

[illegible]

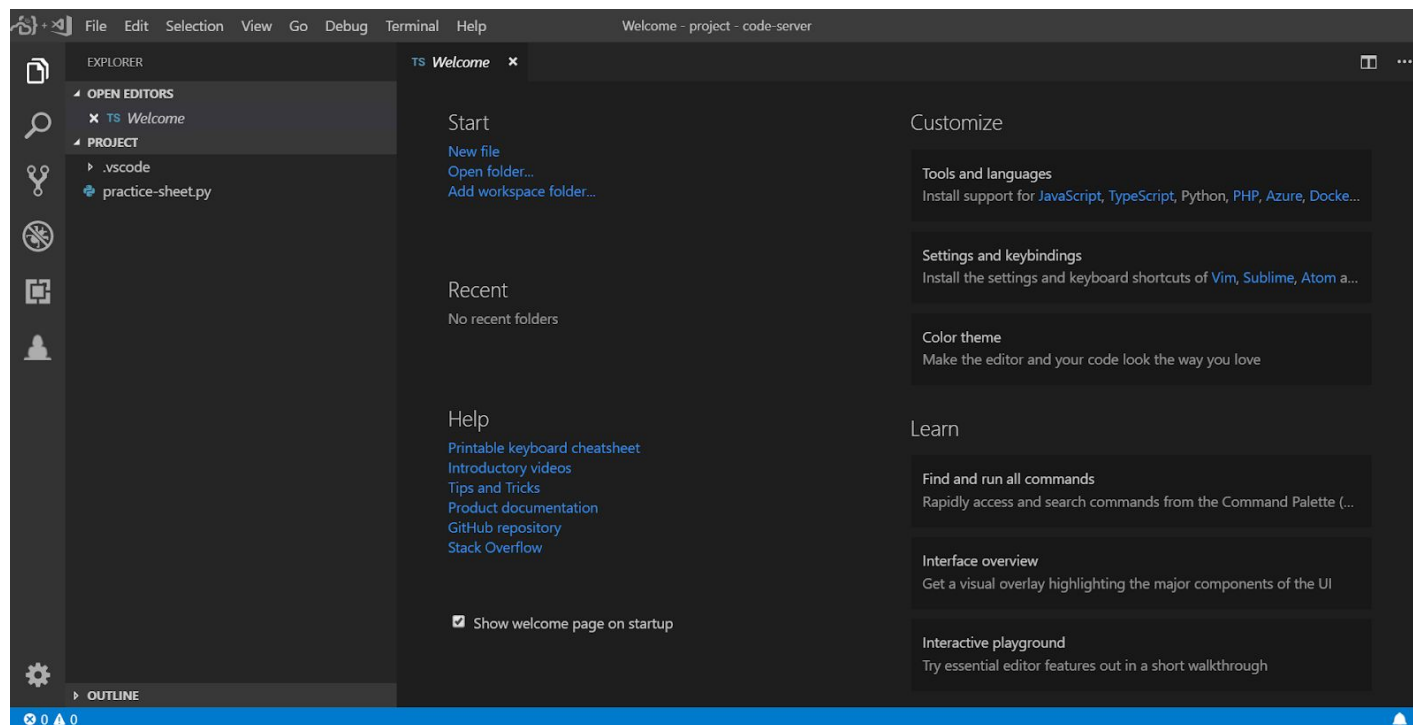
Name	Automating GDB with Python I
URL	https://attackdefense.com/challengedetails?cid=1210
Type	Offensive Python : Debugging

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

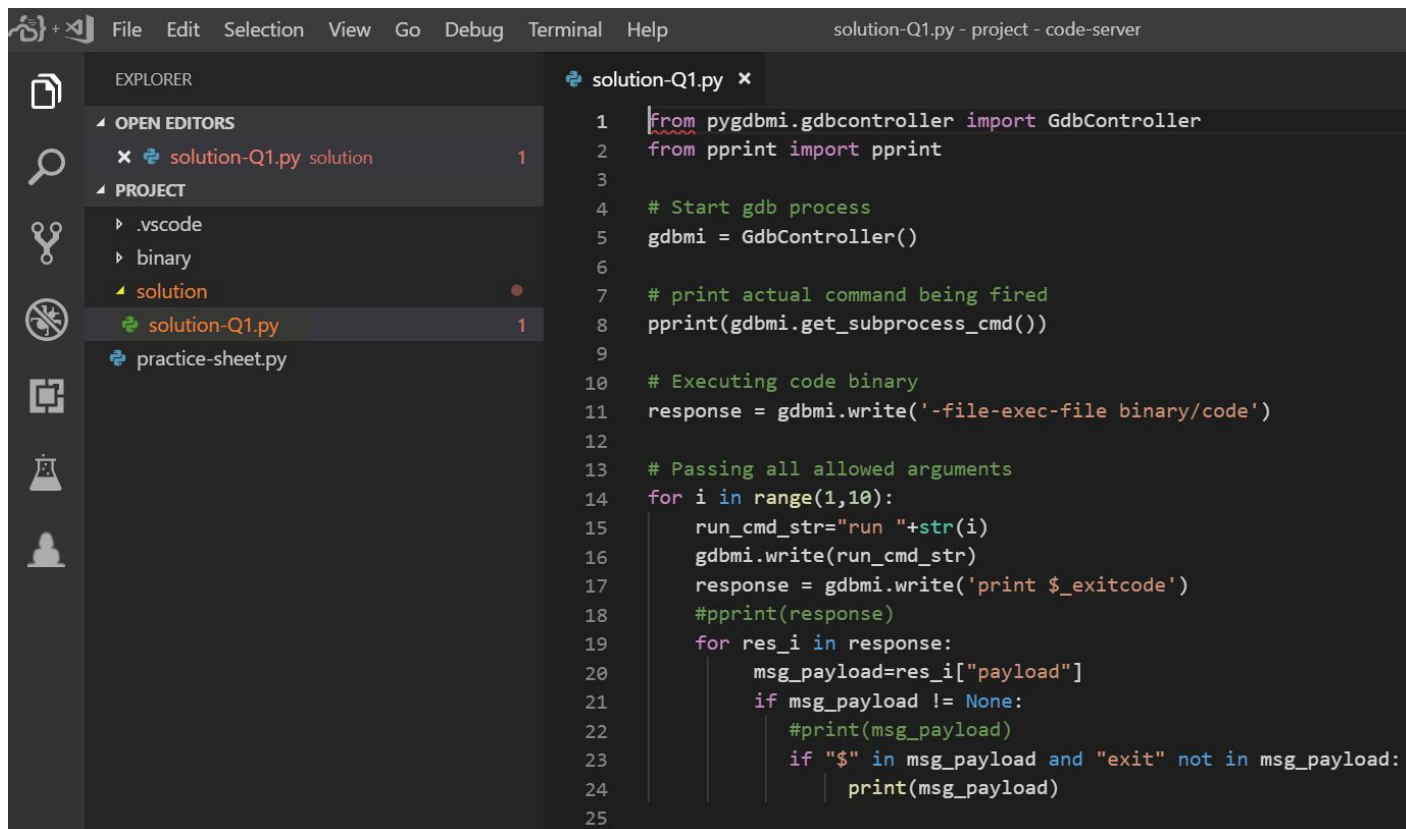
Objective: Use Python pygdbmi library to find all possible values of the exit status codes for all permitted inputs.

Solution:

Landing Page:



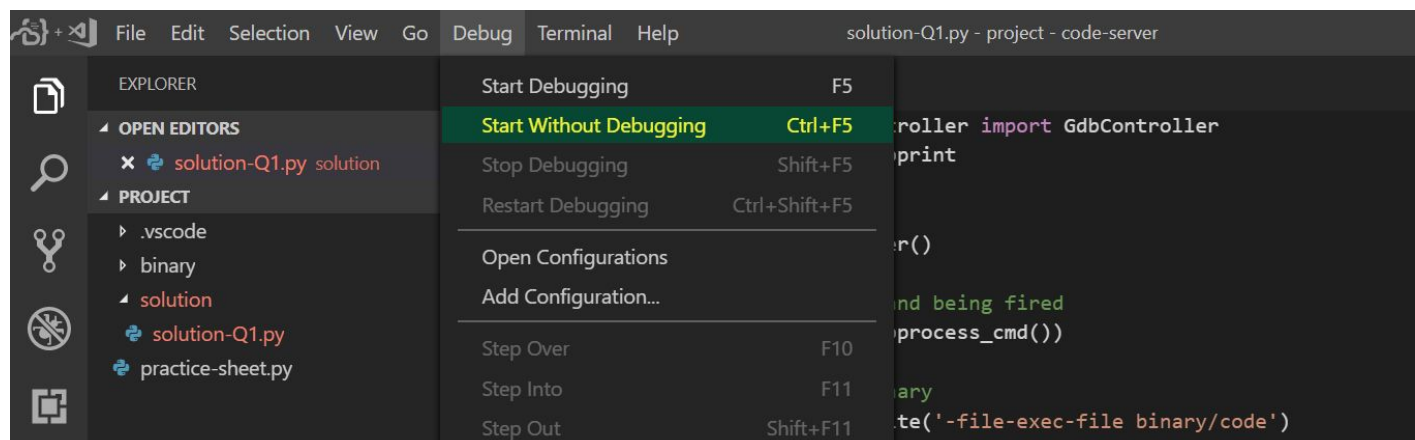
Step 1: Select “solution-Q1.py” from the solution directory listed in Project Explorer.



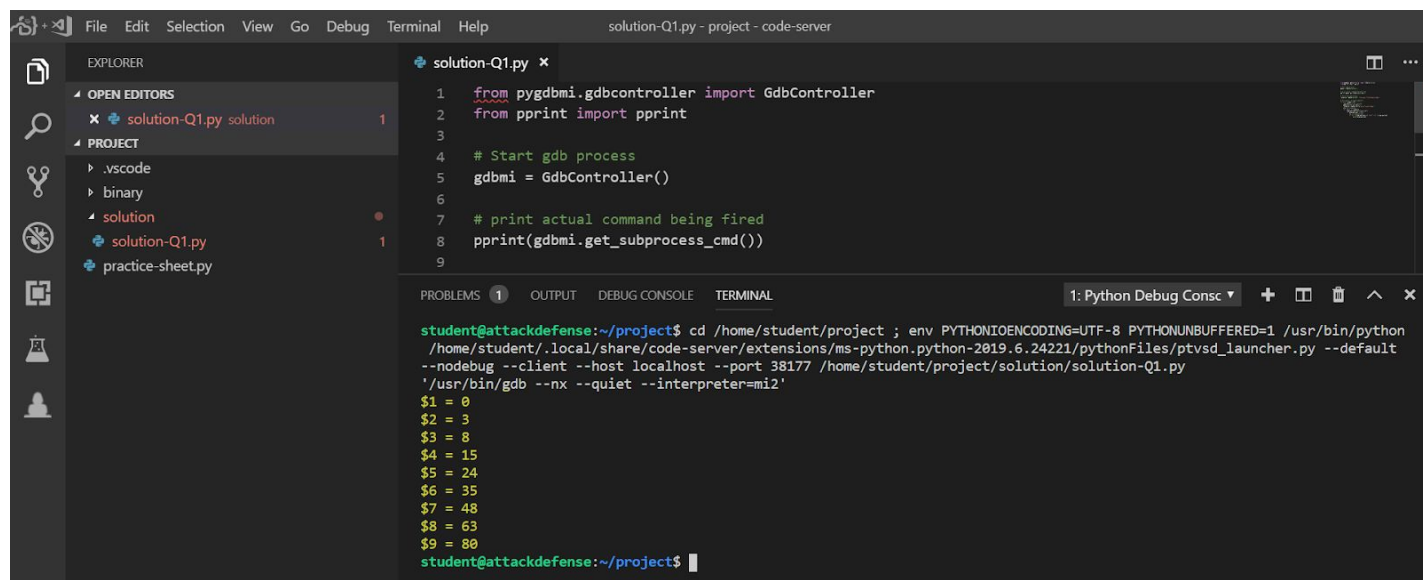
The screenshot shows the Visual Studio Code interface. In the Project Explorer on the left, the 'solution' directory is expanded, and 'solution-Q1.py' is selected. The main editor window displays the code for 'solution-Q1.py'. The code is a Python script that uses the 'pygdbmi' library to interact with GDB. It starts a GDB process, prints the command being fired, and then executes a binary file named 'code' in the 'binary' directory. It also prints the exit code and any messages received from the GDB process.

```
1 from pygdbmi.gdbcontroller import GdbController
2 from pprint import pprint
3
4 # Start gdb process
5 gdbmi = GdbController()
6
7 # print actual command being fired
8 pprint(gdbmi.get_subprocess_cmd())
9
10 # Executing code binary
11 response = gdbmi.write('-file-exec-file binary/code')
12
13 # Passing all allowed arguments
14 for i in range(1,10):
15     run_cmd_str="run "+str(i)
16     gdbmi.write(run_cmd_str)
17     response = gdbmi.write('print $_exitcode')
18     #pprint(response)
19     for res_i in response:
20         msg_payload=res_i["payload"]
21         if msg_payload != None:
22             #print(msg_payload)
23             if "$" in msg_payload and "exit" not in msg_payload:
24                 print(msg_payload)
25
```

Step 2: Navigate to Debug Menu and click on “Start Without Debugging option”.



The python script will be executed and the output will be displayed on the integrated terminal. In this case, the output is input and corresponding exist status code.



The screenshot shows the Visual Studio Code interface with a Python script named `solution-Q1.py` open in the editor. The script imports `GdbController` from `pygdbmi` and `pprint` from `pprint`. It then starts a GDB process and prints the actual command being fired. The terminal output shows the command being executed and the resulting output, which is a list of numbers from 0 to 80.

```
1 from pygdbmi.gdbcontroller import GdbController
2 from pprint import pprint
3
4 # Start gdb process
5 gdbmi = GdbController()
6
7 # print actual command being fired
8 pprint(gdbmi.get_subprocess_cmd())
9
```

```
student@attackdefense:~/project$ cd /home/student/project ; env PYTHONIOENCODING=UTF-8 PYTHONUNBUFFERED=1 /usr/bin/python
/home/student/.local/share/code-server/extensions/ms-python.python-2019.6.24221/pythonFiles/ptvsd_launcher.py --default
--nodebug --client --host localhost --port 38177 /home/student/project/solution/solution-Q1.py
'/usr/bin/gdb --nx --quiet --interpreter=mi2'
$1 = 0
$2 = 3
$3 = 8
$4 = 15
$5 = 24
$6 = 35
$7 = 48
$8 = 63
$9 = 80
student@attackdefense:~/project$
```

References:

1. Visual Studio Code (<https://code.visualstudio.com/>)
2. VS Code Basic Editing (<https://code.visualstudio.com/docs/editor/codebasics>)