Chrome File Edit View History Bookmarks Profiles Tab Window Help

Mon Jun 2 11:21 AM

LIConnections - VA and DoD | Log in to QA's learning platfo | Python: Introduction to Debu | Instructions - platform

about:blank

1h 44m

4 Introduction to PDB

Introduction

Debugging with `print` statements and logs have their use cases. However, sometimes you need something more interactive. The Python debugger enables users to inspect objects and run arbitrary Python code.

The Python debugger is a source code debugging module in the standard library called `pdb`. The `pdb` module provides a console which serves as a gateway between source code and the interpreter.

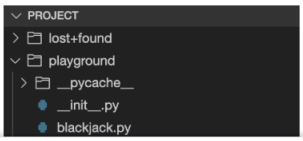
The `pdb` module provides multiple ways to run Python code under the control of the debugger. The debugger includes commands used to navigate through the code and pause on specific lines. While paused on specific lines, developers can inspect the state of the application

The Python debugger allows developers to set specific line numbers at which to pause. These are known as **breakpoints**. Setting breakpoints allows code to flow normally and pause when the interpreter encounters a breakpoint.

Being able to inspect the state of an application provides better insights into what's happening inside the Python runtime.

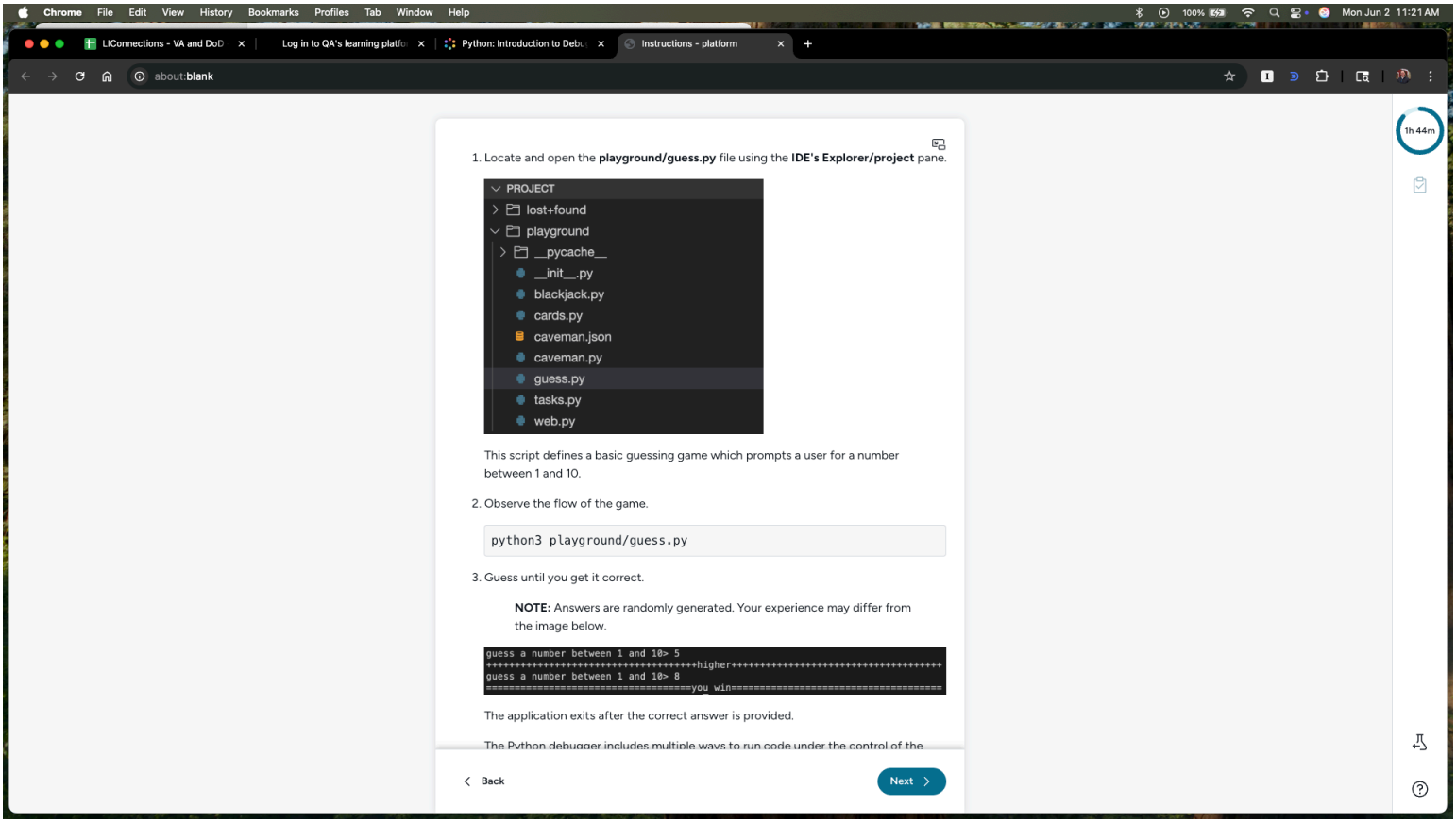
Instructions

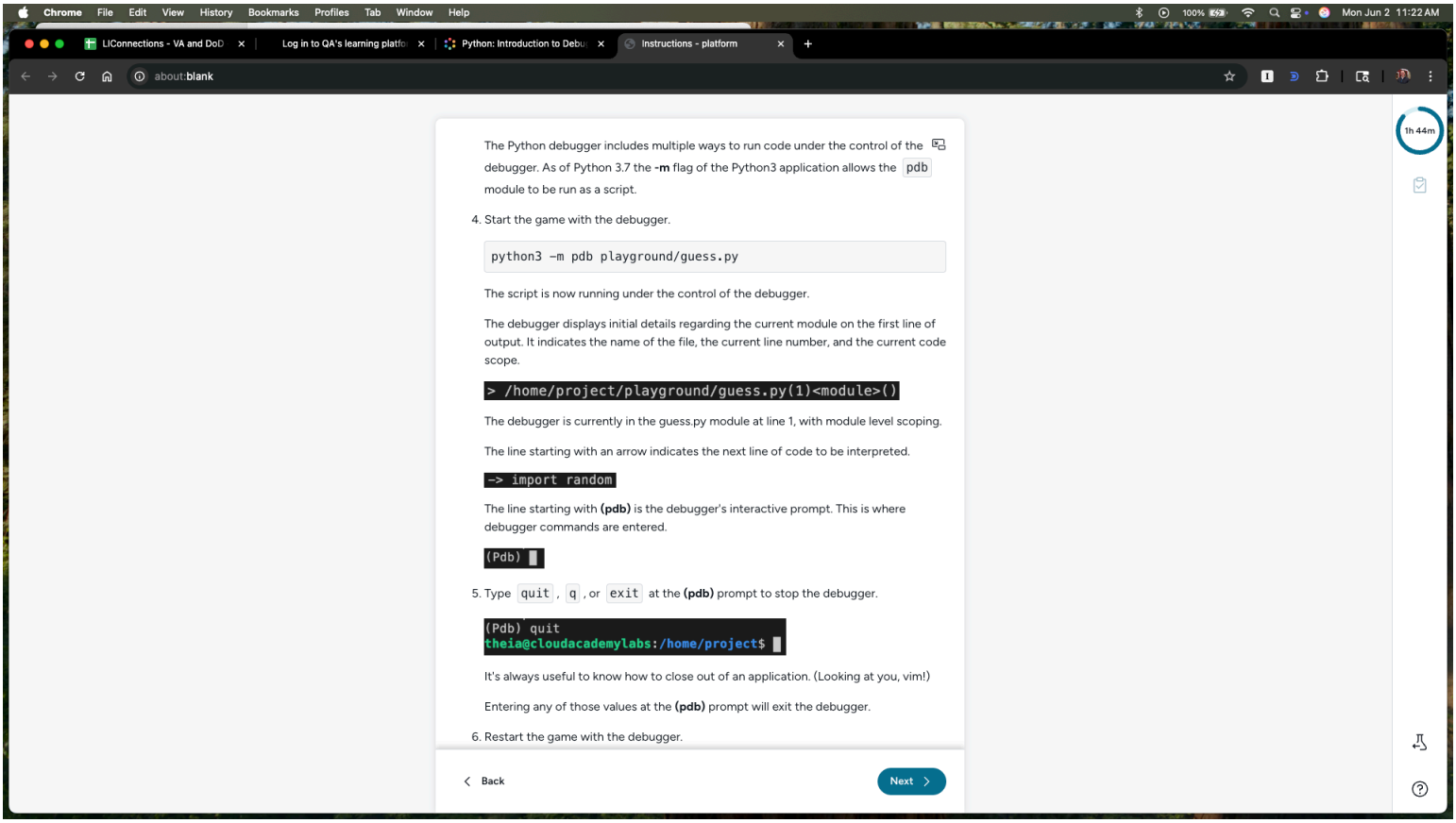
1. Locate and open the **playground/guess.py** file using the **IDE's Explorer/project** pane.

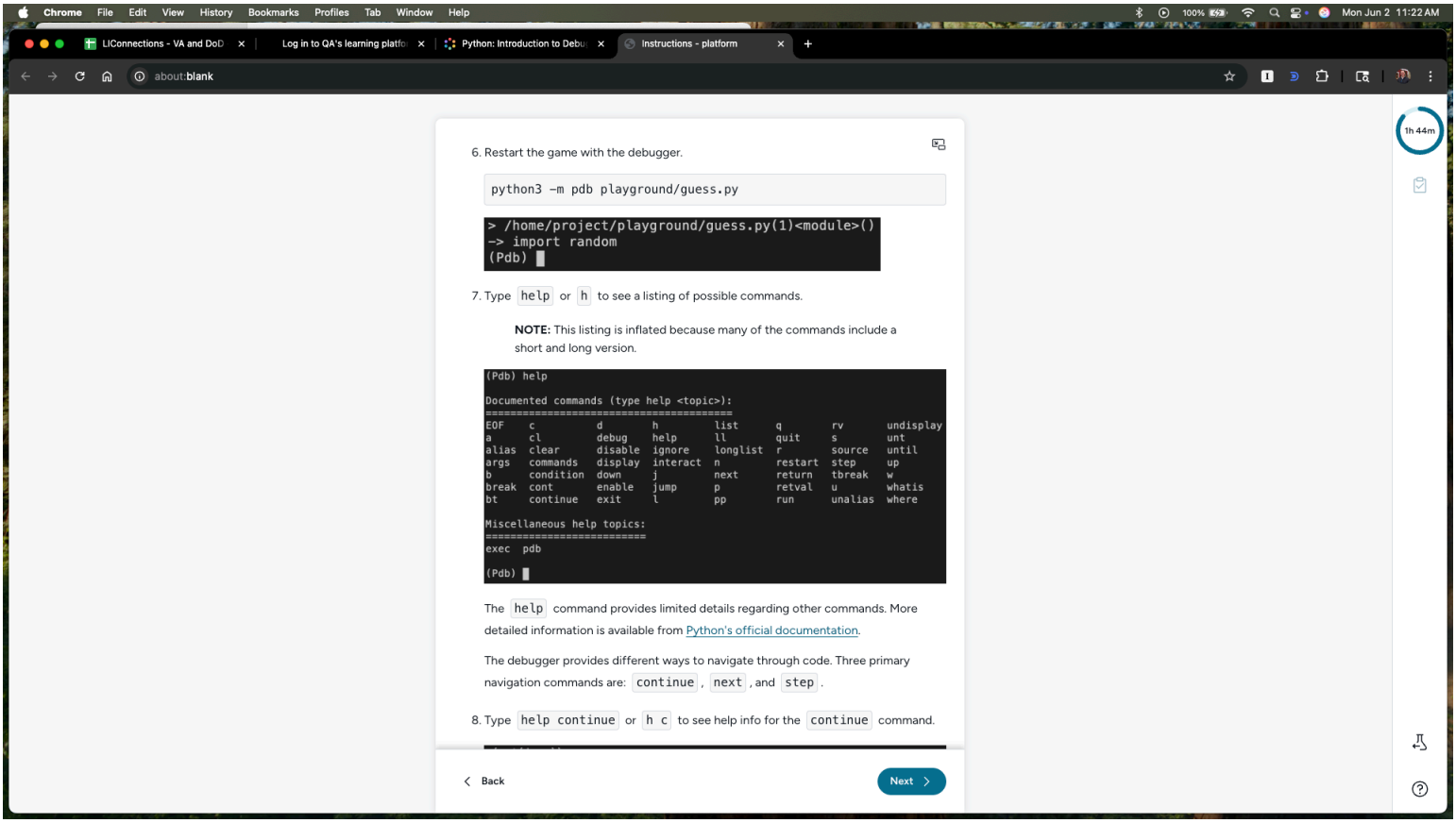


Back

Next


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jun 2 11:21 AM
LIConnections - VA and DoD
Log in to QA's learning platfo
Python: Introduction to Debu
Instructions - platform
about:blank
1h 44m
1. Locate and open the playground/guess.py file using the IDE's Explorer/project pane.
PROJECT
lost+found
playground
__pycache__
__init__.py
blackjack.py
cards.py
caveman.json
caveman.py
guess.py
tasks.py
web.py
This script defines a basic guessing game which prompts a user for a number between 1 and 10.
2. Observe the flow of the game.
python3 playground/guess.py
3. Guess until you get it correct.
NOTE: Answers are randomly generated. Your experience may differ from the image below.
guess a number between 1 and 10> 5
higher
guess a number between 1 and 10> 8
you win
The application exits after the correct answer is provided.
The Python debugger includes multiple ways to run code under the control of the
Back
Next


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jun 2 11:22 AM
LIConnections - VA and DoD
Log in to QA's learning platfo
Python: Introduction to Debu
Instructions - platform
about:blank
The Python debugger includes multiple ways to run code under the control of the debugger. As of Python 3.7 the -m flag of the Python3 application allows the pdb module to be run as a script.
4. Start the game with the debugger.
python3 -m pdb playground/guess.py
The script is now running under the control of the debugger.
The debugger displays initial details regarding the current module on the first line of output. It indicates the name of the file, the current line number, and the current code scope.
> /home/project/playground/guess.py(1)<module>()
The debugger is currently in the guess.py module at line 1, with module level scoping.
The line starting with an arrow indicates the next line of code to be interpreted.
-> import random
The line starting with (pdb) is the debugger's interactive prompt. This is where debugger commands are entered.
(Pdb)
5. Type quit , q , or exit at the (pdb) prompt to stop the debugger.
(Pdb) quit
theia@cloudacademylabs:/home/project$
It's always useful to know how to close out of an application. (Looking at you, vim!)
Entering any of those values at the (pdb) prompt will exit the debugger.
6. Restart the game with the debugger.
Back
Next
1h 44m


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jun 2 11:22 AM
LIConnections - VA and DoD
Log in to QA's learning platfo
Python: Introduction to Debu
Instructions - platform
about:blank
6. Restart the game with the debugger.
python3 -m pdb playground/guess.py
> /home/project/playground/guess.py(1)<module>()
-> import random
(Pdb)
7. Type help or h to see a listing of possible commands.
NOTE: This listing is inflated because many of the commands include a short and long version.
(Pdb) help
Documented commands (type help <topic>):
EOF c d h list q rv undisplay
a cl debug help ll quit s unt
alias clear disable ignore longlist r source until
args commands display interact n restart step up
b condition down j next return tbreak w
break cont enable jump p retval u whatis
bt continue exit l pp run unalias where
Miscellaneous help topics:
exec pdb
(Pdb)
The help command provides limited details regarding other commands. More detailed information is available from Python's official documentation.
The debugger provides different ways to navigate through code. Three primary navigation commands are: continue , next , and step .
8. Type help continue or h c to see help info for the continue command.
Back
Next
1h 44m

8. Type `help continue` or `h c` to see help info for the `continue` command.

```
c(ont(inue))
        Continue execution, only stop when a breakpoint is encountered.
(Pdb)
```

The `continue` (`c` , `cont`) command instructs the debugger to resume the normal code flow until specific events occur.

The debugger yields control of the terminal to the application until either a breakpoint is reached, an exception is encountered, or the application exits. The **(pdb)** prompt disappears when the application has control of the terminal.

9. Type `continue` or `c` to resume the normal code flow.

```
(Pdb) continue
guess a number between 1 and 10>
```

The game's prompt is displayed asking for a guess.

10. Guess until you get it correct.

```
guess a number between 1 and 10> 5
----------------------------------lower-----------------------------------
guess a number between 1 and 10> 3
----------------------------------lower-----------------------------------
guess a number between 1 and 10> 1
==================================you win==================================
The program finished and will be restarted
> /home/project/playground/guess.py(1)<module>()
-> import random
(Pdb)
```

Guessing correctly results in a message being displayed before the application naturally exits. The debugger automatically restarts applications after they exit.

By automatically restarting the application the debugger preserves the current state, including breakpoints. This provides another opportunity to navigate through the code.

Breakpoints are useful when you know where in the code you want the debugger to stop and present a debugger prompt. The debugger includes multiple commands for setting breakpoints. The `break` (`b`) command is able to set breakpoints and display details about existing breakpoints.

11. Type `break 22` or `b 22` to set a breakpoint on line 22 of the current module.

```
(Pdb) break 22
Breakpoint 1 at /home/project/playground/guess.py:22
(Pdb)
```

The output confirms that the breakpoint was set successfully. This breakpoint ensures that the code will always stop when it encounters line 22.

12. Type `continue` or `c` to resume the normal code flow.

```
(Pdb) continue
guess a number between 1 and 10>
```

The code will continue to run normally until line 22 reached. Entering an intentionally high guess will advance to line 22.

13. Enter *100* in the game prompt.

```
guess a number between 1 and 10> 100
> /home/project/playground/guess.py(22)play()
-> inform('+')
(Pdb)
```

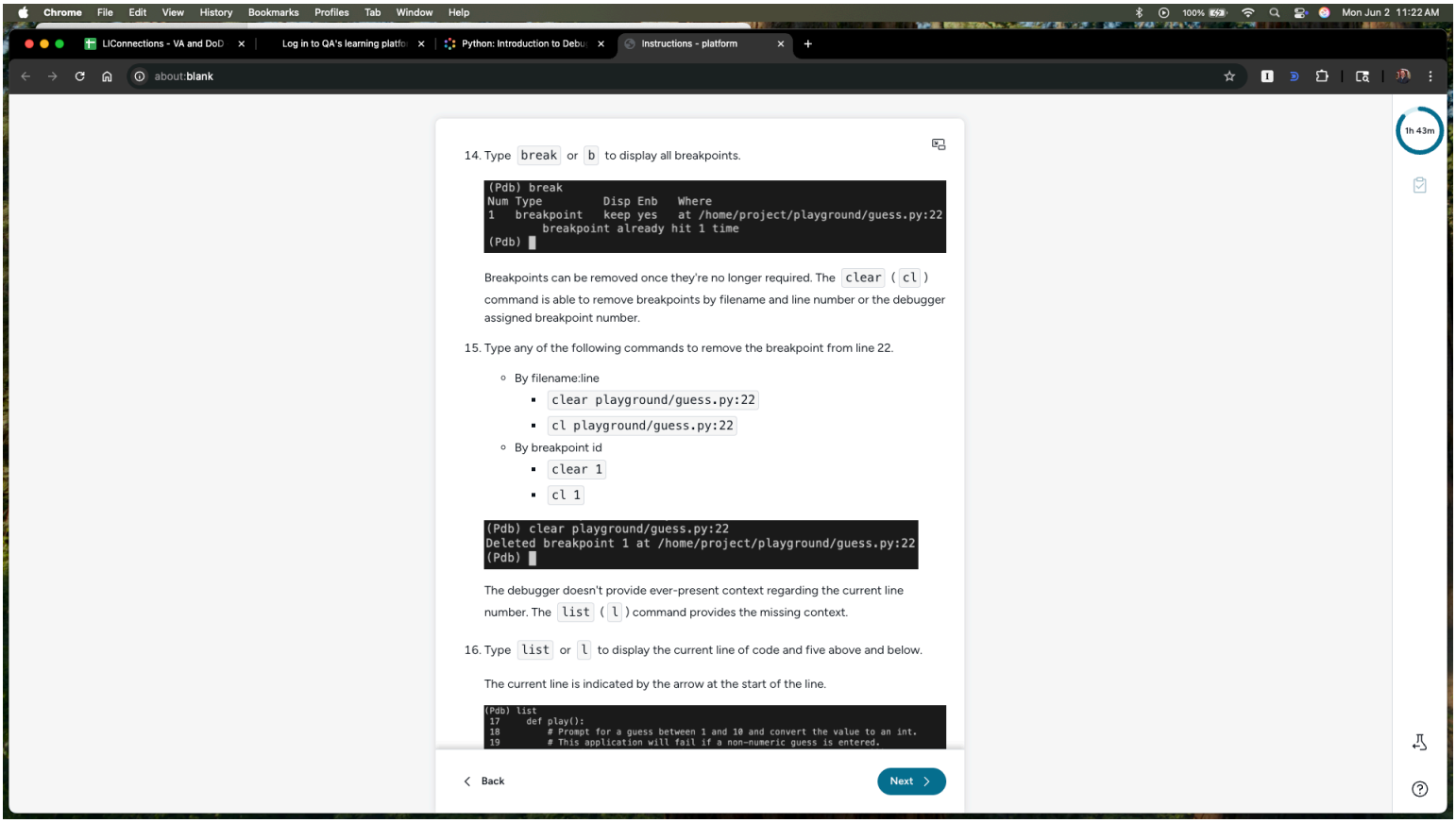
The debugger pauses at line 22 and presents a prompt. The `break` command displays breakpoint details when called without arguments.

14. Type `break` or `b` to display all breakpoints.

```
(Pdb) break
```

Back

Next


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jun 2 11:22 AM
LIConnections - VA and DoD
Log in to QA's learning platfo
Python: Introduction to Debu
Instructions - platform
about:blank
1h 43m
14. Type break or b to display all breakpoints.
(Pdb) break
Num Type Disp Enb Where
1 breakpoint keep yes at /home/project/playground/guess.py:22
breakpoint already hit 1 time
(Pdb)
Breakpoints can be removed once they're no longer required. The clear (cl) command is able to remove breakpoints by filename and line number or the debugger assigned breakpoint number.
15. Type any of the following commands to remove the breakpoint from line 22.
By filename:line
clear playground/guess.py:22
cl playground/guess.py:22
By breakpoint id
clear 1
cl 1
(Pdb) clear playground/guess.py:22
Deleted breakpoint 1 at /home/project/playground/guess.py:22
(Pdb)
The debugger doesn't provide ever-present context regarding the current line number. The list (l) command provides the missing context.
16. Type list or l to display the current line of code and five above and below.
The current line is indicated by the arrow at the start of the line.
(Pdb) list
17 def play():
18 # Prompt for a guess between 1 and 10 and convert the value to an int.
19 # This application will fail if a non-numeric guess is entered.
Back
Next

16. Type `list` or `l` to display the current line of code and five above and below.

    The current line is indicated by the arrow at the start of the line.

```
(Pdb) list
 17     def play():
 18         # Prompt for a guess between 1 and 10 and convert the value to an int.
 19         # This application will fail if a non-numeric guess is entered.
 20         while (guess := int(input('guess a number between 1 and 10> '))) != answer:
 21             if guess > answer:
 22  ->             inform('+')
 23             elif guess < answer:
 24                 inform('-')
 25         else:
 26             inform('=')
 27
(Pdb)
```

    The debugger considers anything entered into the prompt that isn't a command to be a Python expression. The results of expressions are displayed below the prompt, similar to Python's interactive console.

    At this point in the code the name `answer` is bound to a random `int` produced by the `random` module.

17. Type `answer` to display the random number.

    - Take note of the answer.

```
(Pdb) answer
2
(Pdb)
```

18. Type `continue` to resume the normal code flow.

```
(Pdb) continue
-----------------------------------lower-----------------------------------
guess a number between 1 and 10>
```

19. Enter the debugger-obtained answer to complete the game and exit the application.

Back
Next

19. Enter the debugger-obtained answer to complete the game and exit the applicatic

```
guess a number between 1 and 10> 2
==================================you win===================================
The program finished and will be restarted
> /home/project/playground/guess.py(1)<module>()
-> import random
(Pdb)
```

The debugger includes commands used to navigate through code in different ways. The `next` (`n`) and `step` (`s`) are two common commands for navigating code.

The `next` command is used to step through each line in a callable or frame. The debugger remains inside the currently debugged callable. When the debugger encounters calls to other callables they're run and the results are returned; then the debugger moves to the next line in the currently debugged callable.

The `step` command is used to step line-by-line through code, occasionally including code from built-in or third-party modules. When the debugger encounters calls to other callables it also debugs those callables.

Breakpoints are required to help illustrate the difference between these two commands.

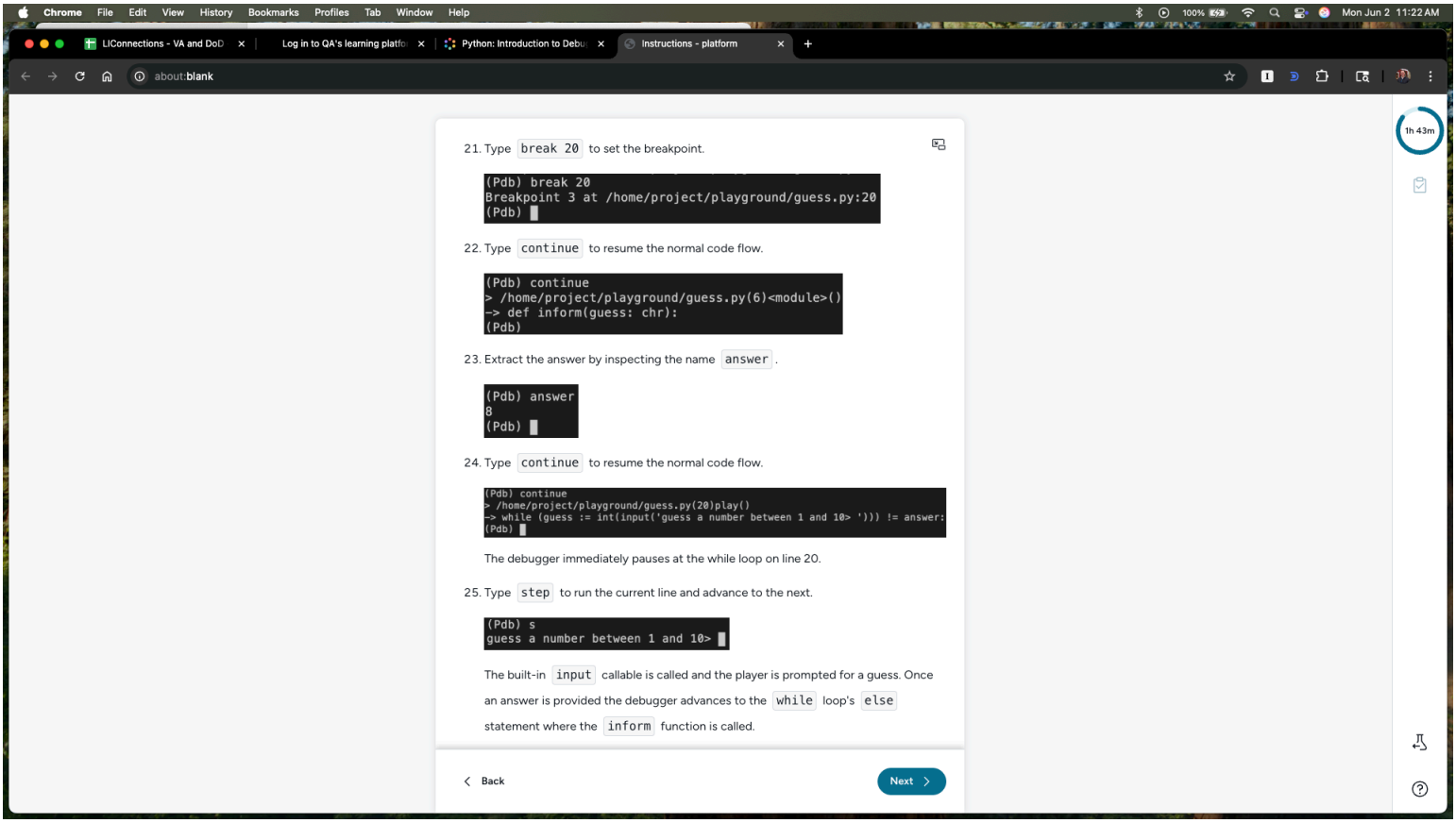
Set a breakpoint after the answer is generated in order to extract the answer.

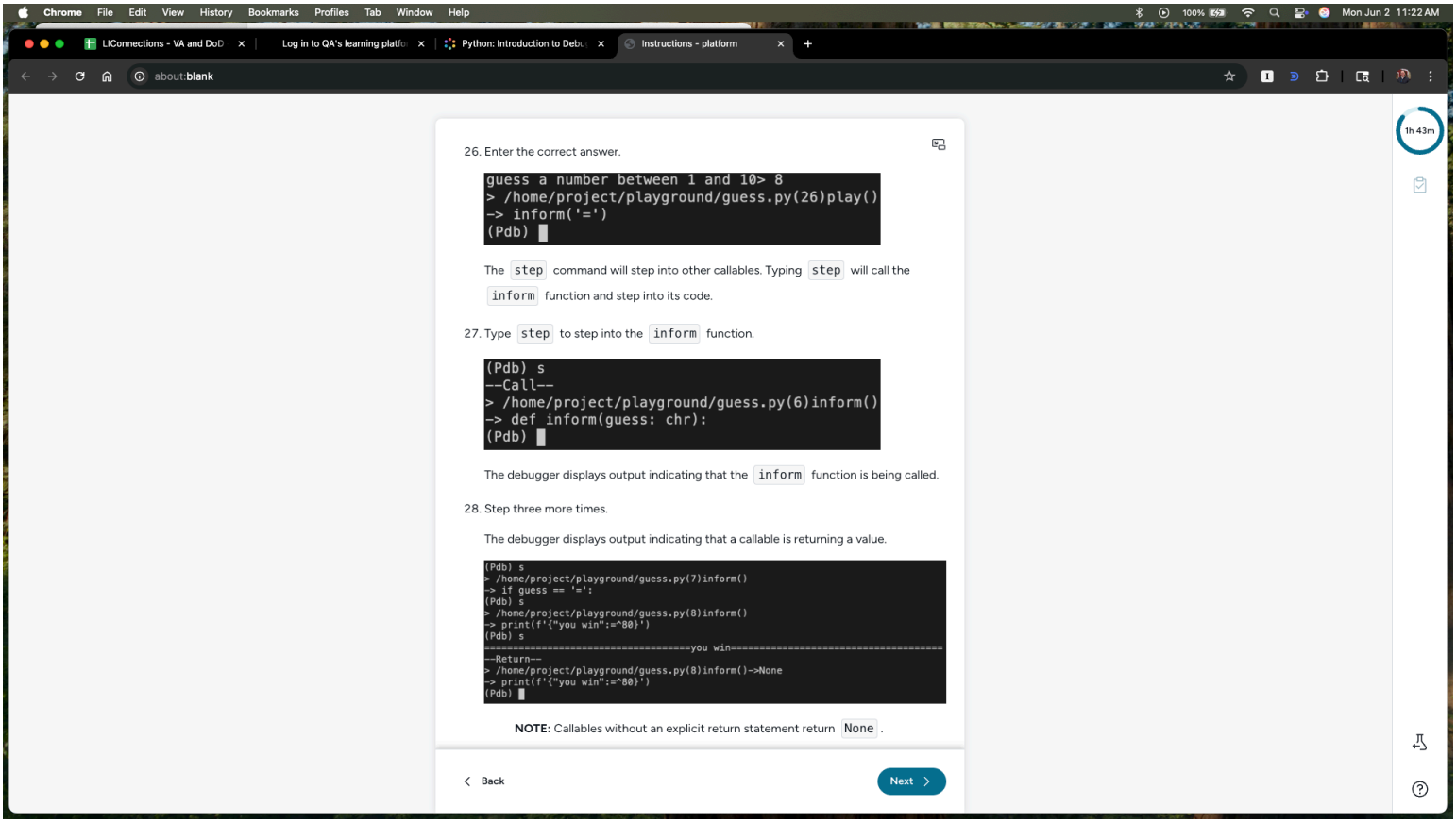
20. Type `break 6` to set the breakpoint.

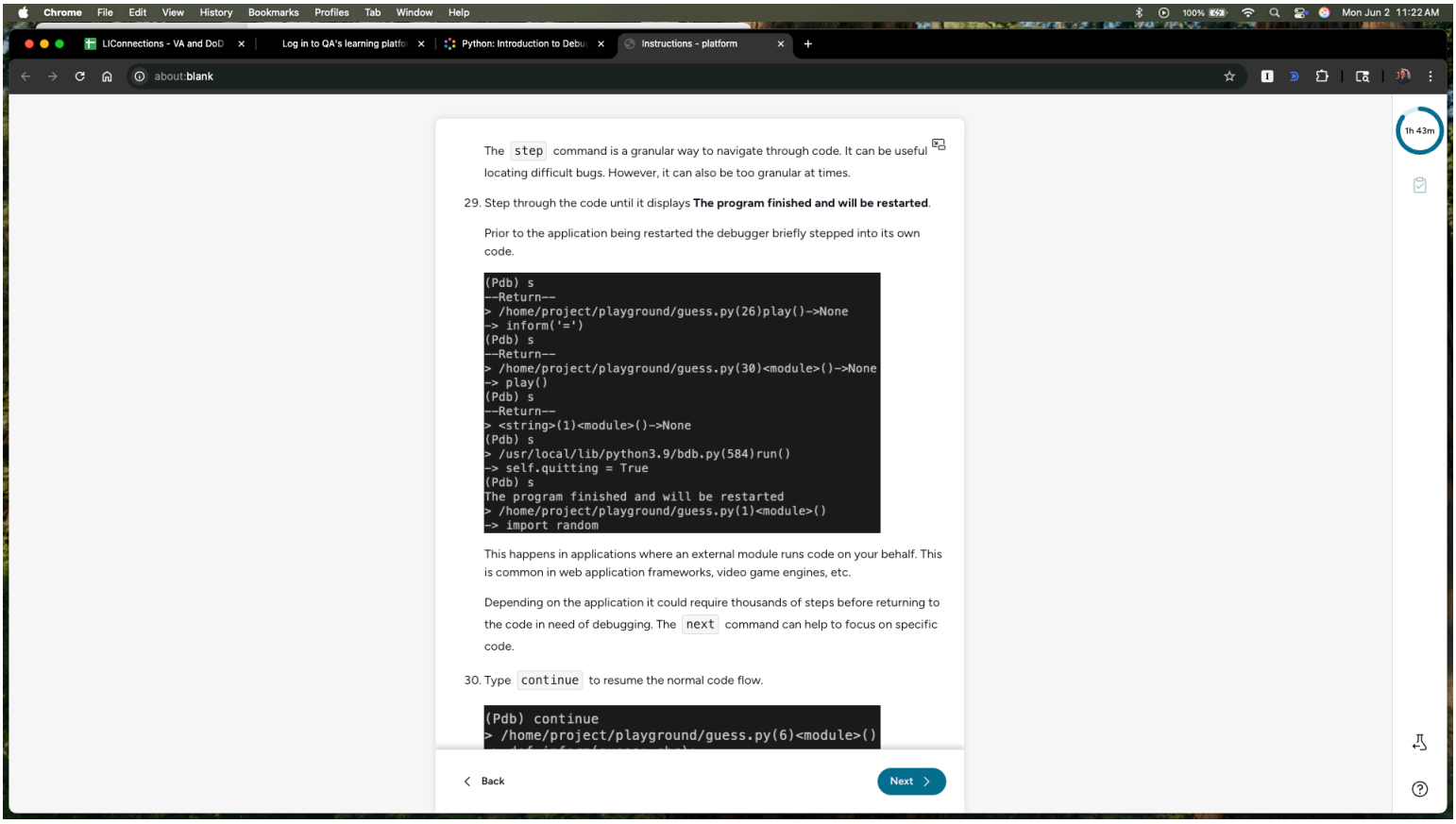
```
(Pdb) break 6
Breakpoint 2 at /home/project/playground/guess.py:6
(Pdb)
```

Setting a breakpoint for line 20 will display the debugger prompt when the `play` function is called.

21. Type `break 20` to set the breakpoint.


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jun 2 11:22 AM
LIConnections - VA and DoD
Log in to QA's learning platfo
Python: Introduction to Debu
Instructions - platform
about:blank
1h 43m
21. Type break 20 to set the breakpoint.
(Pdb) break 20
Breakpoint 3 at /home/project/playground/guess.py:20
(Pdb)
22. Type continue to resume the normal code flow.
(Pdb) continue
> /home/project/playground/guess.py(6)<module>()
-> def inform(guess: chr):
(Pdb)
23. Extract the answer by inspecting the name answer .
(Pdb) answer
8
(Pdb)
24. Type continue to resume the normal code flow.
(Pdb) continue
> /home/project/playground/guess.py(20)play()
-> while (guess := int(input('guess a number between 1 and 10> '))) != answer:
(Pdb)
The debugger immediately pauses at the while loop on line 20.
25. Type step to run the current line and advance to the next.
(Pdb) s
guess a number between 1 and 10>
The built-in input callable is called and the player is prompted for a guess. Once an answer is provided the debugger advances to the while loop's else statement where the inform function is called.
Back
Next


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jun 2 11:22 AM
LIConnections - VA and DoD
Log in to QA's learning platfo
Python: Introduction to Debu
Instructions - platform
about:blank
1h 43m
26. Enter the correct answer.
guess a number between 1 and 10> 8
> /home/project/playground/guess.py(26)play()
-> inform('=')
(Pdb)
The step command will step into other callables. Typing step will call the inform function and step into its code.
27. Type step to step into the inform function.
(Pdb) s
--Call--
> /home/project/playground/guess.py(6)inform()
-> def inform(guess: chr):
(Pdb)
The debugger displays output indicating that the inform function is being called.
28. Step three more times.
The debugger displays output indicating that a callable is returning a value.
(Pdb) s
> /home/project/playground/guess.py(7)inform()
-> if guess == '=':
(Pdb) s
> /home/project/playground/guess.py(8)inform()
-> print(f'{"you win":=^80}')
(Pdb) s
==========you win==========
--Return--
> /home/project/playground/guess.py(8)inform()->None
-> print(f'{"you win":=^80}')
(Pdb)
NOTE: Callables without an explicit return statement return None .
Back
Next


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jun 2 11:22 AM
LIConnections - VA and DoD
Log in to QA's learning platfo
Python: Introduction to Debu
Instructions - platform
about:blank
1h 43m
The step command is a granular way to navigate through code. It can be useful locating difficult bugs. However, it can also be too granular at times.
29. Step through the code until it displays **The program finished and will be restarted**.
Prior to the application being restarted the debugger briefly stepped into its own code.
(Pdb) s
--Return--
> /home/project/playground/guess.py(26)play()->None
-> inform('=')
(Pdb) s
--Return--
> /home/project/playground/guess.py(30)<module>()->None
-> play()
(Pdb) s
--Return--
> <string>(1)<module>()->None
(Pdb) s
> /usr/local/lib/python3.9/bdb.py(584)run()
-> self.quitting = True
(Pdb) s
The program finished and will be restarted
> /home/project/playground/guess.py(1)<module>()
-> import random
This happens in applications where an external module runs code on your behalf. This is common in web application frameworks, video game engines, etc.
Depending on the application it could require thousands of steps before returning to the code in need of debugging. The next command can help to focus on specific code.
30. Type continue to resume the normal code flow.
(Pdb) continue
> /home/project/playground/guess.py(6)<module>()
Back
Next

30. Type `continue` to resume the normal code flow.

```
(Pdb) continue
> /home/project/playground/guess.py(6)<module>()
-> def inform(guess: chr):
(Pdb)
```

31. Extract the answer by inspecting the name `answer` .

```
(Pdb) answer
5
(Pdb)
```

32. Type `continue` to resume the normal code flow.

```
(Pdb) continue
> /home/project/playground/guess.py(20)play()
-> while (guess := int(input('guess a number between 1 and 10> '))) != answer:
(Pdb)
```

33. Type `next` to advance to the next line of code in the `play` function.

```
(Pdb) next
guess a number between 1 and 10>
```

34. Enter the correct answer.

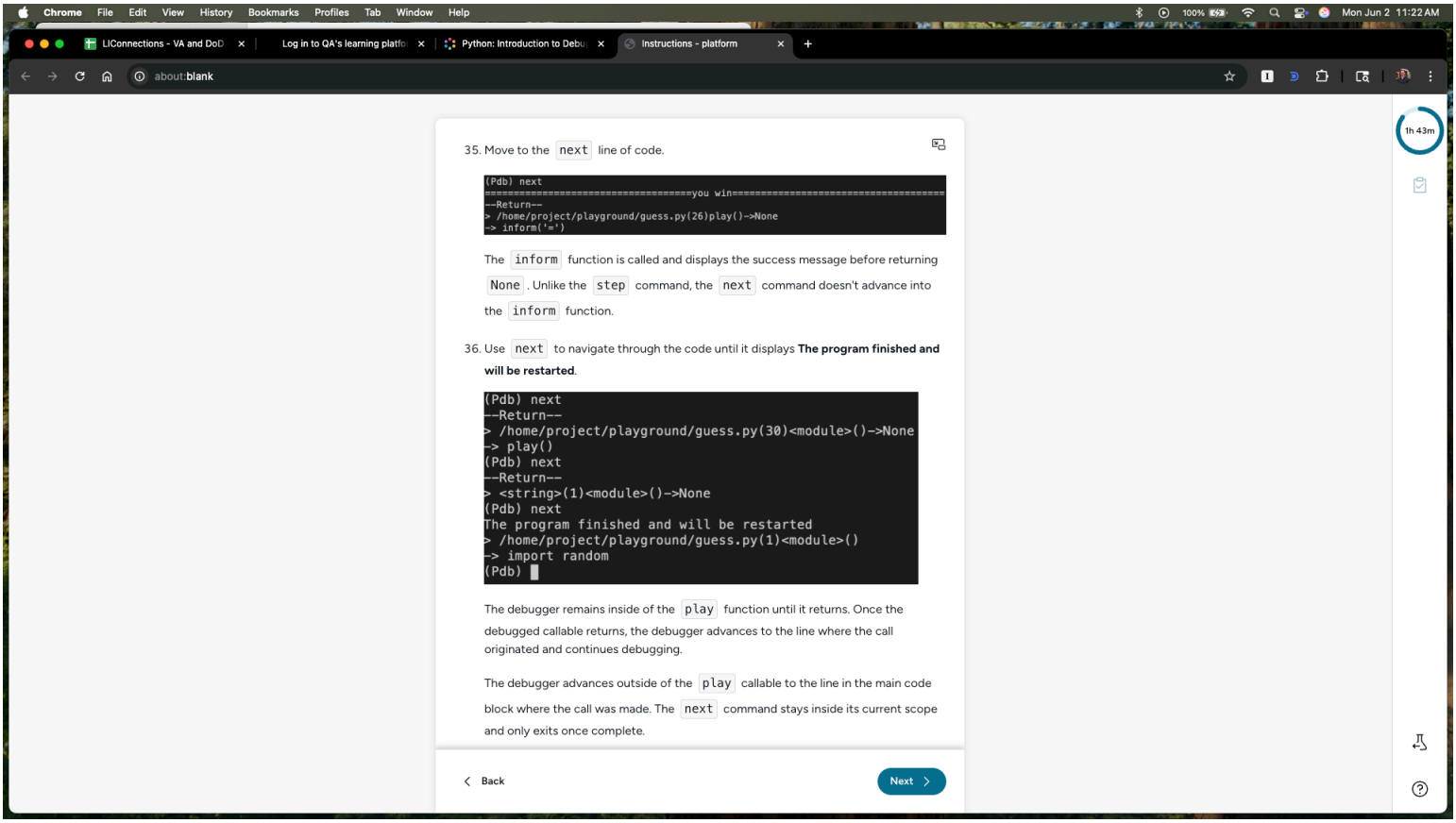
```
guess a number between 1 and 10> 5
> /home/project/playground/guess.py(26)play()
-> inform('=')
(Pdb)
```

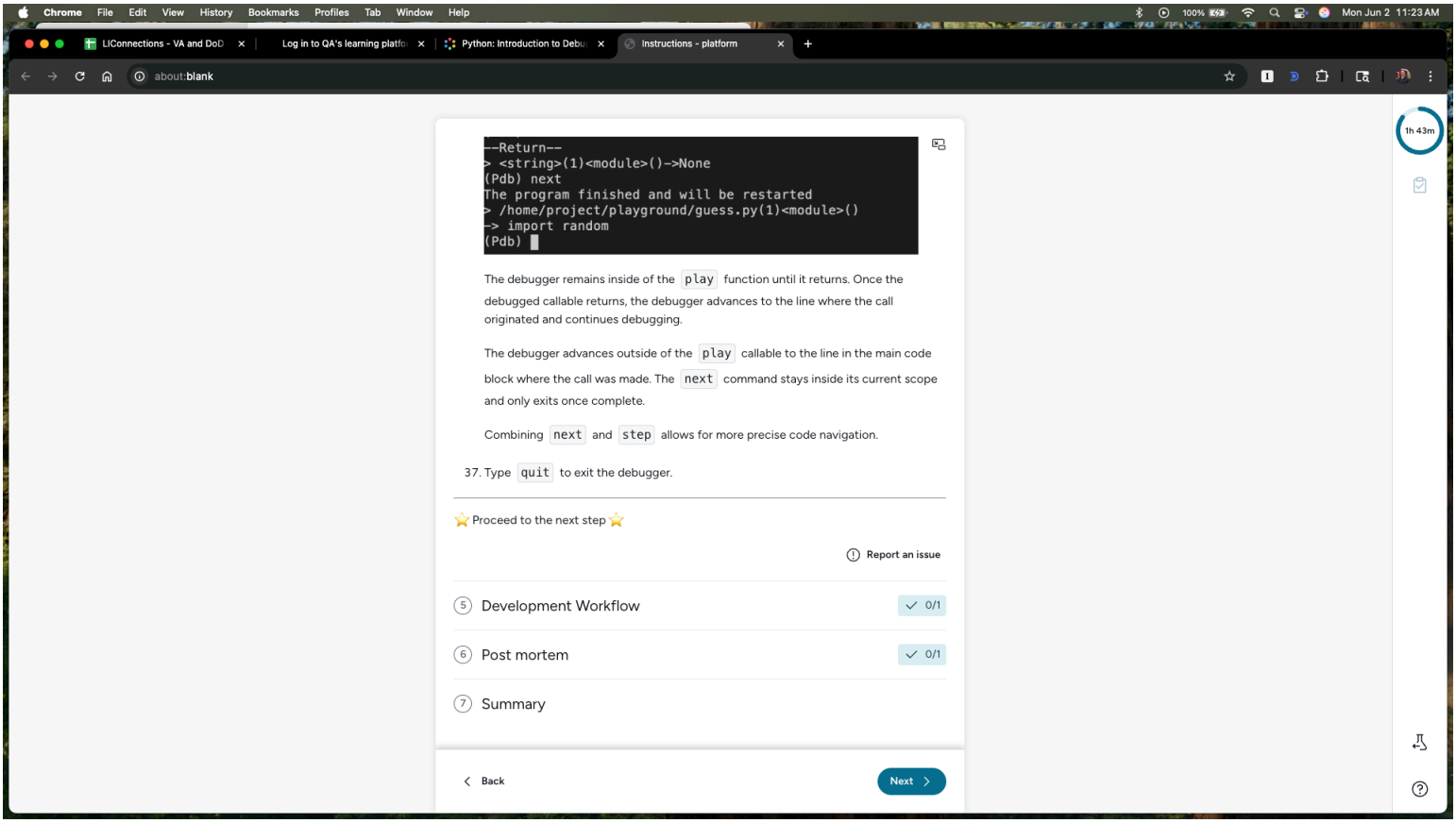
35. Move to the `next` line of code.

```
(Pdb) next
==================================you win==================================
--Return--
> /home/project/playground/guess.py(26)play()->None
```

Back

Next


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jun 2 11:22 AM
LIConnections - VA and DoD
Log in to QA's learning platfo
Python: Introduction to Debu
Instructions - platform
about:blank
1h 43m
35. Move to the next line of code.
(Pdb) next
==========you win==========
--Return--
> /home/project/playground/guess.py(26)play()->None
-> inform('=')
The inform function is called and displays the success message before returning None. Unlike the step command, the next command doesn't advance into the inform function.
36. Use next to navigate through the code until it displays The program finished and will be restarted.
(Pdb) next
--Return--
> /home/project/playground/guess.py(30)<module>()->None
-> play()
(Pdb) next
--Return--
> <string>(1)<module>()->None
(Pdb) next
The program finished and will be restarted
> /home/project/playground/guess.py(1)<module>()
-> import random
(Pdb)
The debugger remains inside of the play function until it returns. Once the debugged callable returns, the debugger advances to the line where the call originated and continues debugging.
The debugger advances outside of the play callable to the line in the main code block where the call was made. The next command stays inside its current scope and only exits once complete.
Back
Next


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jun 2 11:23 AM
LIConnections - VA and DoD
Log in to QA's learning platfo
Python: Introduction to Debu
Instructions - platform
about:blank
--Return--
> <string>(1)<module>()->None
(Pdb) next
The program finished and will be restarted
> /home/project/playground/guess.py(1)<module>()
-> import random
(Pdb)
The debugger remains inside of the play function until it returns. Once the debugged callable returns, the debugger advances to the line where the call originated and continues debugging.
The debugger advances outside of the play callable to the line in the main code block where the call was made. The next command stays inside its current scope and only exits once complete.
Combining next and step allows for more precise code navigation.
37. Type quit to exit the debugger.
Proceed to the next step
Report an issue
5 Development Workflow 0/1
6 Post mortem 0/1
7 Summary
Back
Next
1h 43m