

# Exploring a simple Python shell

In this session, you will create a command shell in Python, and then run it and answer questions about it. You can use the [Jupyter Notebook space in Codio](#) for you work. **I used VSCODE**

Review the blogs at Praka (2018) and Szabo (n.d.) and then create a CLI/ shell that implements the following:

```
#Source: Szabo, G. (2018) Create your own interactive shell with cmd in
Python. Available from: https://code-maven.com/interactive-shell-with-cmd-in-
python
#subclass the Cmd class, so that's what we do here, though as being a skeleton
we don't do anything else with it
from cmd import Cmd
#create an instance object of the MyPrompt class and immediately call the
cmdloop method
class MyPrompt(Cmd):
    prompt = 'pb> '
    intro = "Welcome! Type ? to list commands"
#add commands to the system by implementing the corresponding do_* methods
    def do_exit(self, inp):
        print("Bye")
        return True

    def help_exit(self):
        print('exit the application. Shorthand: x q Ctrl-D.')

    def do_add(self, inp):
        print("adding '{}'.format(inp))

    def help_add(self):
        print("Add a new entry to the system.")

    def default(self, inp):
        if inp == 'x' or inp == 'q':
            return self.do_exit(inp)

        print("Default: {}".format(inp))

    do_EOF = do_exit
    help_EOF = help_exit

if __name__ == '__main__':
    MyPrompt().cmdloop()
```

- When you enter the command LIST it lists the contents of the current directory

```

1 #Inspired: Szabo, R. (2018) Create your own interactive shell with cmd in Python. Available from: https://code.wassio.com/info
2 #Includes the Cmd class, so that's what we do here, though as being a skeleton we don't do anything else with it
3 from cmd import Cmd
4 #Create an instance object of the MyPrompt class and immediately call the cmdloop method
5 class MyPrompt(Cmd):
6     prompt = 'pb> '
7     intro = "Welcome! Type ? to list commands"
8     #Add commands to the system by implementing the corresponding do_* methods
9     def do_exit(self, inp):
10         print("bye")
11         return True
12
13     def help_exit(self):
14         print('exit the application. Shorthand: q Ctrl-D.')
15
16     def do_add(self, inp):
17         print("adding {}".format(inp))
18
19     def help_add(self):
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Python Debug Console

```

PS C:\Users\Neslan Pirtmal-Jetha\Desktop> python interactive_shell.py
Welcome! Type ? to list commands
pb> ?

Documented commands (type help <topic>):
=====
EOF: add exit help

pb>

```

- The ADD command will add the following two numbers together and provide the result

```

1 #Inspired: Szabo, R. (2018) Create your own interactive shell with cmd in Python. Available from: https://code.wassio.com/info
2 #Includes the Cmd class, so that's what we do here, though as being a skeleton we don't do anything else with it
3 from cmd import Cmd
4 #Create an instance object of the MyPrompt class and immediately call the cmdloop method
5 class MyPrompt(Cmd):
6     prompt = 'pb> '
7     intro = "Welcome! Type ? to list commands"
8     #Add commands to the system by implementing the corresponding do_* methods
9     def do_exit(self, inp):
10         print("bye")
11         return True
12
13     def help_exit(self):
14         print('exit the application. Shorthand: q Ctrl-D.')
15
16     def do_add(self, inp):
17         print("adding {}".format(inp))
18
19     def help_add(self):
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Python Debug Console

```

pb> add
adding 2+2

```

- 
- The screenshot displays the Visual Studio Code editor with a Python file named `interactive_shell.py` open. The code implements a simple shell with the following methods:
- `do_exit(self, inp):` Prints "bye" and returns `True`.
  - `help_exit(self):` Prints "exit the application. Shorthand: + a Ctr-C-IL."
  - `do_add(self, inp):` Prompts the user to enter a number, reads input, and prints it.
  - `help_add(self):` Prints "Add a new entry to the system."
  - `default(self, inp):` Prints "Welcome! Type ? to list commands".
- The interface includes a sidebar on the left with panels for **VARIABLES**, **WATCH**, **CALL STACK**, and **BREAKPOINTS**. The **CALL STACK** panel shows the current execution state. The bottom status bar indicates the file is `interactive_shell.py` and the Python version is `Python 3.9.5 64-bit`.

- 
- The screenshot shows a Visual Studio Code editor window with the following details:
- Top Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
  - Left Sidebar:** Explorer, Search, Run and Debug, Extensions, Source Control, Run and Test Explorer, Output.
  - Main Editor:**
    - File: `interactive_shell.py`
    - Code content:
 

```
1 #MyPrompt: Simon, S. (2018) Create your own interactive shell with cmd in Python. Available from: https://code-examples.com/into
2 Encapsulate the Cmd class, so that's what we do here, though as being a skeleton we don't do anything else with it.
3 from cmd import Cmd
4 Create an instance object of the MyPrompt class and immediately call the do_exit method
5 class MyPrompt(Cmd):
6     prompt = 'pb>'
7     intro = 'Welcome! Type ? to list commands'
8     Add commands to the system by implementing the corresponding do_* methods
9
10     def do_exit(self, inp):
11         print("bye")
12         return True
13
14     def help_exit(self):
15         print('exit the application. Shorthand: x,q Ctrl-D.')
16
17     def do_add(self, inp):
18         print("adding {}".format(inp))
19
20     def help_add(self):
```
  - Bottom Panel:**
    - TERMINAL:**

```
Welcome! Type ? to list commands
pb> ?

Documented commands (type help <topic>):
=====
EOF add exit help

pb> help

Documented commands (type help <topic>):
=====
EOF add exit help

pb> help add
Add a new entry to the system.
pb> help exit
exit the application. Shorthand: x,q Ctrl-D.
pb> help help
list available commands with "help" or detailed help with "help cmd".
pb>
```
    - CALL STACK:** Shows the current execution stack.
    - BREAKPOINTS:** Shows any breakpoints set in the code.
  - Status Bar:** Python 3.9.5 64-bit, Python Current File (interactive\_shell.py), Run and Debug, Python 3.9.5 64-bit.
  - System Tray:** Shows the date and time: 05/01/2022.

- The EXIT command exits the shell

```

File Edit Selection View Go Run Terminal Help
Python: interactive_shell.py - interactive_shell.py - Visual Studio Code
Python: interactive_shell.py
21: return True
22:
23: def help_exit(self):
24:     print('exit the application. Shorthand: x q Ctrl-D.')
25:
26: def do_add(self, inp):
27:     print('adding {}'.format(inp))
28:
29: def help_add(self):
30:     print('Add a new entry to the system.')
31:
32: def default(self, inp):
33:     if inp == 'x' or inp == 'q':
34:         return self.do_exit(inp)
35:
36:     print('Default: {}'.format(inp))
37:
38: do_EOF = do_exit
39: help_EOF = help_exit
40:
41: if __name__ == '__main__':
42:     shell = InteractiveShell()
43:     shell.do_EOF('')
44:
45: Documented commands (type help <topic>):
46: EOF add exit help
47:
48: pto help add
49: Add a new entry to the system.
50: pto help exit
51: exit the application. Shorthand: x q Ctrl-D.
52: pto help help
53: list available commands with "help" or detailed help with "help cmd".
54: pto add
55: adding ''
56: pto add 'Hello'
57: adding 'Hello'
58: pto add 123
59: adding '123'
60: pto exit
61: Bye
62: PS C:\Users\NeeLan Pirmal-Jethava\Desktop> interactive_shell.py
  
```

Add suitable comments to your code and add the program to your e-portfolio. **Be prepared to demonstrate it in the seminar session next week.**

Run the shell you have created, try a few commands and then answer the questions below. Be prepared to discuss your answers in the seminar.

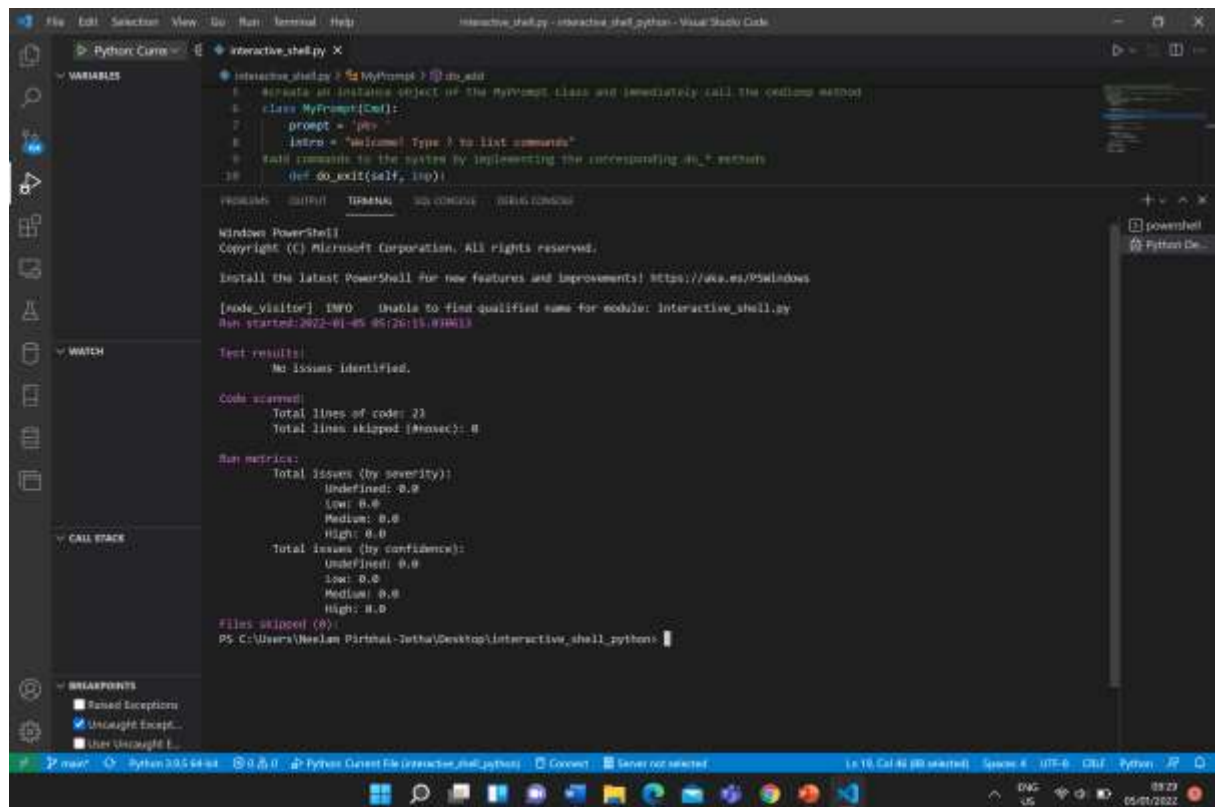
- What are the two main security vulnerabilities with your shell?

To test the vulnerabilities, type:

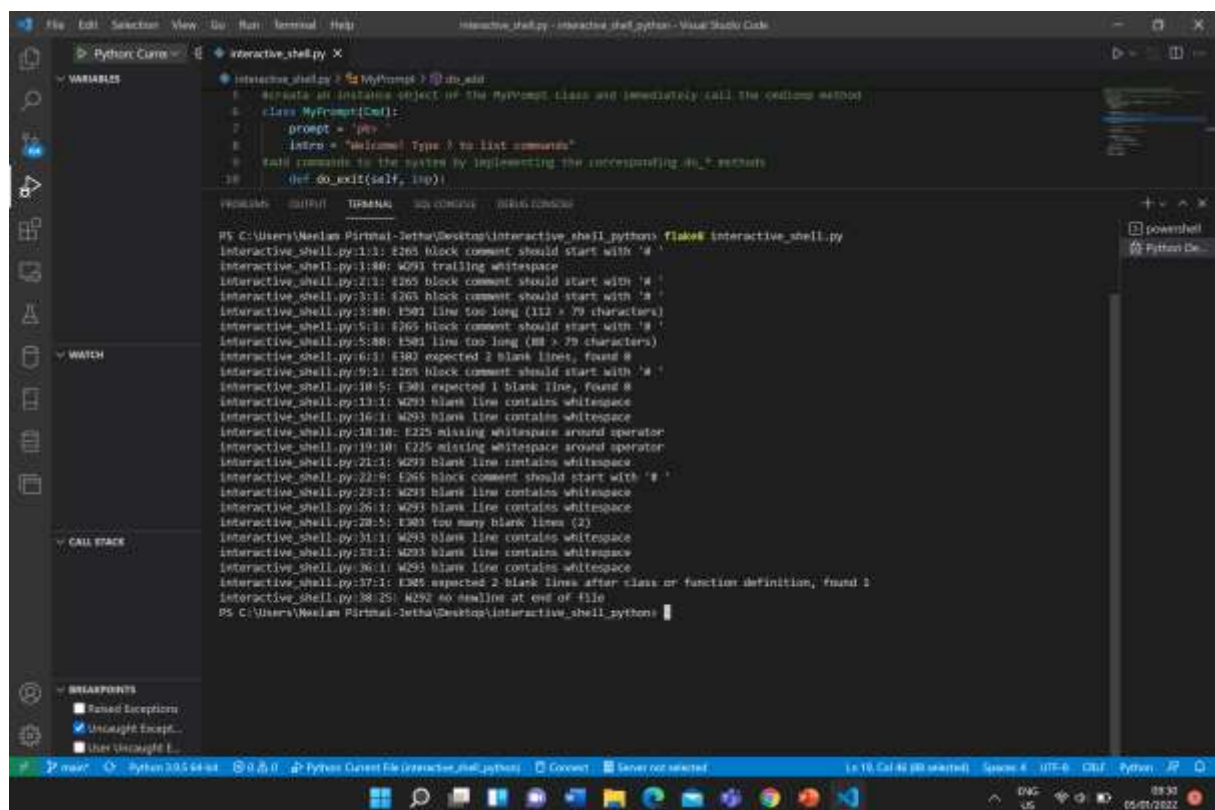
**bandit file\_name.py**

Here, it is

**bandit interactive\_shell.py**



Use flake8 to check



- What is one recommendation you would make to increase the security of the shell?

Review the spaces etc.

But the code taken online seems ok.

- Add a section to your e-portfolio that provides a (pseudo)code example of changes you would make to the shell to improve its security.