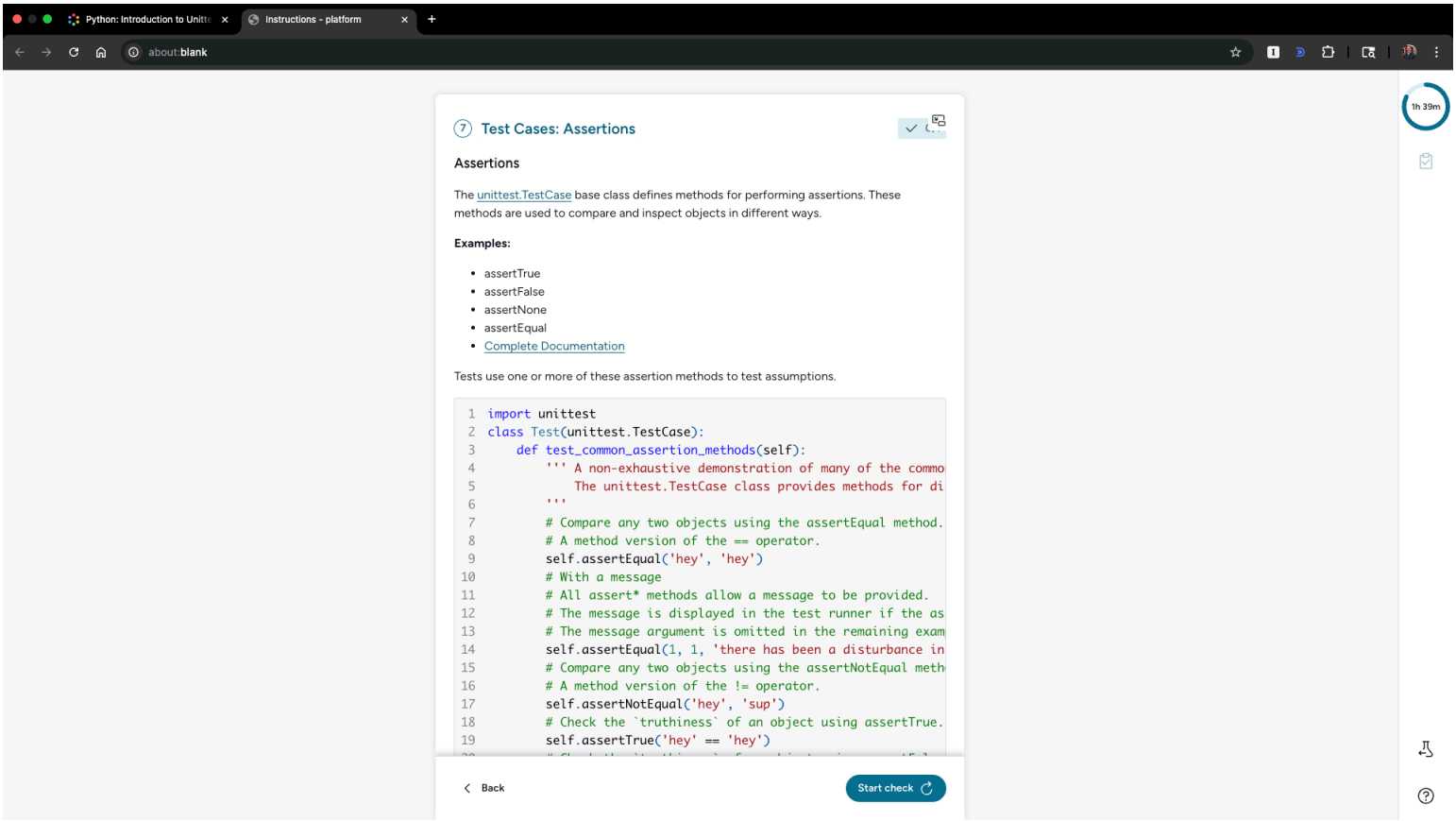
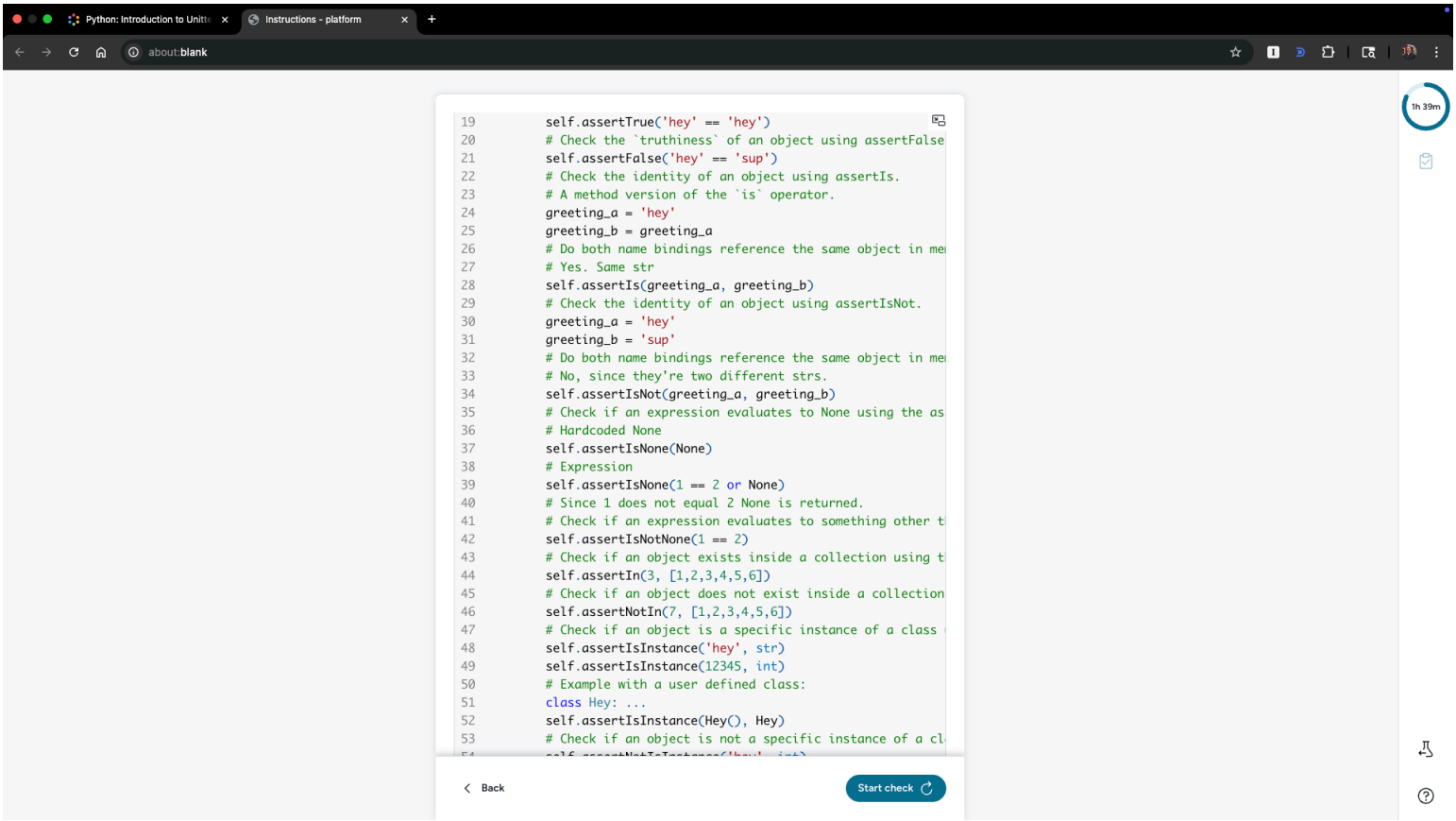
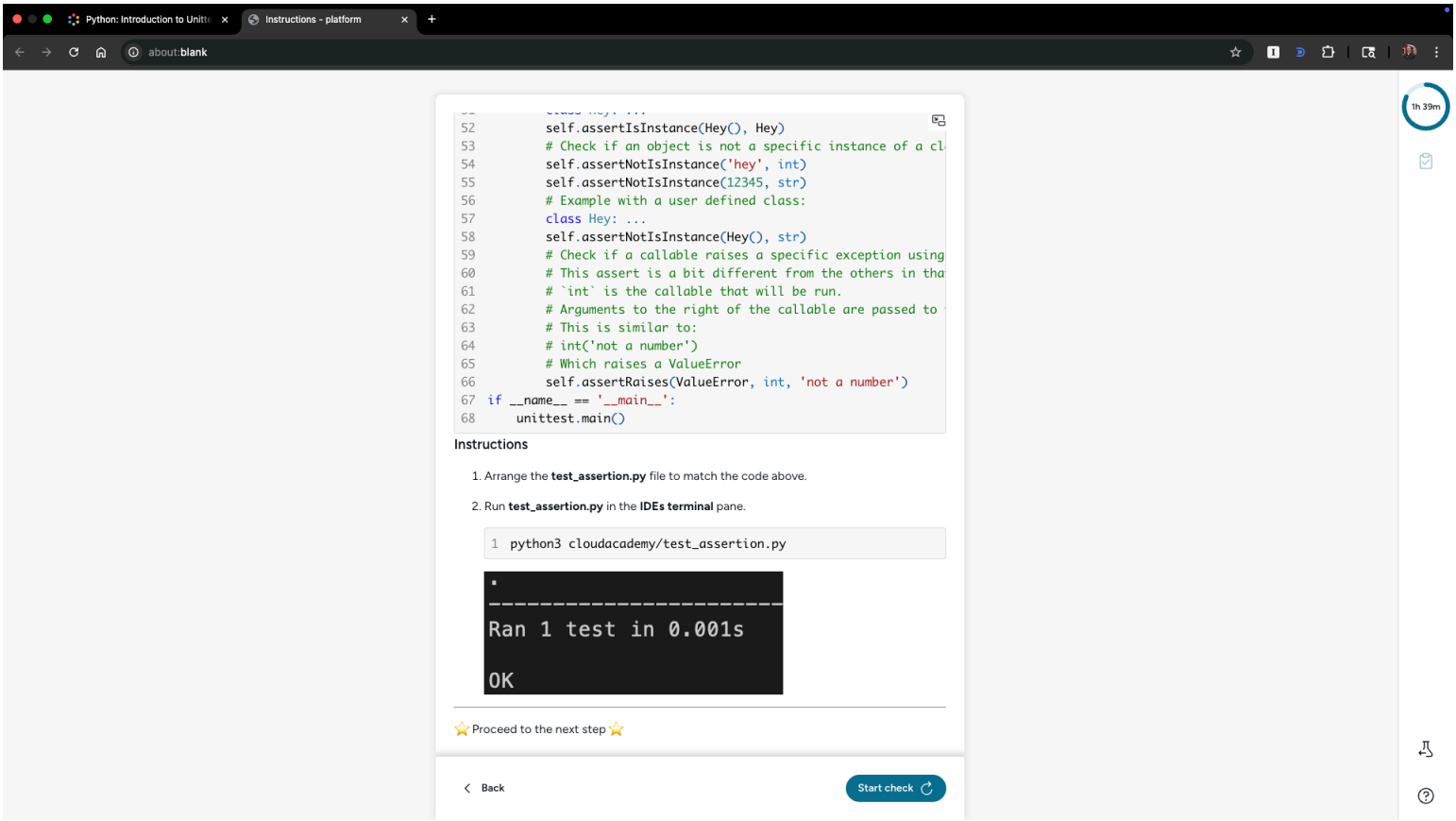

Python: Introduction to Unitte
Instructions - platform
about:blank
1h 39m
7 Test Cases: Assertions
Assertions
The unittest.TestCase base class defines methods for performing assertions. These methods are used to compare and inspect objects in different ways.
Examples:
assertTrue
assertFalse
assertNone
assertEqual
Complete Documentation
Tests use one or more of these assertion methods to test assumptions.
1 import unittest
2 class Test(unittest.TestCase):
3 def test_common_assertion_methods(self):
4 ''' A non-exhaustive demonstration of many of the commo
5 The unittest.TestCase class provides methods for di
6 '''
7 # Compare any two objects using the assertEqual method.
8 # A method version of the == operator.
9 self.assertEqual('hey', 'hey')
10 # With a message
11 # All assert* methods allow a message to be provided.
12 # The message is displayed in the test runner if the as
13 # The message argument is omitted in the remaining exam
14 self.assertEqual(1, 1, 'there has been a disturbance in
15 # Compare any two objects using the assertNotEqual meth
16 # A method version of the != operator.
17 self.assertNotEqual('hey', 'sup')
18 # Check the `truthiness` of an object using assertTrue.
19 self.assertTrue('hey' == 'hey')
Back
Start check




```
        self.assertTrue('hey' == 'hey')
        # Check the `truthiness` of an object using assertFalse
        self.assertFalse('hey' == 'sup')
        # Check the identity of an object using assertIs.
        # A method version of the `is` operator.
        greeting_a = 'hey'
        greeting_b = greeting_a
        # Do both name bindings reference the same object in me
        # Yes. Same str
        self.assertIs(greeting_a, greeting_b)
        # Check the identity of an object using assertIsNot.
        greeting_a = 'hey'
        greeting_b = 'sup'
        # Do both name bindings reference the same object in me
        # No, since they're two different strs.
        self.assertIsNot(greeting_a, greeting_b)
        # Check if an expression evaluates to None using the as
        # Hardcoded None
        self.assertIsNone(None)
        # Expression
        self.assertIsNone(1 == 2 or None)
        # Since 1 does not equal 2 None is returned.
        # Check if an expression evaluates to something other t
        self.assertIsNotNone(1 == 2)
        # Check if an object exists inside a collection using t
        self.assertIn(3, [1,2,3,4,5,6])
        # Check if an object does not exist inside a collection
        self.assertNotIn(7, [1,2,3,4,5,6])
        # Check if an object is a specific instance of a class
        self.assertIsInstance('hey', str)
        self.assertIsInstance(12345, int)
        # Example with a user defined class:
        class Hey: ...
        self.assertIsInstance(Hey(), Hey)
        # Check if an object is not a specific instance of a cl
```


Python: Introduction to Unitte
Instructions - platform
about:blank
1h 39m
52 self.assertIsInstance(Hey(), Hey)
53 # Check if an object is not a specific instance of a cl
54 self.assertNotIsInstance('hey', int)
55 self.assertNotIsInstance(12345, str)
56 # Example with a user defined class:
57 class Hey: ...
58 self.assertNotIsInstance(Hey(), str)
59 # Check if a callable raises a specific exception using
60 # This assert is a bit different from the others in tha
61 # `int` is the callable that will be run.
62 # Arguments to the right of the callable are passed to
63 # This is similar to:
64 # int('not a number')
65 # Which raises a ValueError
66 self.assertRaises(ValueError, int, 'not a number')
67 if __name__ == '__main__':
68 unittest.main()
Instructions
1. Arrange the test_assertion.py file to match the code above.
2. Run test_assertion.py in the IDEs terminal pane.
1 python3 cloudacademy/test_assertion.py
.
Ran 1 test in 0.001s
OK
Proceed to the next step
Back
Start check