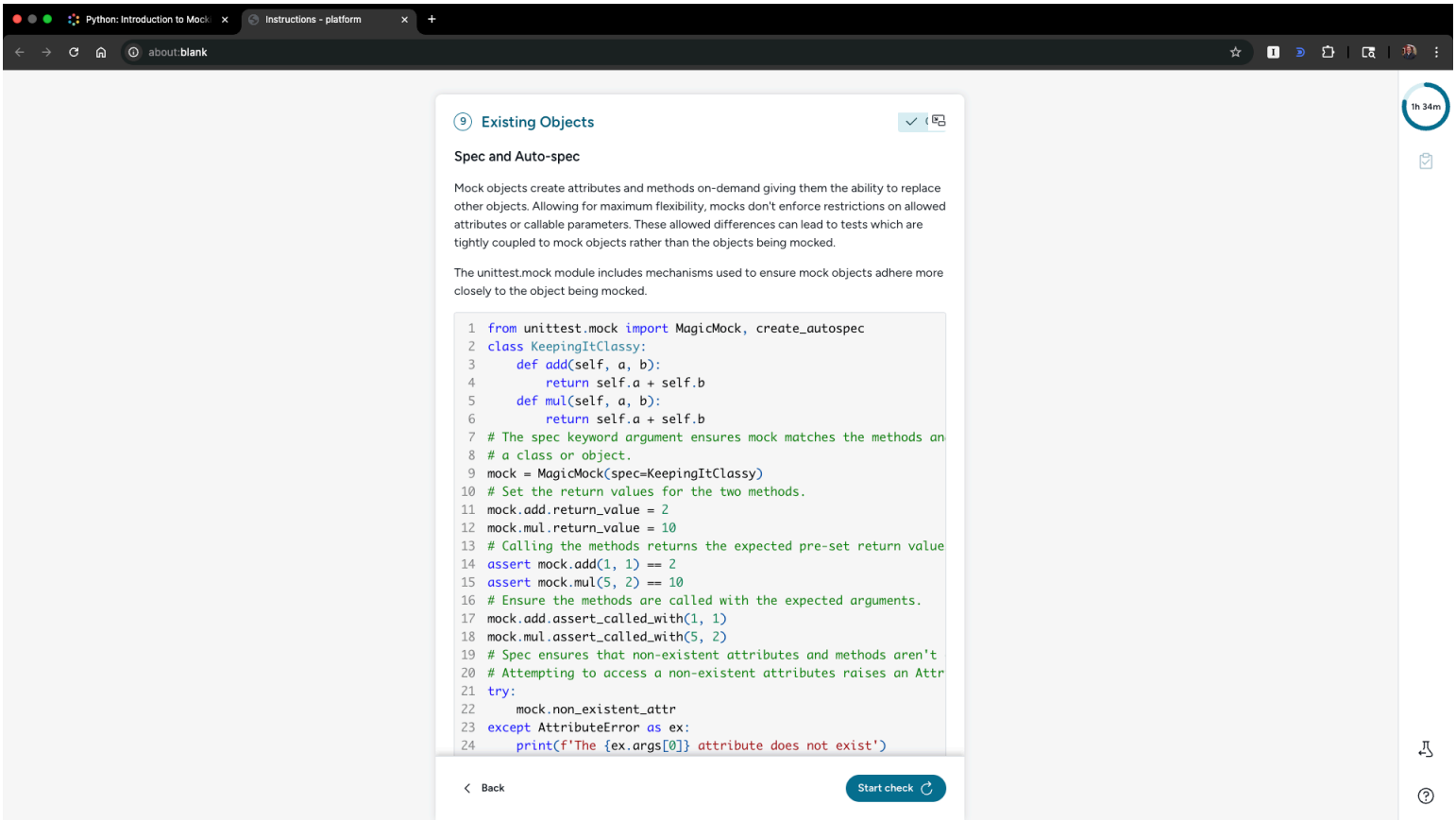
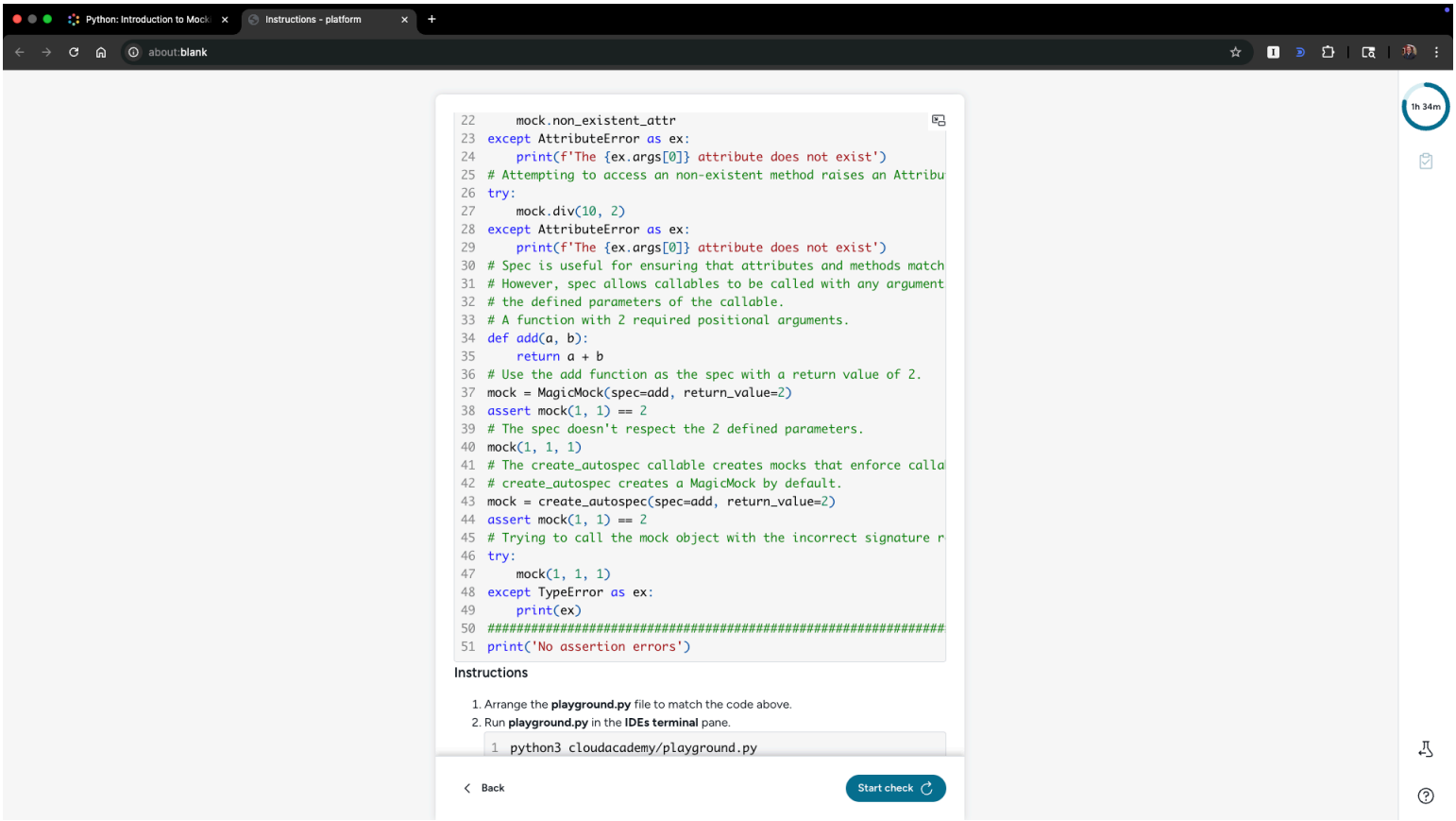

Python: Introduction to Mock
Instructions - platform
about:blank
9 Existing Objects
Spec and Auto-spec
Mock objects create attributes and methods on-demand giving them the ability to replace other objects. Allowing for maximum flexibility, mocks don't enforce restrictions on allowed attributes or callable parameters. These allowed differences can lead to tests which are tightly coupled to mock objects rather than the objects being mocked.
The unittest.mock module includes mechanisms used to ensure mock objects adhere more closely to the object being mocked.
from unittest.mock import MagicMock, create_autospec
class KeepingItClassy:
def add(self, a, b):
return self.a + self.b
def mul(self, a, b):
return self.a + self.b
# The spec keyword argument ensures mock matches the methods an
# a class or object.
mock = MagicMock(spec=KeepingItClassy)
# Set the return values for the two methods.
mock.add.return_value = 2
mock.mul.return_value = 10
# Calling the methods returns the expected pre-set return value
assert mock.add(1, 1) == 2
assert mock.mul(5, 2) == 10
# Ensure the methods are called with the expected arguments.
mock.add.assert_called_with(1, 1)
mock.mul.assert_called_with(5, 2)
# Spec ensures that non-existent attributes and methods aren't
# Attempting to access a non-existent attributes raises an Attr
try:
mock.non_existent_attr
except AttributeError as ex:
print(f'The {ex.args[0]} attribute does not exist')
Back
Start check
1h 34m


Python: Introduction to Mock
Instructions - platform
about:blank
1h 34m
22 mock.non_existent_attr
23 except AttributeError as ex:
24 print(f'The {ex.args[0]} attribute does not exist')
25 # Attempting to access an non-existent method raises an Attribu
26 try:
27 mock.div(10, 2)
28 except AttributeError as ex:
29 print(f'The {ex.args[0]} attribute does not exist')
30 # Spec is useful for ensuring that attributes and methods match
31 # However, spec allows callables to be called with any argument
32 # the defined parameters of the callable.
33 # A function with 2 required positional arguments.
34 def add(a, b):
35 return a + b
36 # Use the add function as the spec with a return value of 2.
37 mock = MagicMock(spec=add, return_value=2)
38 assert mock(1, 1) == 2
39 # The spec doesn't respect the 2 defined parameters.
40 mock(1, 1, 1)
41 # The create_autospec callable creates mocks that enforce calla
42 # create_autospec creates a MagicMock by default.
43 mock = create_autospec(spec=add, return_value=2)
44 assert mock(1, 1) == 2
45 # Trying to call the mock object with the incorrect signature r
46 try:
47 mock(1, 1, 1)
48 except TypeError as ex:
49 print(ex)
50 ##
51 print('No assertion errors')
Instructions
1. Arrange the playground.py file to match the code above.
2. Run playground.py in the IDEs terminal pane.
1 python3 cloudacademy/playground.py
Back
Start check