LAB – PYDOC AND PYDOCSTYLE

OBJECTIVE

In this lab, you will use pydoc to generate documentation and pydocstyle to ensure that the documentation conforms to the Python Enhancement Proposals (PEP) which aims to standardise docstring conventions, PEP 257, for best practise.

NOTE:
More detail on the programs and features used in this lab can be found at:

- pydoc - https://docs.python.org/3/library/pydoc.html
- pydocstyle – https://pypi.org/project/pydocstyle/
- pydocstyle error codes - http://www.pydocstyle.org/en/stable/error_codes.html
- PEP 257 - https://peps.python.org/pep-0257/

PART 1

In this part you will open a terminal and switch to the lab directory.

STEP 1: OPEN A TERMINAL WINDOW

Double-click the Terminal icon on the desktop to open the terminal window for use in this lab.

STEP 2: CHANGE DIRECTORY

Change to the directory **labs/prne/** in the user home directory, which holds the files for the course labs.

```
~$ cd labs/prne/
```

PART 2

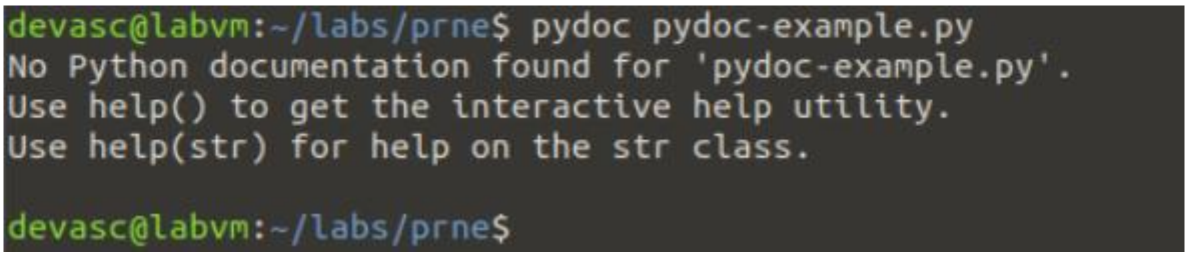
In this part you will view the what documentation is currently available for the module and its compliance with convention.

STEP 1: CHECK DOCUMENTATION STATUS

Check the status of documentation for the **pydoc-example.py** module using the command below from the terminal window:

```
~/labs/prne$ pydoc pydoc-example.py
```

Verify that the command was successful and the output is comparable to below.



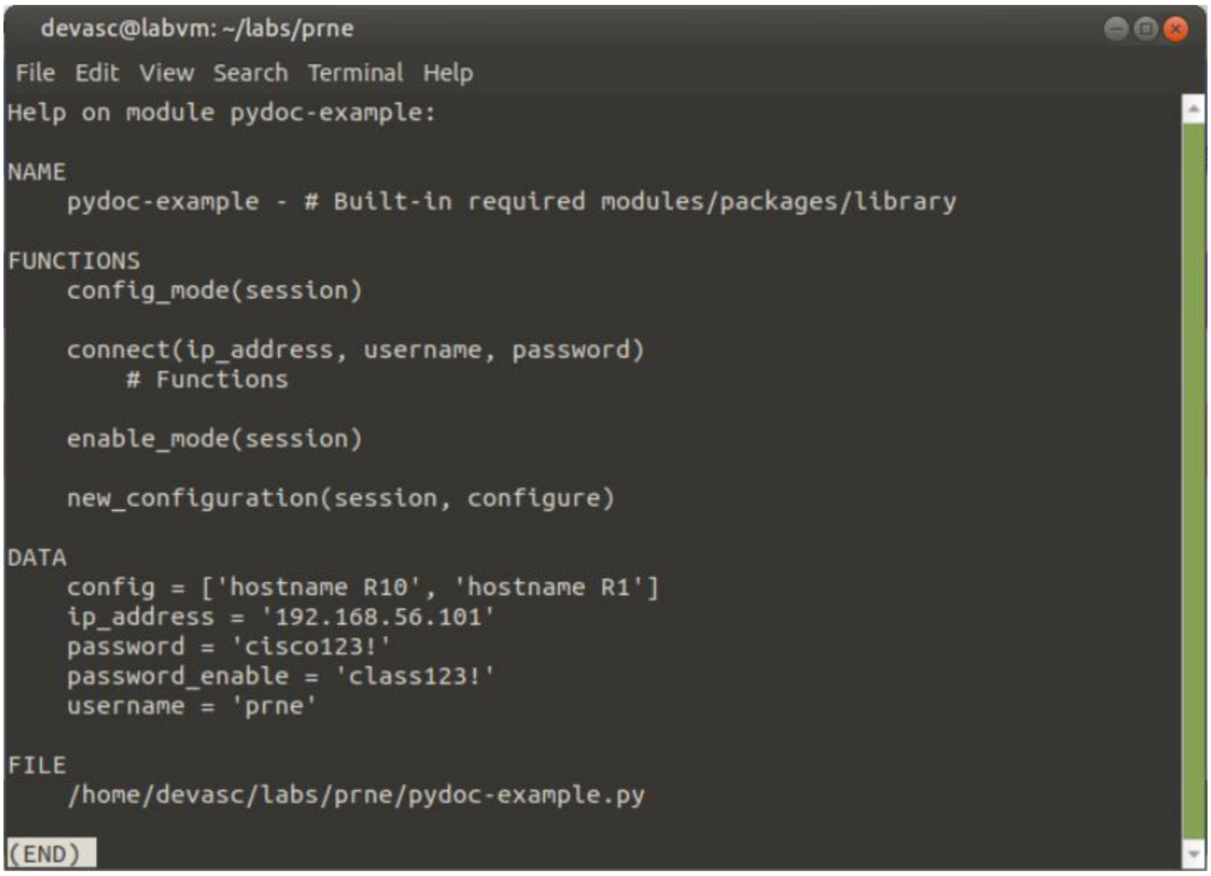
```
devasc@labvm:~/labs/prne$ pydoc pydoc-example.py
No Python documentation found for 'pydoc-example.py'.
Use help() to get the interactive help utility.
Use help(str) for help on the str class.

devasc@labvm:~/labs/prne$
```

As can be seen in the output, there is no documentation for the module.
Although that may not be correct, depending on the version of **Python** and/or **pydoc** it may just require a slight modification of the command to show any documentation that is currently in the module.
To check using the modified command, run the command below from the terminal window:

```
~/labs/prne$ pydoc pydoc-example
```

NOTE:
It is the same command just **without** the extension (**.py**) for the file.


devasc@labvm: ~/labs/prne
File Edit View Search Terminal Help

```
Help on module pydoc-example:

NAME
    pydoc-example - # Built-in required modules/packages/library

FUNCTIONS
    config_mode(session)

    connect(ip_address, username, password)
        # Functions

    enable_mode(session)

    new_configuration(session, configure)

DATA
    config = ['hostname R10', 'hostname R1']
    ip_address = '192.168.56.101'
    password = 'cisco123!'
    password_enable = 'class123!'
    username = 'prne'

FILE
    /home/devasc/labs/prne/pydoc-example.py

(END)
```

From looking at the documentation that is now shown, you can see that there is no summary as to what the module does, or what the arguments of the functions are and what each function does. However, you can see the module name, path, functions created or data that is held within the code.

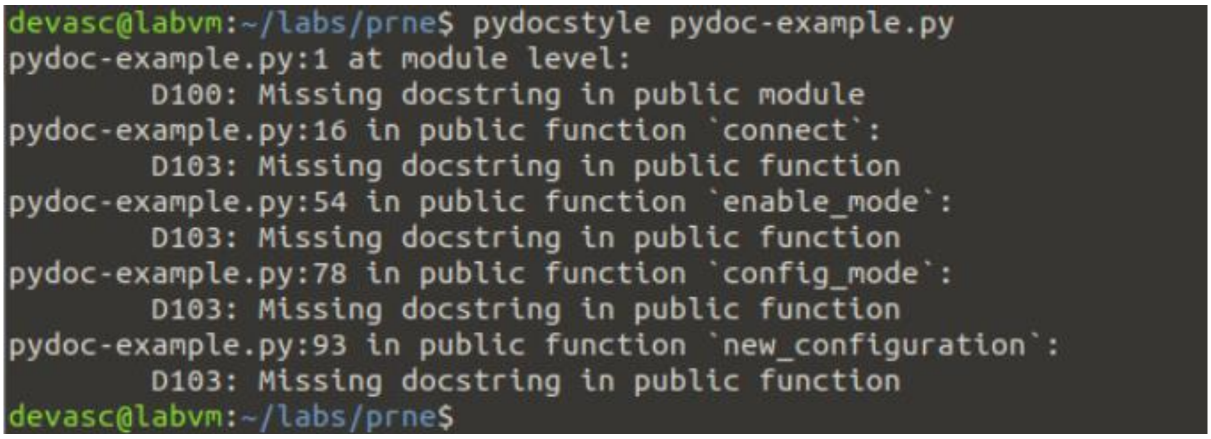
To exit this information, press the key **q** to return to the command prompt.

STEP 2: CHECK DOCUMENTATION CONVENTION COMPLIANCE

As you have already seen, there is little to no documentation in the current module. One option that we have, is to use another tool to check the module against the documentation convention detailed in PEP 257.
To view the current status for the module regarding compliance with the convention, use the command below from the terminal window:

```
~/labs/prne$ pydocstyle pydoc-example.py
```

Verify that the command was successful and the output is comparable to below.



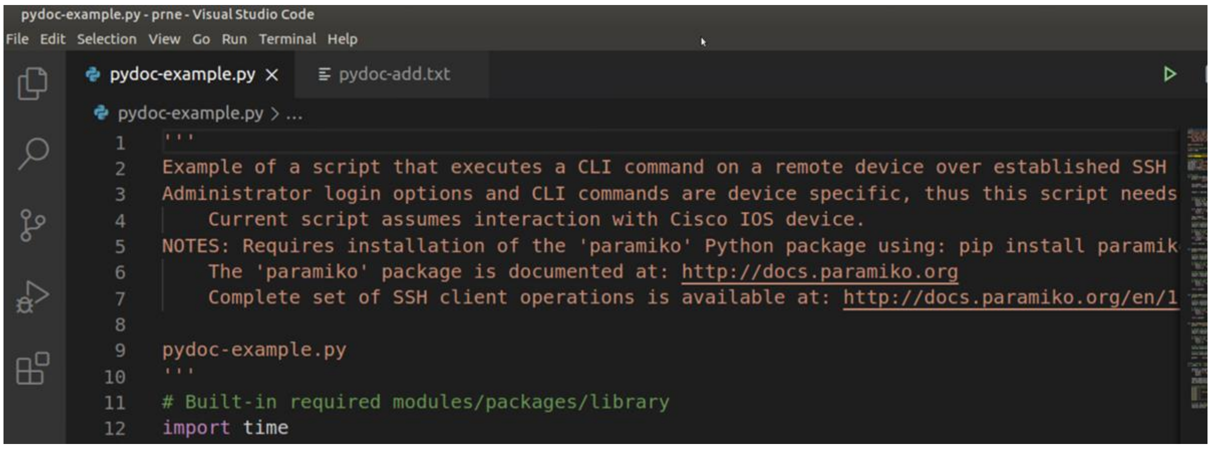
```
devasc@labvm:~/labs/prne$ pydocstyle pydoc-example.py
pydoc-example.py:1 at module level:
        D100: Missing docstring in public module
pydoc-example.py:16 in public function `connect`:
        D103: Missing docstring in public function
pydoc-example.py:54 in public function `enable_mode`:
        D103: Missing docstring in public function
pydoc-example.py:78 in public function `config_mode`:
        D103: Missing docstring in public function
pydoc-example.py:93 in public function `new_configuration`:
        D103: Missing docstring in public function
devasc@labvm:~/labs/prne$
```

PART 3

In this part, you will add some basic docstrings for documentation.
Open the files **pydoc-example.py** and **pydoc-add.txt** located in the **~/labs/prne/** directory in **Visual Studio Code**, this will allow the addition of the comments from the text file to the module for the documentation.
Comments for documentation are added as a **docstring** using either one-line docstring or a multi-line docstring. A docstring is a string literal that occurs as the first statement in a module, function, class, or method definition. This type of docstring becomes the **__doc__** special attribute for that object.

STEP 1: ADD DESCRIPTION COMMENT

Add the module **description** comment (lines 5-14 in the **pydoc-add.txt**) by copy and pasting them into the **pydoc-example.py** at line **1**, verifying the output should be comparable to below.

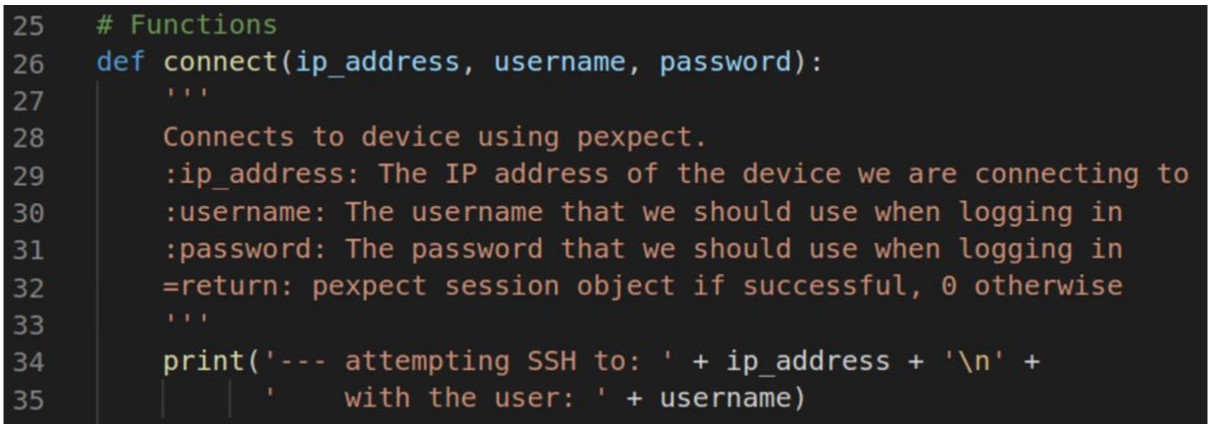

pydoc-example.py - prne - Visual Studio Code
File Edit Selection View Go Run Terminal Help
pydoc-example.py
pydoc-add.txt
pydoc-example.py > ...

```
'''
Example of a script that executes a CLI command on a remote device over established SSH
Administrator login options and CLI commands are device specific, thus this script needs
    Current script assumes interaction with Cisco IOS device.
NOTES: Requires installation of the 'paramiko' Python package using: pip install paramik
    The 'paramiko' package is documented at: http://docs.paramiko.org
    Complete set of SSH client operations is available at: http://docs.paramiko.org/en/1

pydoc-example.py
'''
# Built-in required modules/packages/library
import time
```

STEP 2: ADD CONNECT FUNCTION COMMENT

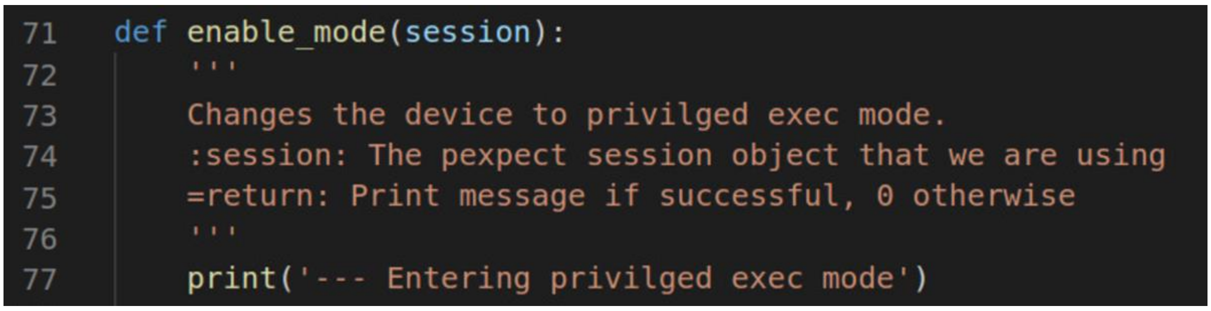
Add the **connect** function comment (lines 26-32 in the **pydoc-add.txt**) by copy and pasting them into the **pydoc-example.py** at line **27**, verifying the output should be comparable to below.



```
25  # Functions
26  def connect(ip_address, username, password):
27      '''
28      Connects to device using pexpect.
29      :ip_address: The IP address of the device we are connecting to
30      :username: The username that we should use when logging in
31      :password: The password that we should use when logging in
32      =return: pexpect session object if successful, 0 otherwise
33      '''
34      print('--- attempting SSH to: ' + ip_address + '\n' +
35              '     with the user: ' + username)
```

STEP 3: ADD ENABLE_MODE FUNCTION COMMENT

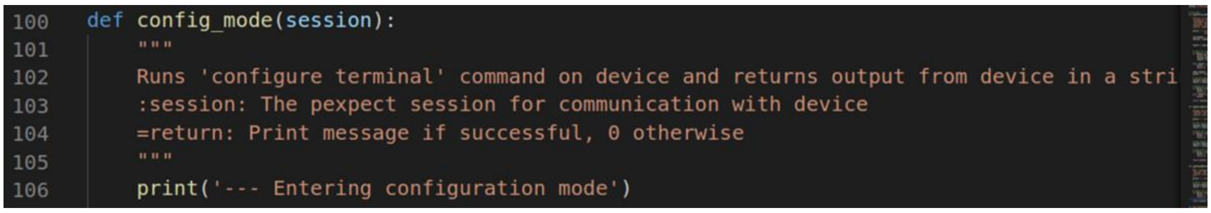
Add the **enable_mode** function comment (lines 37-41 in the **pydoc-add.txt**) by copy and pasting them into the **pydoc-example.py** at line **72**, verifying the output should be comparable to below.



```
71  def enable_mode(session):
72      '''
73      Changes the device to privilged exec mode.
74      :session: The pexpect session object that we are using
75      =return: Print message if successful, 0 otherwise
76      '''
77      print('--- Entering privilged exec mode')
```

STEP 4: ADD CONFIG_MODE FUNCTION COMMENT

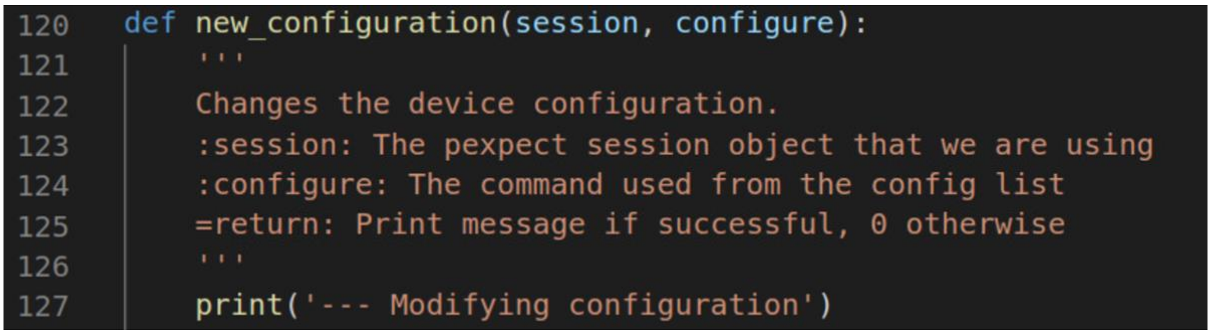
Add the **config_mode** function comment (lines 46-50 in the **pydoc-add.txt**) by copy and pasting them into the **pydoc-example.py** at line **101**, verifying the output should be comparable to below.



```
100 def config_mode(session):
101     """
102     Runs 'configure terminal' command on device and returns output from device in a stri
103     :session: The pexpect session for communication with device
104     =return: Print message if successful, 0 otherwise
105     """
106     print('--- Entering configuration mode')
```

STEP 5: ADD NEW_CONFIGURATION FUNCTION COMMENT

Add the **new_configuration** function comment (lines 55-60 in the **pydoc-add.txt**) by copy and pasting them into the **pydoc-example.py** at line **121**, verifying the output should be comparable to below.



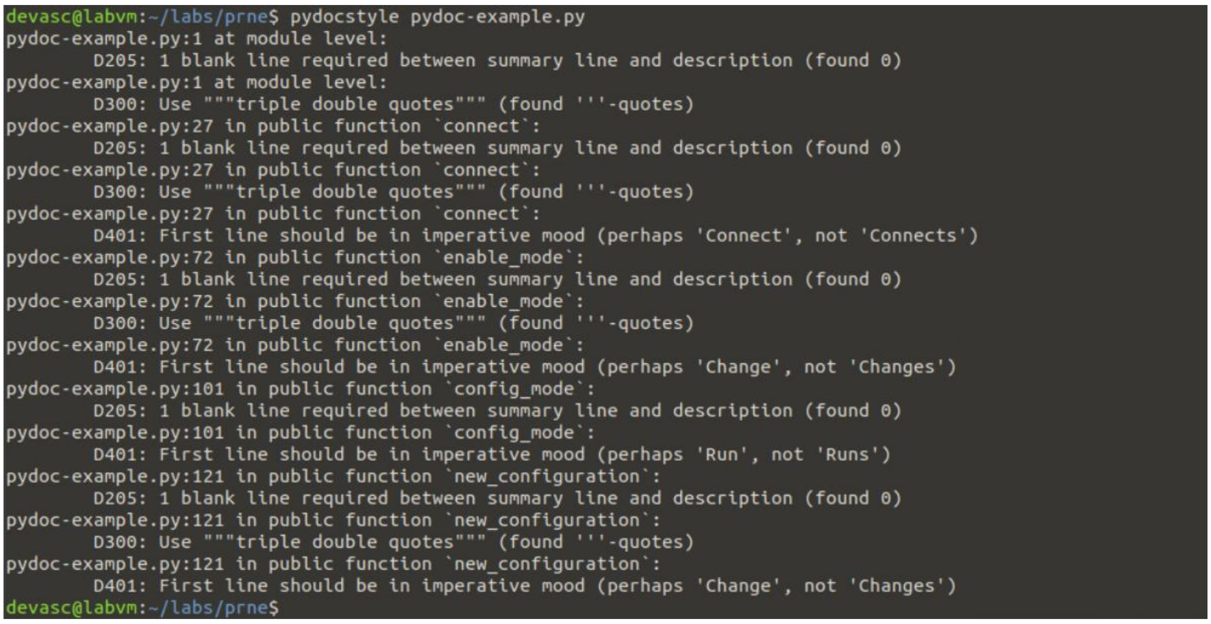
```
120  def new_configuration(session, configure):
121      '''
122      Changes the device configuration.
123      :session: The pexpect session object that we are using
124      :configure: The command used from the config list
125      =return: Print message if successful, 0 otherwise
126      '''
127      print('--- Modifying configuration')
```

STEP6: CHECK DOCUMENTATION CONVENTION COMPLIANCE

Now that you have added some documentation to the module, it’s now time to re-check the documentation convention detailed in PEP 257.
Ensure that you have saved the file and then to view the current status for the module regarding compliance with the convention, use the command below from the terminal window:

```
~/labs/prne$ pydocstyle pydoc-example.py
```

Verify that the command was successful and the output is comparable to below.



```
devasc@labvm:~/labs/prne$ pydocstyle pydoc-example.py
pydoc-example.py:1 at module level:
        D205: 1 blank line required between summary line and description (found 0)
pydoc-example.py:1 at module level:
        D300: Use """triple double quotes""" (found '''-quotes)
pydoc-example.py:27 in public function `connect`:
        D205: 1 blank line required between summary line and description (found 0)
pydoc-example.py:27 in public function `connect`:
        D300: Use """triple double quotes""" (found '''-quotes)
pydoc-example.py:27 in public function `connect`:
        D401: First line should be in imperative mood (perhaps 'Connect', not 'Connects')
pydoc-example.py:72 in public function `enable_mode`:
        D205: 1 blank line required between summary line and description (found 0)
pydoc-example.py:72 in public function `enable_mode`:
        D300: Use """triple double quotes""" (found '''-quotes)
pydoc-example.py:72 in public function `enable_mode`:
        D401: First line should be in imperative mood (perhaps 'Change', not 'Changes')
pydoc-example.py:101 in public function `config_mode`:
        D205: 1 blank line required between summary line and description (found 0)
pydoc-example.py:101 in public function `config_mode`:
        D401: First line should be in imperative mood (perhaps 'Run', not 'Runs')
pydoc-example.py:121 in public function `new_configuration`:
        D205: 1 blank line required between summary line and description (found 0)
pydoc-example.py:121 in public function `new_configuration`:
        D300: Use """triple double quotes""" (found '''-quotes)
pydoc-example.py:121 in public function `new_configuration`:
        D401: First line should be in imperative mood (perhaps 'Change', not 'Changes')
devasc@labvm:~/labs/prne$
```

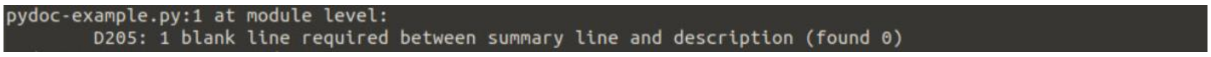
Now that you have added some documentation, you can see that there are various issues regarding how the documentation is formatted in regards to the PEP 257 convention.

PART 4

In this part, you will resolve all of the documentation convention issues, so that it conforms to PEP 257.
With the file **pydoc-example.py** located in the **~/labs/prne/** directory open in **Visual Studio Code**, adjust the code so that all error codes are resolved.

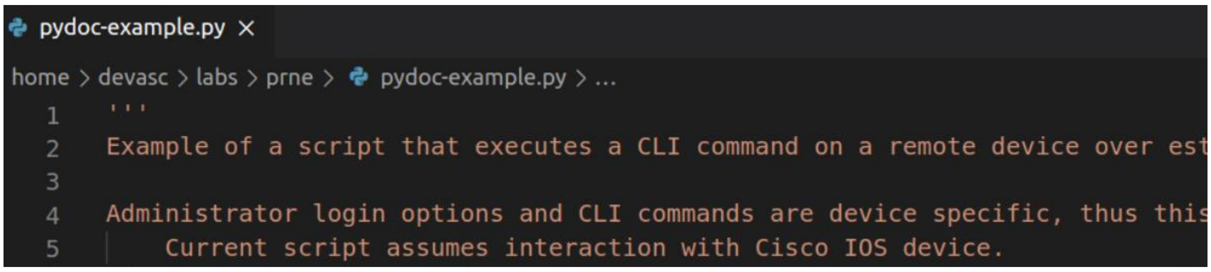
STEP 1: CORRECT THE FIRST D205 ERROR CODE

Looking at the first error code in the last check of the document, the error code is **D205** and is on **line 1**.



```
pydoc-example.py:1 at module level:
        D205: 1 blank line required between summary line and description (found 0)
```

From the description, you would expect the formatting error to be on line 1, however, as it's an error in the docstring comment, it can be anywhere within the full comment (lines 1-10). In this case, the error is between the summary line and the description, as such the actual error is between lines 2-3.
The summary line may be used by automatic indexing tools, as such it's important that it fits on one line and is separated from the rest of the docstring by a blank line.
To resolve this error, add a blank line between those lines, verifying the output should be comparable to below.

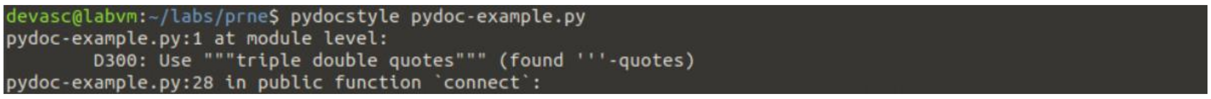


```
1  '''
2  Example of a script that executes a CLI command on a remote device over est
3
4  Administrator login options and CLI commands are device specific, thus this
5      Current script assumes interaction with Cisco IOS device.
```

Once you have saved the file, it's time to recheck module for compliance to ensure that the error is fixed, using the command below from the terminal window:

```
~/labs/prne$ pydocstyle pydoc-example.py
```

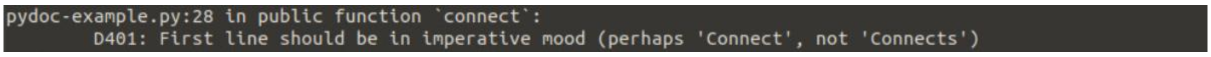
Verify that the command was successful and the output is comparable to below.



```
devasc@labvm:~/labs/prne$ pydocstyle pydoc-example.py
pydoc-example.py:1 at module level:
        D300: Use """triple double quotes""" (found '''-quotes)
pydoc-example.py:28 in public function `connect`:
```

STEP 2: CORRECT THE FIRST D401 ERROR CODE

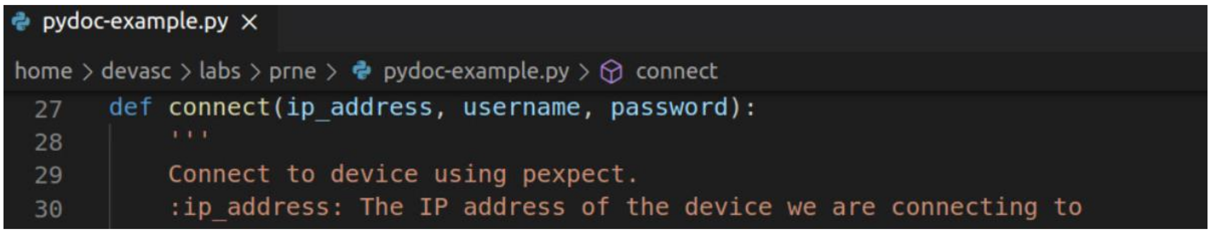
Looking at the first **D401** error code, the error code is on **line 28**, within the **connect** function.



```
pydoc-example.py:28 in public function `connect`:
        D401: First line should be in imperative mood (perhaps 'Connect', not 'Connects')
```

From the description, you would expect the formatting error to be on line 28, however, again, as it's an error in the docstring comment, it can be anywhere within the full comment (lines 28-34). In this case, the error is in the summary line, as such the actual error is on line 29.
The issue in this instance is that the summary line is not written in imperative mood, in other words, it prescribes the function or method's effect as a command ("Do this", "Return that"), not as a description.

To resolve this error, in this case it gives an option of how to resolve the issue in the error message. As such, you will use the suggestion to resolve the error by just removing the **s** at the end of the word **Connect**, verifying the output should be comparable to below.

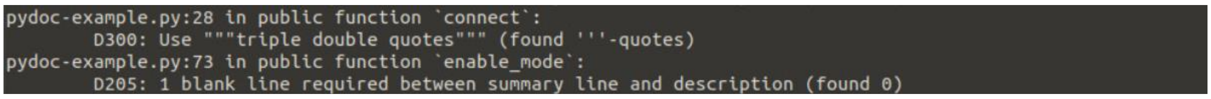


```
def connect(ip_address, username, password):
    '''
    Connect to device using pexpect.
    :ip_address: The IP address of the device we are connecting to
```

Once you have saved the file, it’s time to recheck module for compliance to ensure that the error is fixed, using the command below from the terminal window:

```
~/labs/prne$ pydocstyle pydoc-example.py
```

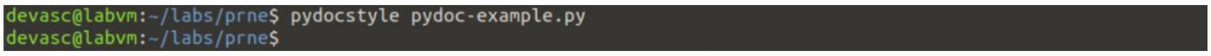
Verify that the command was successful and that the error has been resolved, with the output now comparable to below.



```
pydoc-example.py:28 in public function `connect`:
        D300: Use """triple double quotes""" (found '''-quotes)
pydoc-example.py:73 in public function `enable_mode`:
        D205: 1 blank line required between summary line and description (found 0)
```

STEP 3: CHALLENGE

Continue going through the convention errors until you have resolved all errors in the document and the output is comparable to below.



```
devasc@labvm:~/labs/prne$ pydocstyle pydoc-example.py
devasc@labvm:~/labs/prne$
```

NOTE:
You don’t get any confirmation that the file is now compliant with the PEP 257 convention, but rather you just receive a new line without any errors being displayed.

PART 5

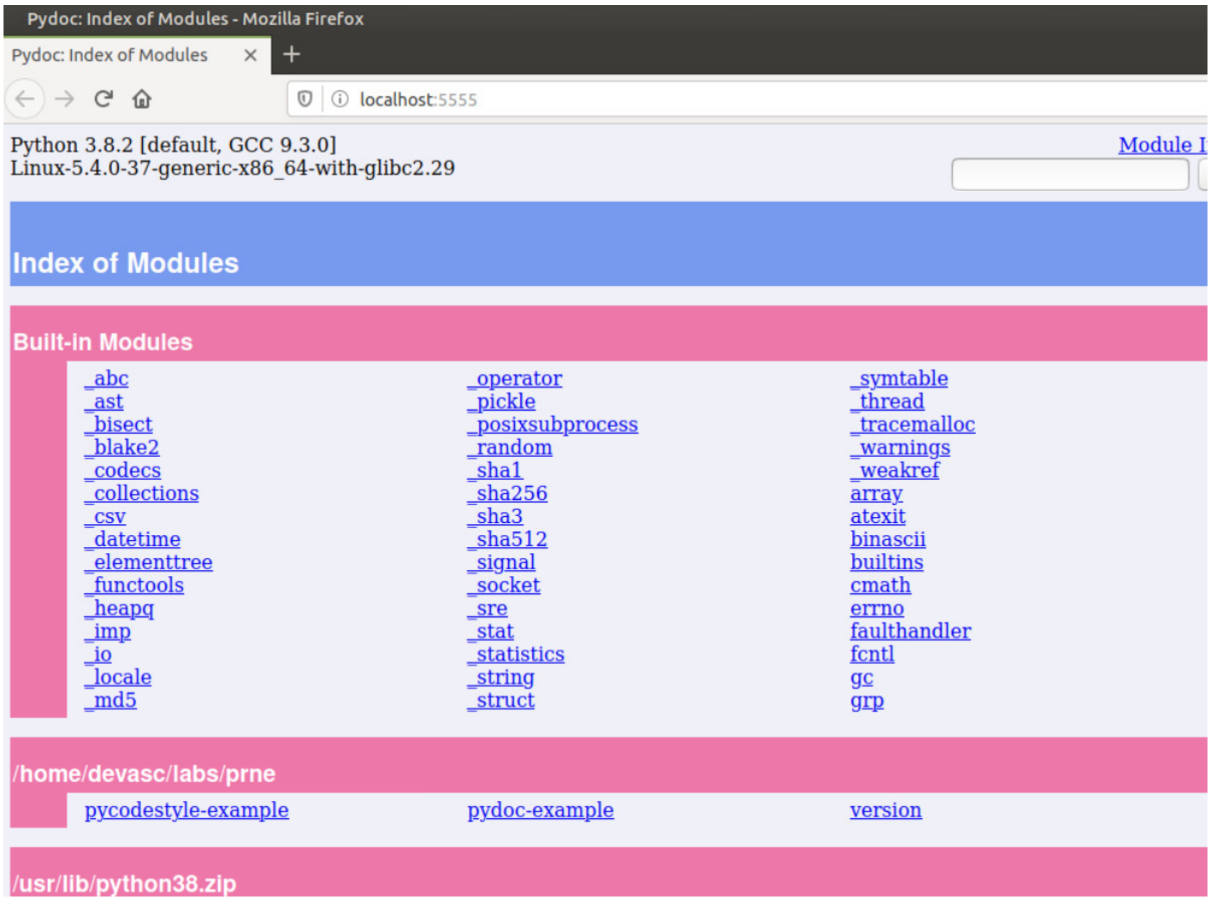
In this part, you will view the documentation in a web browser on the local machine that allows you to view detail for all modules and packages that are currently installed in your Python instance, this can help when you are writing code from scratch and need to see what options are available without having to view each module/package individually in a terminal window.
There are multiple options that can be used when starting the HTTP server on the local machine, these can be seen in the documentation for pydoc.

STEP 1: START HTTP SERVER

Using the command below from the terminal window, start the pydoc web server:

```
~/labs/prne$ pydoc -b
```

This will automatically start the web server on a random port and open up a web browser to the homepage.



Pydoc: Index of Modules - Mozilla Firefox

localhost:5555

Python 3.8.2 [default, GCC 9.3.0]
Linux-5.4.0-37-generic-x86_64-with-glibc2.29

Module I

Index of Modules

Built-in Modules

_abc	_operator	_symtable
_ast	_pickle	_thread
_bisect	_posixsubprocess	_tracemalloc
_blake2	_random	_warnings
_codecs	_sha1	_weakref
_collections	_sha256	array
_csv	_sha3	atexit
_datetime	_sha512	binascii
_elementtree	_signal	builtins
_functools	_socket	cmath
_heapq	_sre	errno
_imp	_stat	faulthandler
_io	_statistics	fcntl
_locale	_string	gc
_md5	_struct	grp

/home/devasc/labs/prne

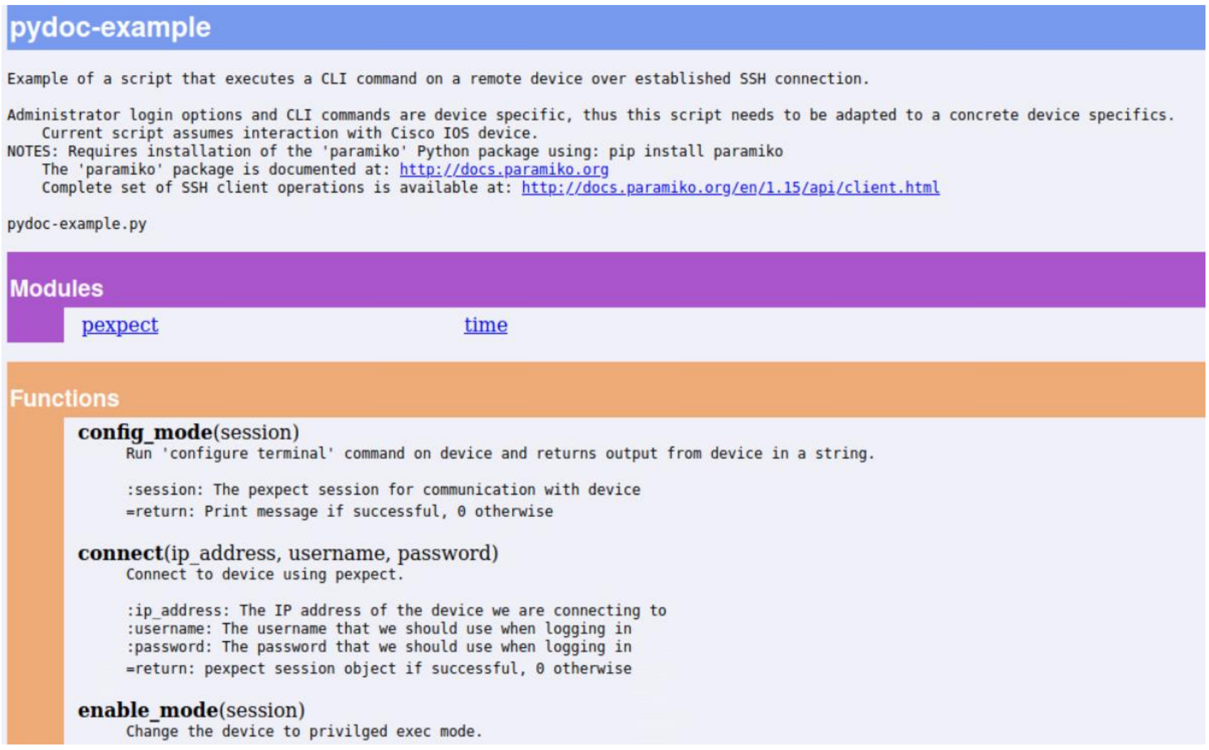
pycodestyle-example	pydoc-example	version

/usr/lib/python38.zip

NOTE:
This displays all of the modules and packages in the current paths available.

STEP 2: VIEW A MODULE

Under the section **/home/devasc/labs/prne**, select the module **pydoc-example**, that you have just ensured is fully compliant with the PEP 257 docstring conventions to view the documentation in an easier to read format.



pydoc-example

```
Example of a script that executes a CLI command on a remote device over established SSH connection.

Administrator login options and CLI commands are device specific, thus this script needs to be adapted to a concrete device specifics.
    Current script assumes interaction with Cisco IOS device.
NOTES: Requires installation of the 'paramiko' Python package using: pip install paramiko
    The 'paramiko' package is documented at: http://docs.paramiko.org
    Complete set of SSH client operations is available at: http://docs.paramiko.org/en/1.15/api/client.html

pydoc-example.py
```

Modules

pexpect time

Functions

config_mode(session)

```
    Run 'configure terminal' command on device and returns output from device in a string.

    :session: The pexpect session for communication with device
    =return: Print message if successful, 0 otherwise
```

connect(ip_address, username, password)

```
    Connect to device using pexpect.

    :ip_address: The IP address of the device we are connecting to
    :username: The username that we should use when logging in
    :password: The password that we should use when logging in
    =return: pexpect session object if successful, 0 otherwise
```

enable_mode(session)

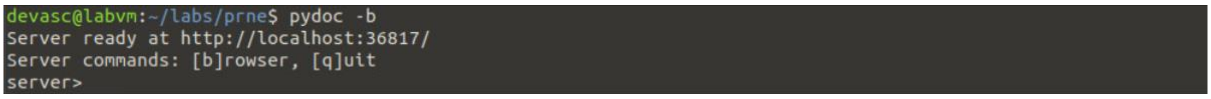
```
    Change the device to privilged exec mode.
```

STEP 3: VIEW OTHER MOUDLES

Return to the **index** page of the web server and review some of the other modules and packages to see how fully commented modules should look.

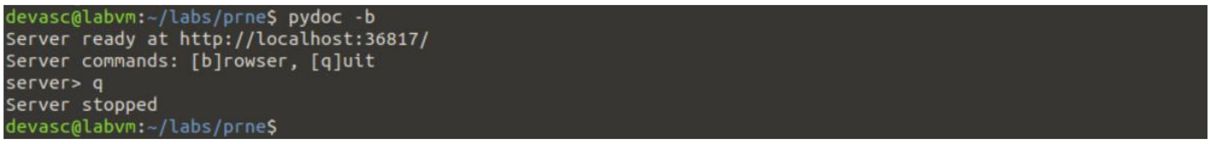
STEP 4: CLOSE THE WEB SERVER

Shutting down the web server should be done once you have finished using it. Firstly, close the web browser, you should now see the following in the terminal:



```
devasc@labvm:~/labs/prne$ pydoc -b
Server ready at http://localhost:36817/
Server commands: [b]rowser, [q]uit
server>
```

To exit the web server, press the key **q** to return to the command prompt.



```
devasc@labvm:~/labs/prne$ pydoc -b
Server ready at http://localhost:36817/
Server commands: [b]rowser, [q]uit
server> q
Server stopped
devasc@labvm:~/labs/prne$
```

PART 6 (OPTIONAL BUT HIGHLY RECOMMENDED)

As this lab is completed in NETLAB+ and your code files will be erased when the reservation ends, it is advisable to save your files in GitHub under your repository for this course.