pexpect.spawn(...)` method:

```
# Create Pexpect telnet session
session = pexpect.spawn('telnet ' + ip_address, timeout=20)
```

The `spawn` method creates a new session, using the command given inside the quotation marks. In this case, the command is 'telnet', passing in the IP address of the destination network device. The `expect` method tells the program what to expect as a result of the connection request. The following code shows how to 'expect' a response to the connection request and store the response in a variable named 'result':

```
result = session.expect(['Username:', pexpect.TIMEOUT])

# Check for error, if so then print error and exit
if result != 0:
    print '--- FAILURE! creating session for: ', ip_address
    exit()
```

In the example, the session object expects the device to respond with either 'Username:' or a timeout. If the result is not equal to 0 – the first item in our list of possible responses – then that indicates an error, and the code prints an error message and exits. If the result is 0, the telnet connection was successful and the process continues by entering the username and password. Your application must send the login credentials for the device which is done using the `sendline` command, passing a string containing the value of your desired command. For the login process, the code would look as follows:

```
# Session expecting username, enter it here
session.sendline(username)
result = session.expect(['Password:', pexpect.TIMEOUT])

# Check for error, if so then print error and exit
if result != 0:
    print '--- FAILURE! entering username: ', username
    exit()

# Session expecting password, enter it here
session.sendline(password)
result = session.expect(['>', pexpect.TIMEOUT])

# Check for error, if so then print error and exit
if result != 0:
    print ' FAILURE! entering password: ', password
    exit()

print '--- Success! connecting to: ', ip_address
print '---               Username: ', username
print '---               Password: ', password
print '------------------------------------------------------\n'
```

In the example, the application sends the username, expecting to receive the 'Password' prompt, then sends the password, expecting the CLI prompt ('>') for an IOS device. If everything works successfully, a message to that effect is printed.

Sending CLI Requests

The general model of Pexpect interaction is as follows:

- Issue a command
- Tell Pexpect to 'expect' a list of potential responses
- Take action based on the actual response

For example:

```
# Send 'show interface' command to gather data about links
session.sendline('show interface summary')   # Send 'show interface' command
result = session.expect(['>', pexpect.TIMEOUT, pexpect.EOF])
if result == 0:    # successful reply to show interface command
    print 'Command sent successfully'
    return True
elif result == 1:  # timeout occurred
    print 'Timed out'
    return False
elif result == 2:   # EOF occurred
    print 'Received unexpected EOF response'
    return False
else:
    print 'Unexpected response'
    return False
```

The `elif` statement is used to select between multiple options, similar to the `case` statement in other languages.

Receiving CLI Responses

Once your CLI request has completed successfully, your application will often receive output data. Output data is provided to your application in the 'before' member variable of your session:

```
show_int_output = session.before
```

The output is received in the form of a string. If the output consists of multiple lines, your application may use the `splitlines()` function to separate the string into a list of strings, one item for each line of output.

```
int_output_lines = show_int_output.splitlines()
```

If the output is more complex, it may require parsing using something such as regular expressions.

Close Connection

When you are done interacting with your device, you should send the appropriate CLI commands to exit the Telnet connection on the device. You should also close the session within the application to free up local resources..

```
session.close()
```