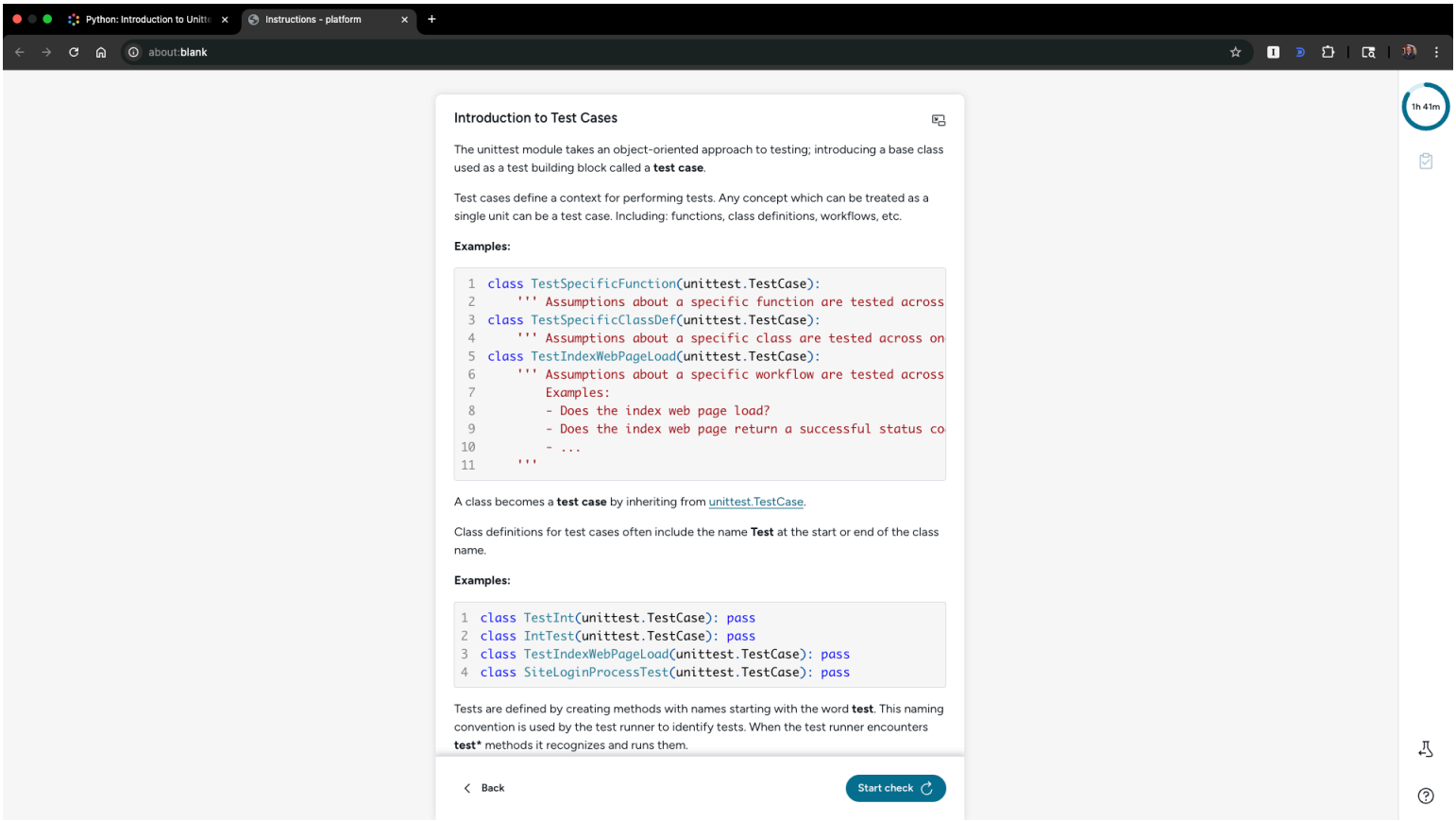
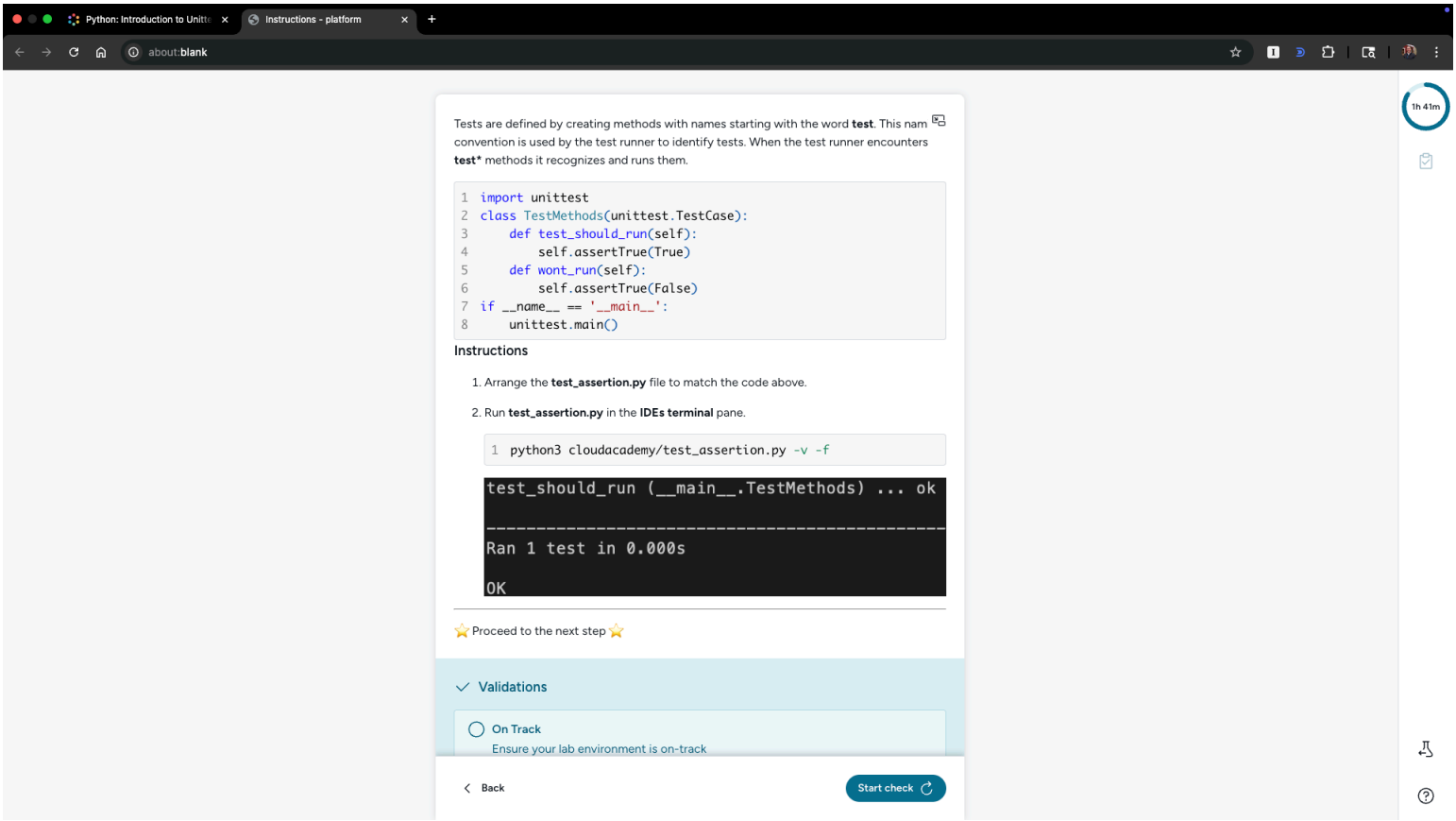

Python: Introduction to Unitte
Instructions - platform
about:blank
Introduction to Test Cases
The unittest module takes an object-oriented approach to testing; introducing a base class used as a test building block called a **test case**.
Test cases define a context for performing tests. Any concept which can be treated as a single unit can be a test case. Including: functions, class definitions, workflows, etc.
Examples:
class TestSpecificFunction(unittest.TestCase):
''' Assumptions about a specific function are tested across
class TestSpecificClassDef(unittest.TestCase):
''' Assumptions about a specific class are tested across on
class TestIndexWebPageLoad(unittest.TestCase):
''' Assumptions about a specific workflow are tested across
Examples:
- Does the index web page load?
- Does the index web page return a successful status co
- ...
'''
A class becomes a **test case** by inheriting from unittest.TestCase.
Class definitions for test cases often include the name **Test** at the start or end of the class name.
Examples:
class TestInt(unittest.TestCase): pass
class IntTest(unittest.TestCase): pass
class TestIndexWebPageLoad(unittest.TestCase): pass
class SiteLoginProcessTest(unittest.TestCase): pass
Tests are defined by creating methods with names starting with the word **test**. This naming convention is used by the test runner to identify tests. When the test runner encounters **test*** methods it recognizes and runs them.
Back
Start check
1h 41m


Python: Introduction to Unitte
Instructions - platform
about:blank
Tests are defined by creating methods with names starting with the word test. This nam
convention is used by the test runner to identify tests. When the test runner encounters
test* methods it recognizes and runs them.
1 import unittest
2 class TestMethods(unittest.TestCase):
3 def test_should_run(self):
4 self.assertTrue(True)
5 def wont_run(self):
6 self.assertTrue(False)
7 if __name__ == '__main__':
8 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 -v -f
test_should_run (__main__.TestMethods) ... ok
Ran 1 test in 0.000s
OK
Proceed to the next step
Validations
On Track
Ensure your lab environment is on-track
Back
Start check
1h 41m