# Python CyberStart Level 3

Challenges 1 – 5 in CyberStart Moon Base Level 3 begin to touch on some fun topics: reading and writing files, network communications, HTTP, and regular expressions. They are very basic, however, so do not be alarmed. Python information is available in the CyberStart Field Manual, or in the free online book, Automate the Boring Stuff with Python <https://automatetheboringstuff.com/> (scroll down to see the table of contents.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Title | Subject | Field Manual | Automate book |
| 1 | Fun with Files | open, close, read, write files | 5.12 Reading Writing Files | Ch 9 Read Write Fil |
| 2 | Sockets and Servers | network tcp socket | n/a | n/a |
| 3 | Don't Forget HTTP | urllib, http server | n/a | n/a |
| 4 | ASCII Encoding | chr, ord | 3.5.1 ASCII |  |
| 5 | Ready for Regex | re.findall() | n/a | n/a |

Please complete the Level 3 challenges. There is no other hand-in for this lab, so if you have joined our class group in CyberStart (access code is email-ocean-pilot) I'll see your work and you will not need to turn anything in to Canvas.

Note: Sometimes CyberStart can be picky about the answer it wants to see. If CyberStart won't give you credit when you are sure you are correct, run the code in your own Python. Then send me the results via email through Canvas.

## Notes for Level 3

### Challenge 1

The with statement is one of the first where you see the way that Python groups statements. The statement that starts the group always ends with a colon ':'. Statements that are part of that group are all indented. You can use either spaces or tabs, but you must be consistent. A mix of spaces and tabs will crash your program.

### Challenge 2

The socket module has some strange syntax  
socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
but it is the way that Python connects to a network. You will recognize this code in anything the uses a network.

Note that when you send data using the socket module, it always uses bytes and not strings. If you want to send a string you must convert it to bytes first. For example,  
'knock, knock'  
will generate an error but  
'knock, knock'.encode()  
will not. Execute these statements to show you how they are different.  
type('knock, knock')  
type('knock, knock'.encode())

When you print output from a socket it will appear with a 'b' in front to denote that it is in byte format.  
b'this is the output'

### Challenge 3

This challenge demonstrates the Python urllib module, which performs HTTP and HTTPS transactions. I am not sure why they use urllib instead of the current version, urllib3, or the easier to use requests module, but they do. If you are familiar with the other modules, use urllib anyway as that is all their web page understands.

This challenge is very simple; just make minor changes to their example.

Your output will be prettier if you convert it from bytes to strings, but you get credit for either. To convert, use the decode() function.  
print(myoutput.decode())

### Challenge 4

This is a simple challenge converting integers to characters and using a for loop. The for loop was introduced in L2C5.

### Challenge 5

This challenge introduces regular expressions, which are wild cards on steroids. Some basic syntax:  
. A period is the wild card for a single character.

.\* This means any character (the .) and can occur any number of times, including zero.

() Parentheses tell the regex to return what is inside them. It is not matching parentheses.

It took me a while to figure out just what they wanted. It turned out that if a line was  
 <div class='randomDiv'>Hello</div>  
the answer they want is randomDiv.