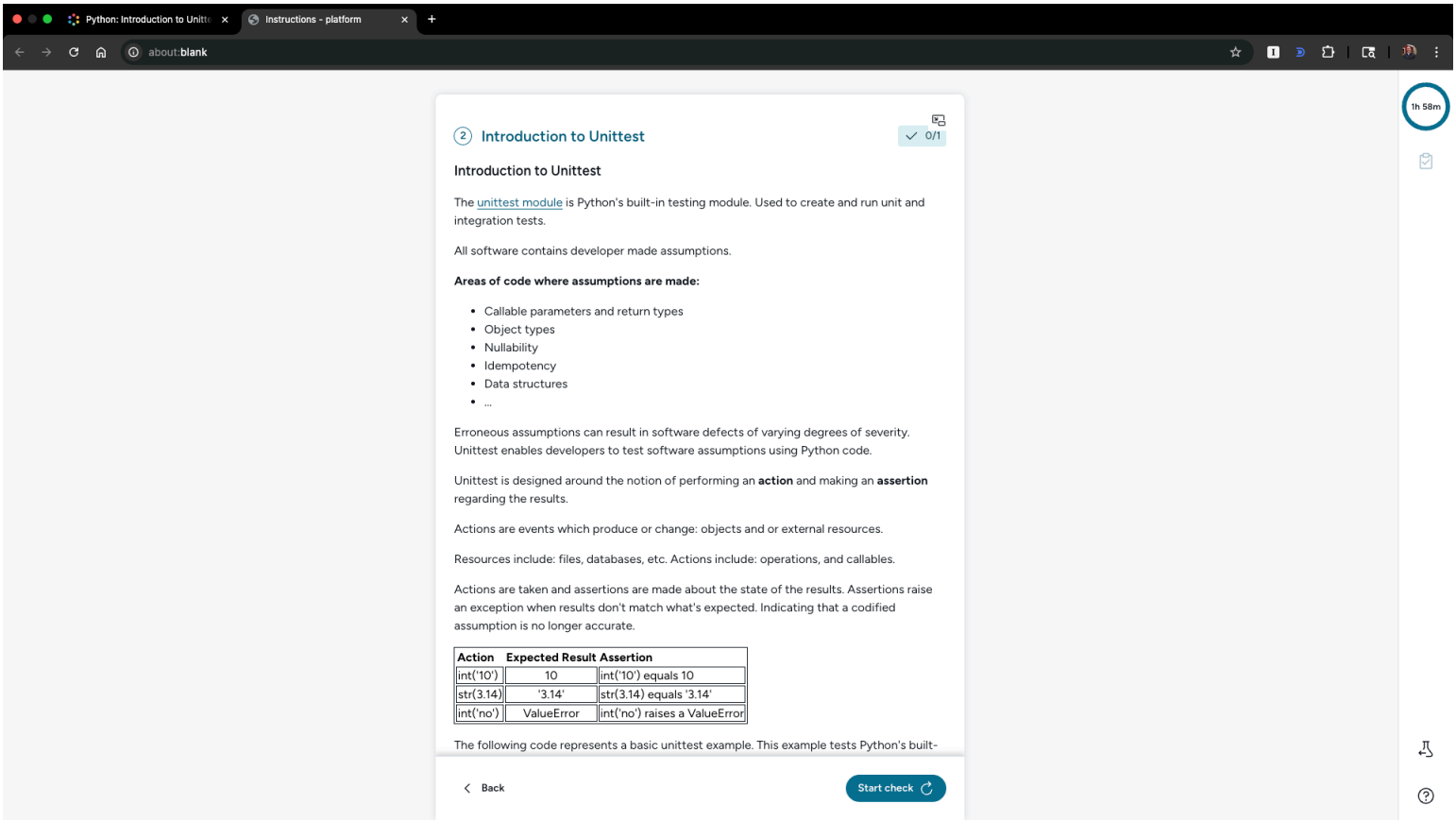
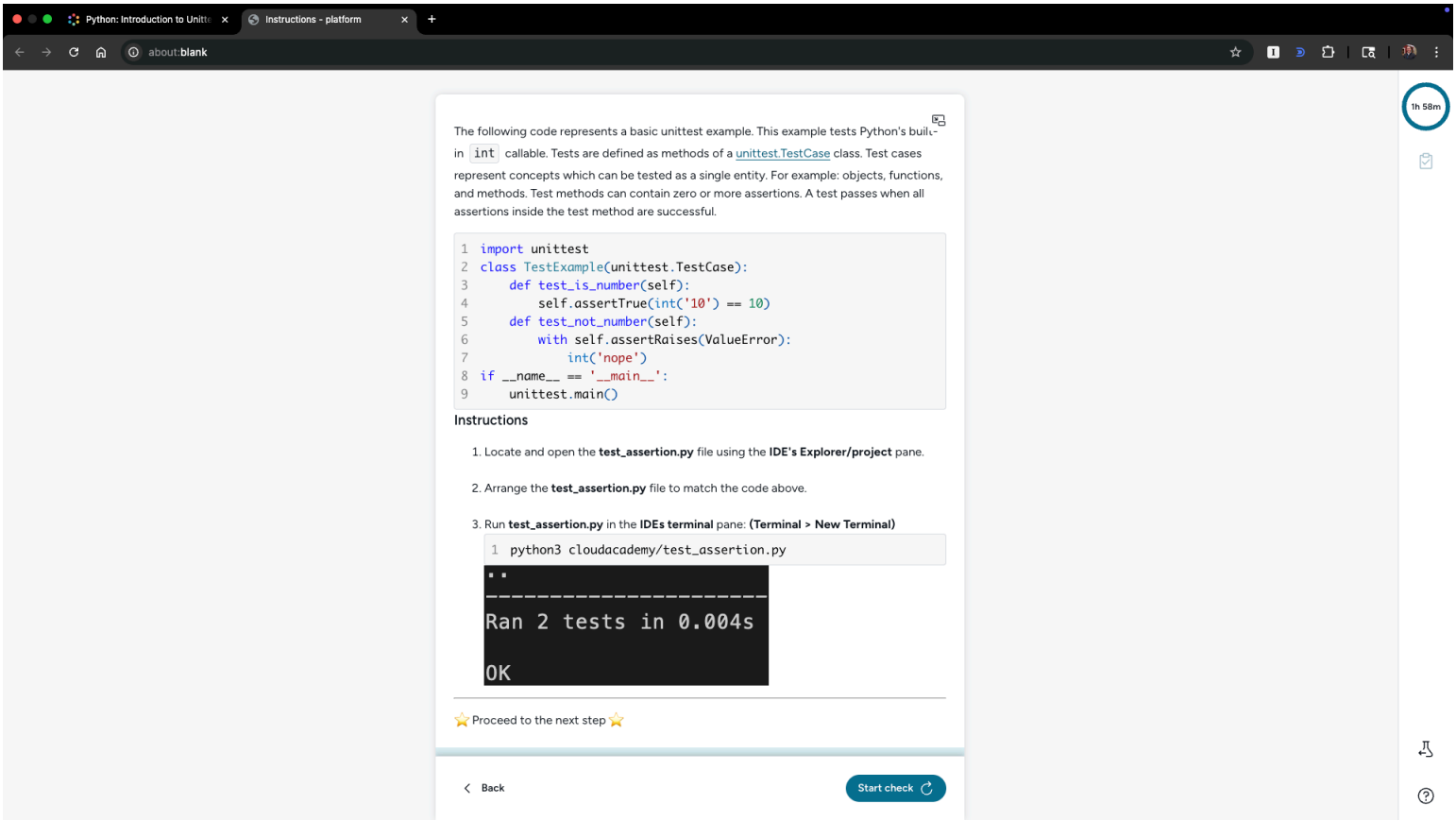

Python: Introduction to Unitte
Instructions - platform
about:blank
1h 58m
2 Introduction to Unittest
0/1
Introduction to Unittest
The unittest module is Python's built-in testing module. Used to create and run unit and integration tests.
All software contains developer made assumptions.
Areas of code where assumptions are made:
Callable parameters and return types
Object types
Nullability
Idempotency
Data structures
...
Erroneous assumptions can result in software defects of varying degrees of severity. Unittest enables developers to test software assumptions using Python code.
Unittest is designed around the notion of performing an action and making an assertion regarding the results.
Actions are events which produce or change: objects and or external resources.
Resources include: files, databases, etc. Actions include: operations, and callables.
Actions are taken and assertions are made about the state of the results. Assertions raise an exception when results don't match what's expected. Indicating that a codified assumption is no longer accurate.
Action Expected Result Assertion
int('10') 10 int('10') equals 10
str(3.14) '3.14' str(3.14) equals '3.14'
int('no') ValueError int('no') raises a ValueError
The following code represents a basic unittest example. This example tests Python's built-
Back
Start check


Python: Introduction to Unitte
Instructions - platform
about:blank
1h 58m
The following code represents a basic unittest example. This example tests Python's built-in int callable. Tests are defined as methods of a unittest.TestCase class. Test cases represent concepts which can be tested as a single entity. For example: objects, functions, and methods. Test methods can contain zero or more assertions. A test passes when all assertions inside the test method are successful.
1 import unittest
2 class TestExample(unittest.TestCase):
3 def test_is_number(self):
4 self.assertTrue(int('10') == 10)
5 def test_not_number(self):
6 with self.assertRaises(ValueError):
7 int('nope')
8 if __name__ == '__main__':
9 unittest.main()
Instructions
1. Locate and open the test_assertion.py file using the IDE's Explorer/project pane.
2. Arrange the test_assertion.py file to match the code above.
3. Run test_assertion.py in the IDEs terminal pane: (Terminal > New Terminal)
1 python3 cloudacademy/test_assertion.py
..

Ran 2 tests in 0.004s
OK
Proceed to the next step
Back
Start check