6 Post mortem

Introduction

Exceptions are a part of the development process. The `pdb` module includes a feature called **post mortem debugging**. When debugged applications raise unhandled exceptions the debugger automatically enters post mortem debugging mode.

Post mortem debugging allows developers to inspect the state of an application at the time it crashed.

Instructions

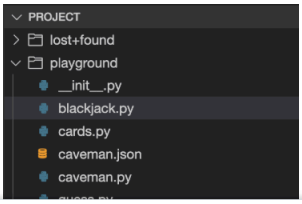
Imagine being assigned the following end-user reported bug ticket.

Subject: I lost $100,000!

Details: I was having the best winning streak of my life! Up to $100,000 when the game crashed!! I was prompted to place a bet and I mistakenly entered text. When prompted again I placed my bet and the application crashed. Zero stars!

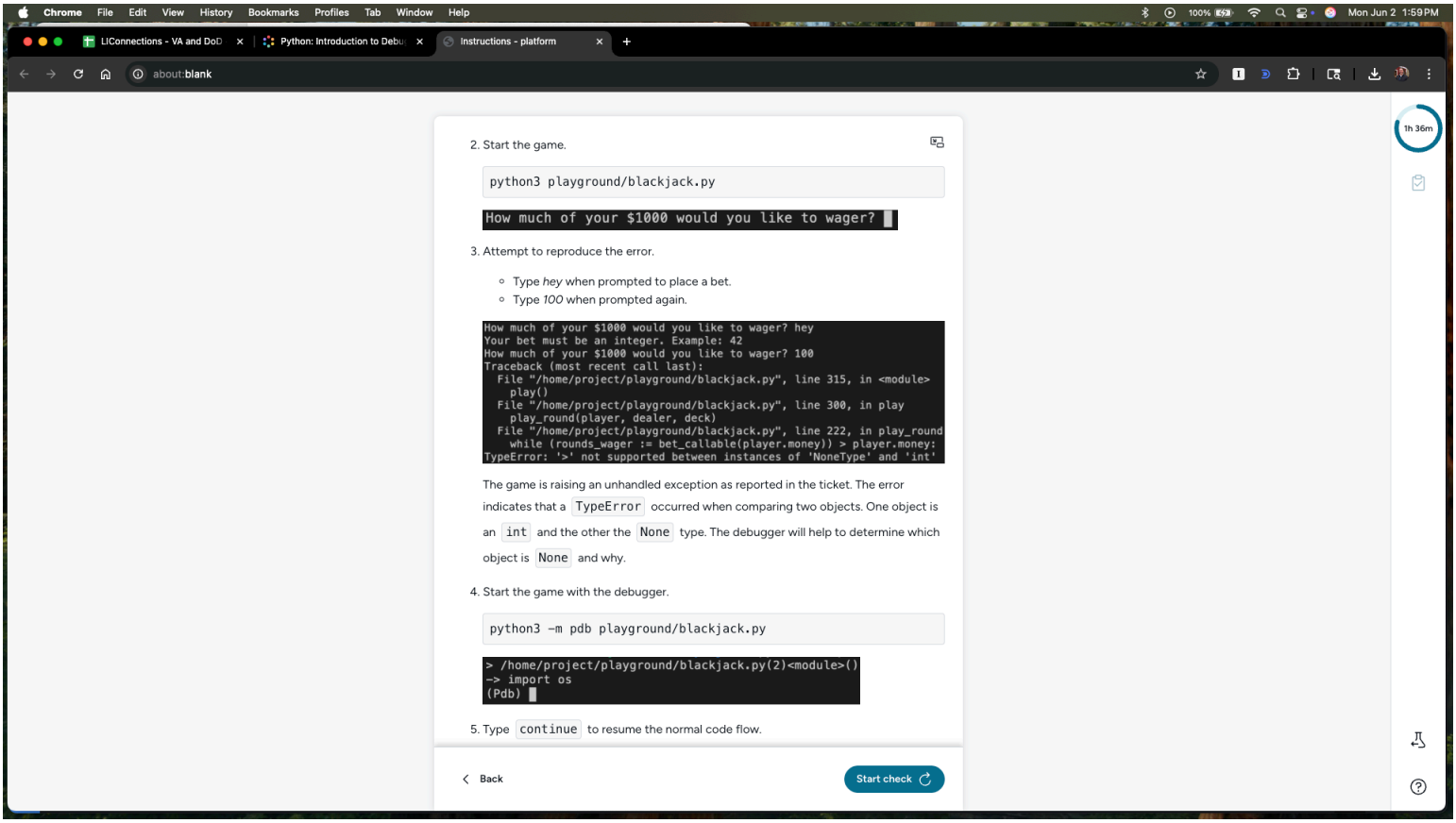
---

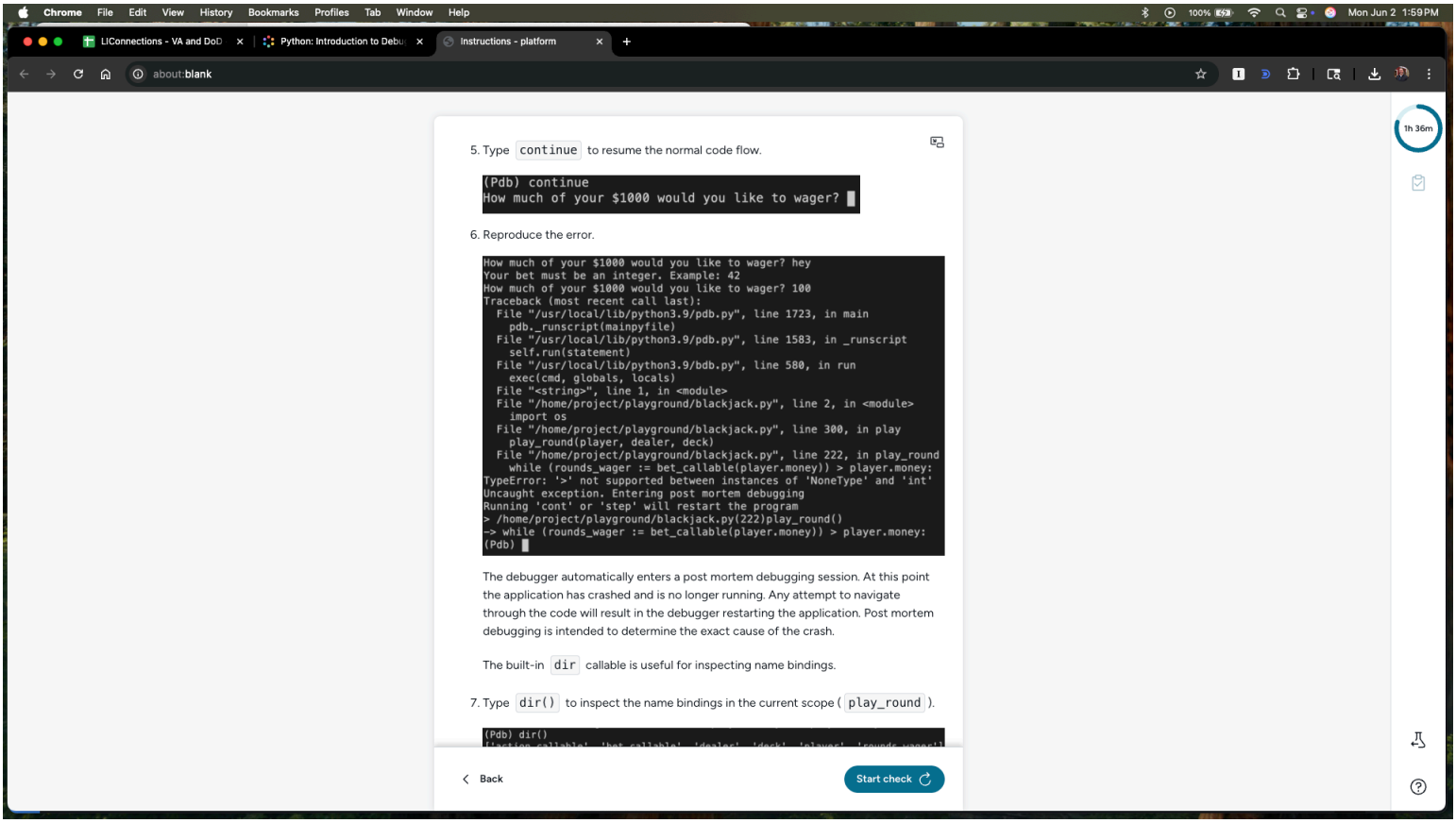
1. Locate and open the **playground/blackjack.py** file using the **IDE's Explorer/project** pane.

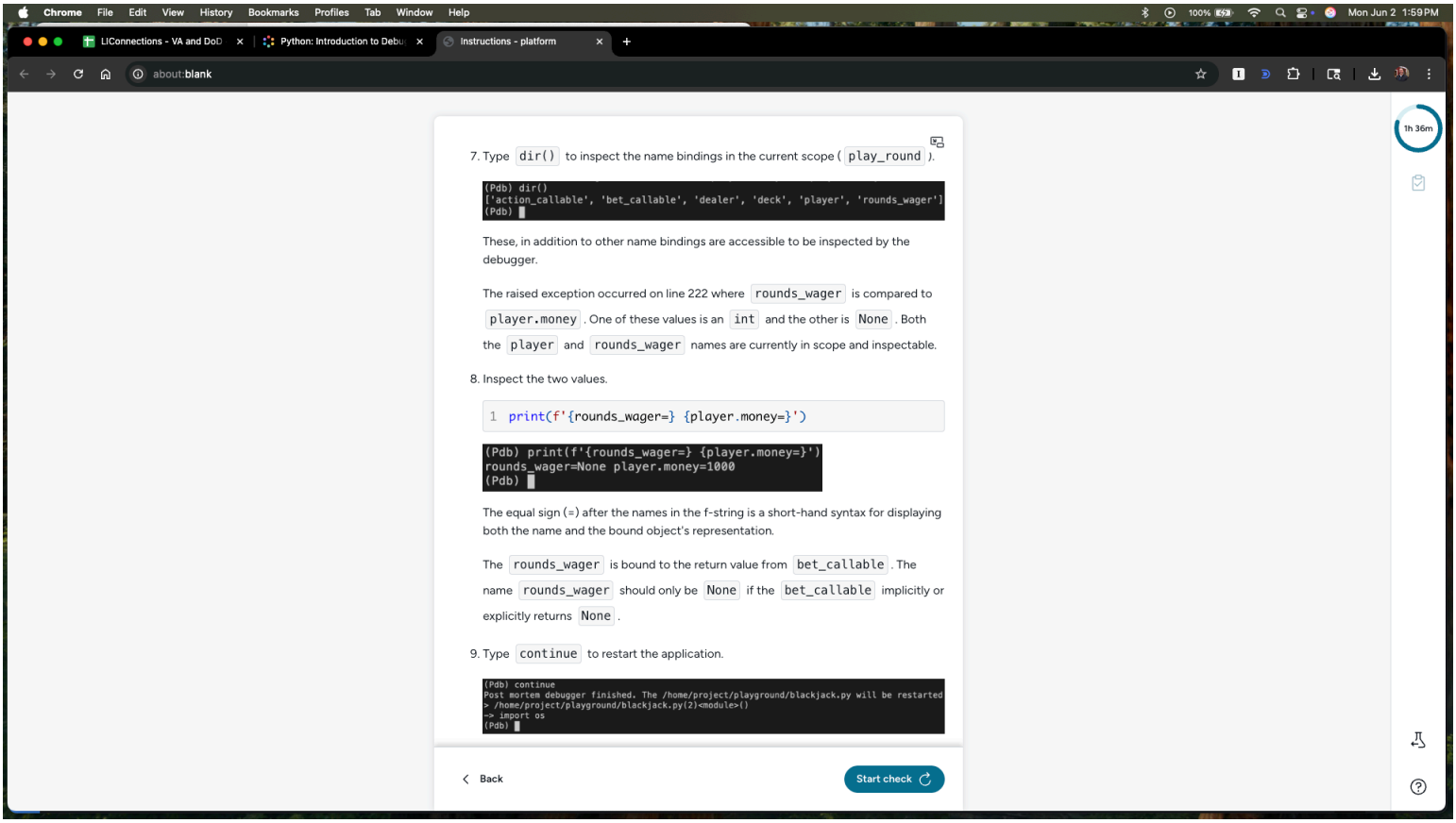


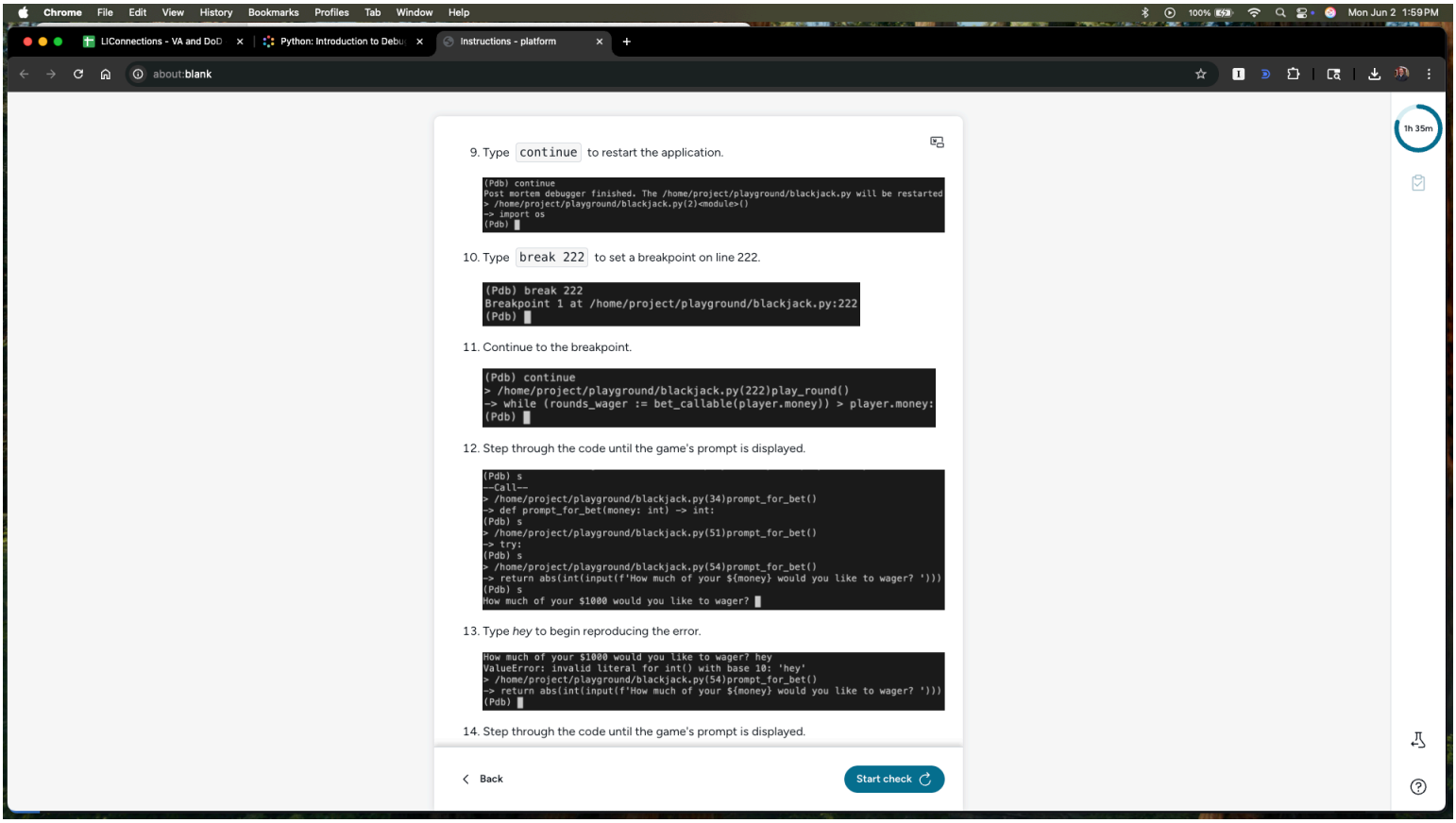
Back

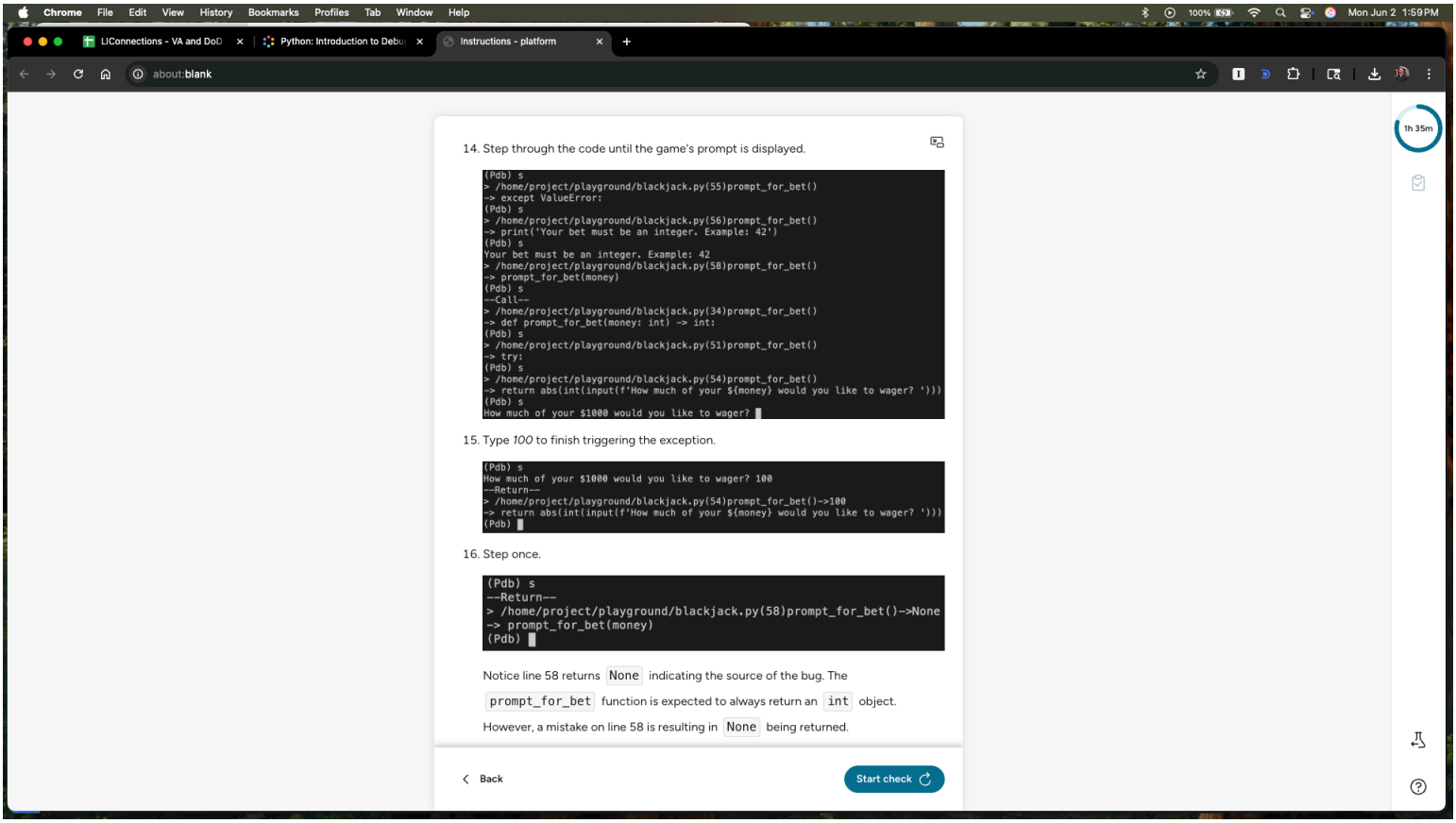
Start check

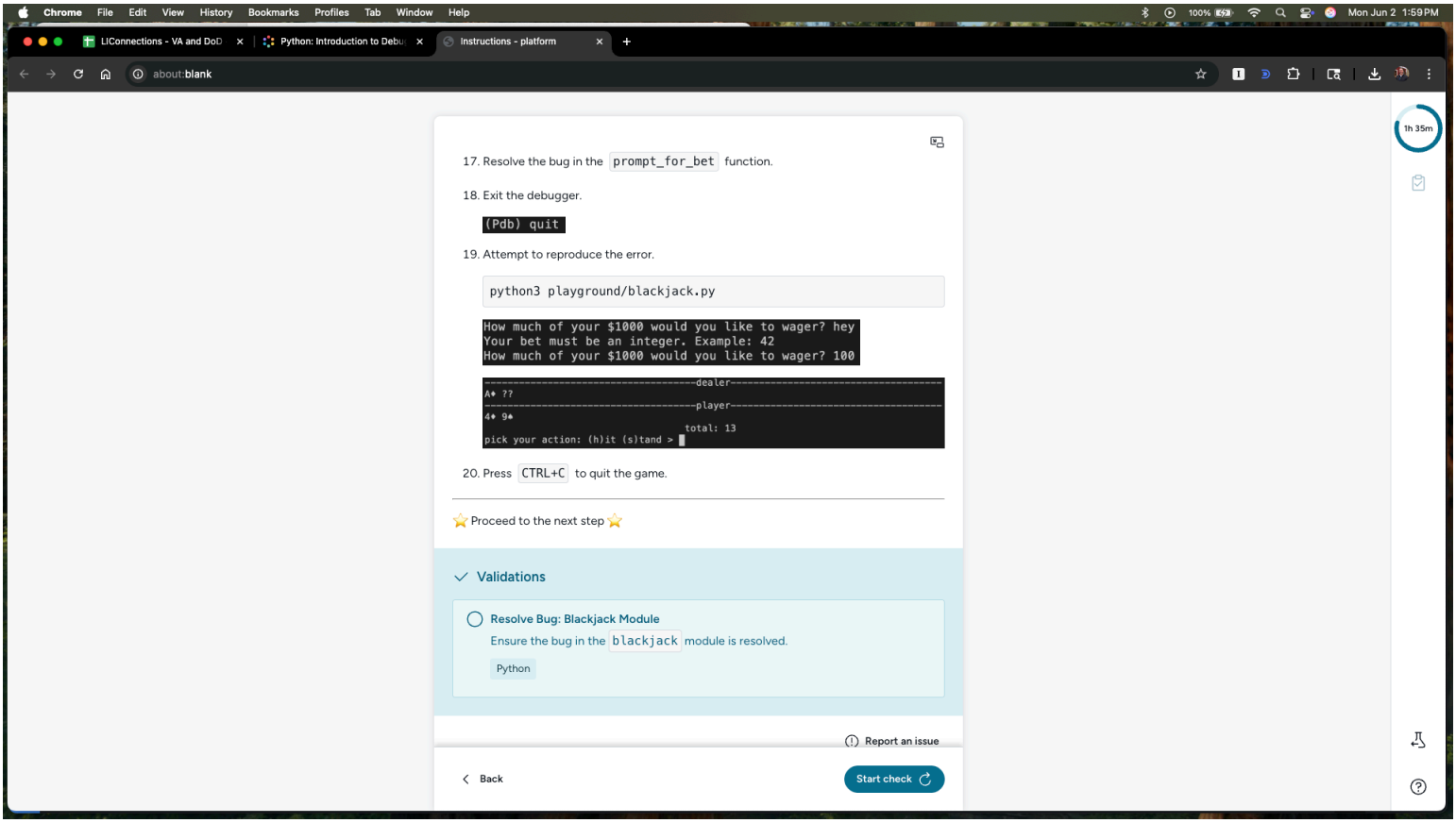

Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jun 2 1:59 PM
LIConnections - VA and DoD
Python: Introduction to Debu
Instructions - platform
about:blank
2. Start the game.
python3 playground/blackjack.py
How much of your $1000 would you like to wager?
3. Attempt to reproduce the error.
Type hey when prompted to place a bet.
Type 100 when prompted again.
How much of your $1000 would you like to wager? hey
Your bet must be an integer. Example: 42
How much of your $1000 would you like to wager? 100
Traceback (most recent call last):
File "/home/project/playground/blackjack.py", line 315, in <module>
play()
File "/home/project/playground/blackjack.py", line 300, in play
play_round(player, dealer, deck)
File "/home/project/playground/blackjack.py", line 222, in play_round
while (rounds_wager := bet_callable(player.money)) > player.money:
TypeError: '>' not supported between instances of 'NoneType' and 'int'
The game is raising an unhandled exception as reported in the ticket. The error indicates that a TypeError occurred when comparing two objects. One object is an int and the other the None type. The debugger will help to determine which object is None and why.
4. Start the game with the debugger.
python3 -m pdb playground/blackjack.py
> /home/project/playground/blackjack.py(2)<module>()
-> import os
(Pdb)
5. Type continue to resume the normal code flow.
Back
Start check
1h 36m


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jun 2 1:59 PM
LIConnections - VA and DoD
Python: Introduction to Debu
Instructions - platform
about:blank
1h 36m
5. Type continue to resume the normal code flow.
(Pdb) continue
How much of your $1000 would you like to wager?
6. Reproduce the error.
How much of your $1000 would you like to wager? hey
Your bet must be an integer. Example: 42
How much of your $1000 would you like to wager? 100
Traceback (most recent call last):
File "/usr/local/lib/python3.9/pdb.py", line 1723, in main
pdb._runscript(mainpyfile)
File "/usr/local/lib/python3.9/pdb.py", line 1583, in _runscript
self.run(statement)
File "/usr/local/lib/python3.9/bdb.py", line 580, in run
exec(cmd, globals, locals)
File "<string>", line 1, in <module>
File "/home/project/playground/blackjack.py", line 2, in <module>
import os
File "/home/project/playground/blackjack.py", line 300, in play
play_round(player, dealer, deck)
File "/home/project/playground/blackjack.py", line 222, in play_round
while (rounds_wager := bet_callable(player.money)) > player.money:
TypeError: '>' not supported between instances of 'NoneType' and 'int'
Uncaught exception. Entering post mortem debugging
Running 'cont' or 'step' will restart the program
> /home/project/playground/blackjack.py(222)play_round()
-> while (rounds_wager := bet_callable(player.money)) > player.money:
(Pdb)
The debugger automatically enters a post mortem debugging session. At this point the application has crashed and is no longer running. Any attempt to navigate through the code will result in the debugger restarting the application. Post mortem debugging is intended to determine the exact cause of the crash.
The built-in dir callable is useful for inspecting name bindings.
7. Type dir() to inspect the name bindings in the current scope (play_round).
(Pdb) dir()
Back
Start check


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jun 2 1:59 PM
LIConnections - VA and DoD
Python: Introduction to Debu
Instructions - platform
about:blank
1h 36m
7. Type dir() to inspect the name bindings in the current scope (play_round).
(Pdb) dir()
['action_callable', 'bet_callable', 'dealer', 'deck', 'player', 'rounds_wager']
(Pdb)
These, in addition to other name bindings are accessible to be inspected by the debugger.
The raised exception occurred on line 222 where rounds_wager is compared to player.money . One of these values is an int and the other is None . Both the player and rounds_wager names are currently in scope and inspectable.
8. Inspect the two values.
1 print(f'{rounds_wager=} {player.money=}')
(Pdb) print(f'{rounds_wager=} {player.money=}')
rounds_wager=None player.money=1000
(Pdb)
The equal sign (=) after the names in the f-string is a short-hand syntax for displaying both the name and the bound object's representation.
The rounds_wager is bound to the return value from bet_callable . The name rounds_wager should only be None if the bet_callable implicitly or explicitly returns None .
9. Type continue to restart the application.
(Pdb) continue
Post mortem debugger finished. The /home/project/playground/blackjack.py will be restarted
> /home/project/playground/blackjack.py(2)<module>()
-> import os
(Pdb)
Back
Start check


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jun 2 1:59 PM
LIConnections - VA and DoD
Python: Introduction to Debu
Instructions - platform
about:blank
1h 35m
9. Type continue to restart the application.
(Pdb) continue
Post mortem debugger finished. The /home/project/playground/blackjack.py will be restarted
> /home/project/playground/blackjack.py(2)<module>()
-> import os
(Pdb)
10. Type break 222 to set a breakpoint on line 222.
(Pdb) break 222
Breakpoint 1 at /home/project/playground/blackjack.py:222
(Pdb)
11. Continue to the breakpoint.
(Pdb) continue
> /home/project/playground/blackjack.py(222)play_round()
-> while (rounds_wager := bet_callable(player.money)) > player.money:
(Pdb)
12. Step through the code until the game's prompt is displayed.
(Pdb) s
--Call--
> /home/project/playground/blackjack.py(34)prompt_for_bet()
-> def prompt_for_bet(money: int) -> int:
(Pdb) s
> /home/project/playground/blackjack.py(51)prompt_for_bet()
-> try:
(Pdb) s
> /home/project/playground/blackjack.py(54)prompt_for_bet()
-> return abs(int(input(f'How much of your ${money} would you like to wager? ')))
(Pdb) s
How much of your $1000 would you like to wager?
13. Type hey to begin reproducing the error.
How much of your $1000 would you like to wager? hey
ValueError: invalid literal for int() with base 10: 'hey'
> /home/project/playground/blackjack.py(54)prompt_for_bet()
-> return abs(int(input(f'How much of your ${money} would you like to wager? ')))
(Pdb)
14. Step through the code until the game's prompt is displayed.
Back
Start check


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jun 2 1:59 PM
LIConnections - VA and DoD
Python: Introduction to Debu
Instructions - platform
about:blank
14. Step through the code until the game's prompt is displayed.
(Pdb) s
> /home/project/playground/blackjack.py(55)prompt_for_bet()
-> except ValueError:
(Pdb) s
> /home/project/playground/blackjack.py(56)prompt_for_bet()
-> print('Your bet must be an integer. Example: 42')
(Pdb) s
Your bet must be an integer. Example: 42
> /home/project/playground/blackjack.py(58)prompt_for_bet()
-> prompt_for_bet(money)
(Pdb) s
--Call--
> /home/project/playground/blackjack.py(34)prompt_for_bet()
-> def prompt_for_bet(money: int) -> int:
(Pdb) s
> /home/project/playground/blackjack.py(51)prompt_for_bet()
-> try:
(Pdb) s
> /home/project/playground/blackjack.py(54)prompt_for_bet()
-> return abs(int(input(f'How much of your ${money} would you like to wager? ')))
(Pdb) s
How much of your $1000 would you like to wager?
15. Type 100 to finish triggering the exception.
(Pdb) s
How much of your $1000 would you like to wager? 100
--Return--
> /home/project/playground/blackjack.py(54)prompt_for_bet()->100
-> return abs(int(input(f'How much of your ${money} would you like to wager? ')))
(Pdb)
16. Step once.
(Pdb) s
--Return--
> /home/project/playground/blackjack.py(58)prompt_for_bet()->None
-> prompt_for_bet(money)
(Pdb)
Notice line 58 returns None indicating the source of the bug. The prompt_for_bet function is expected to always return an int object. However, a mistake on line 58 is resulting in None being returned.
Back
Start check
1h 35m


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon Jun 2 1:59 PM
LIConnections - VA and DoD
Python: Introduction to Debu
Instructions - platform
about:blank
1h 35m
17. Resolve the bug in the prompt_for_bet function.
18. Exit the debugger.
(Pdb) quit
19. Attempt to reproduce the error.
python3 playground/blackjack.py
How much of your $1000 would you like to wager? hey
Your bet must be an integer. Example: 42
How much of your $1000 would you like to wager? 100
dealer
A♦ ??
player
4♦ 9♠
total: 13
pick your action: (h)it (s)tand >
20. Press CTRL+C to quit the game.
Proceed to the next step
Validations
Resolve Bug: Blackjack Module
Ensure the bug in the blackjack module is resolved.
Python
Report an issue
Back
Start check