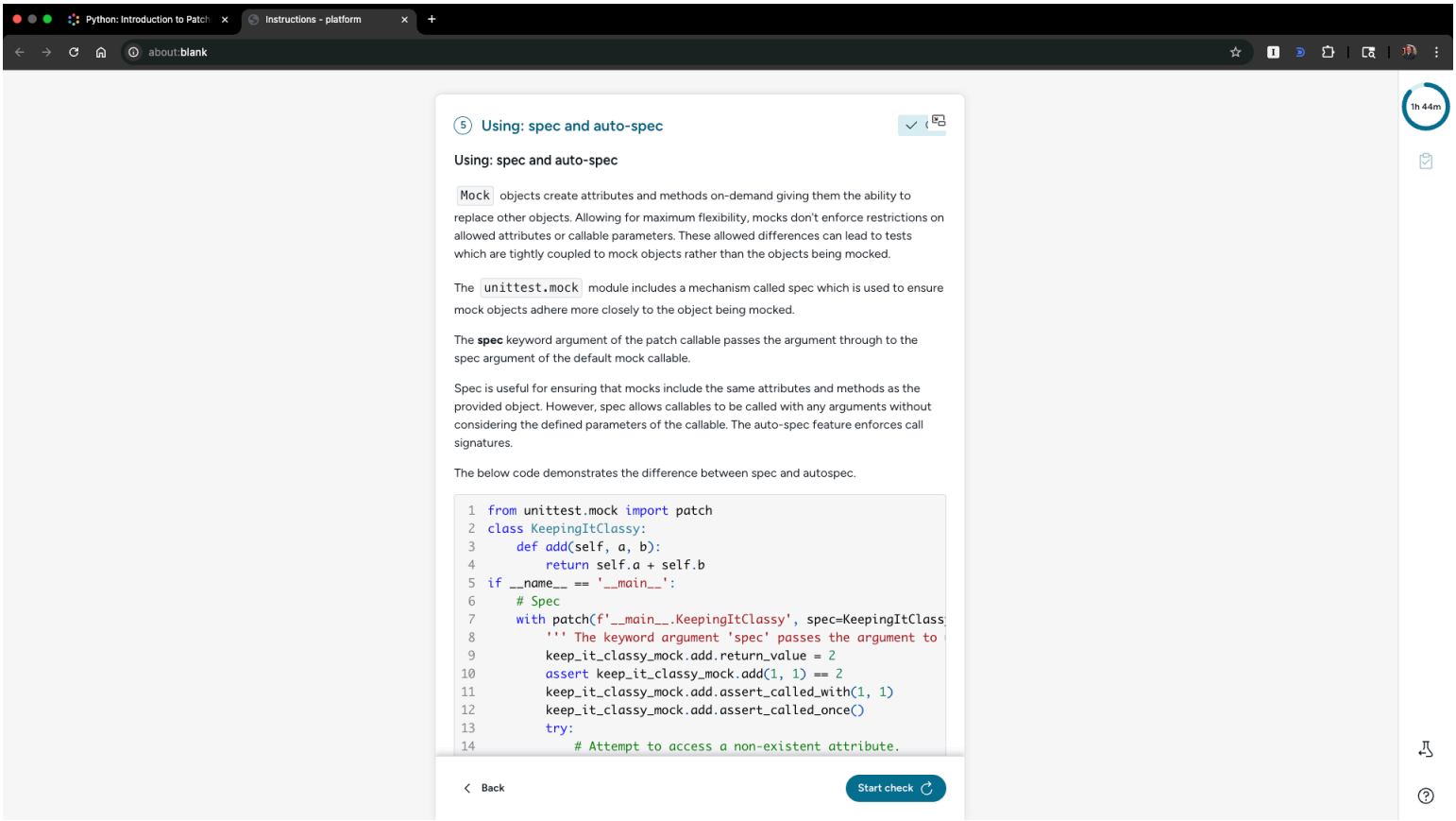
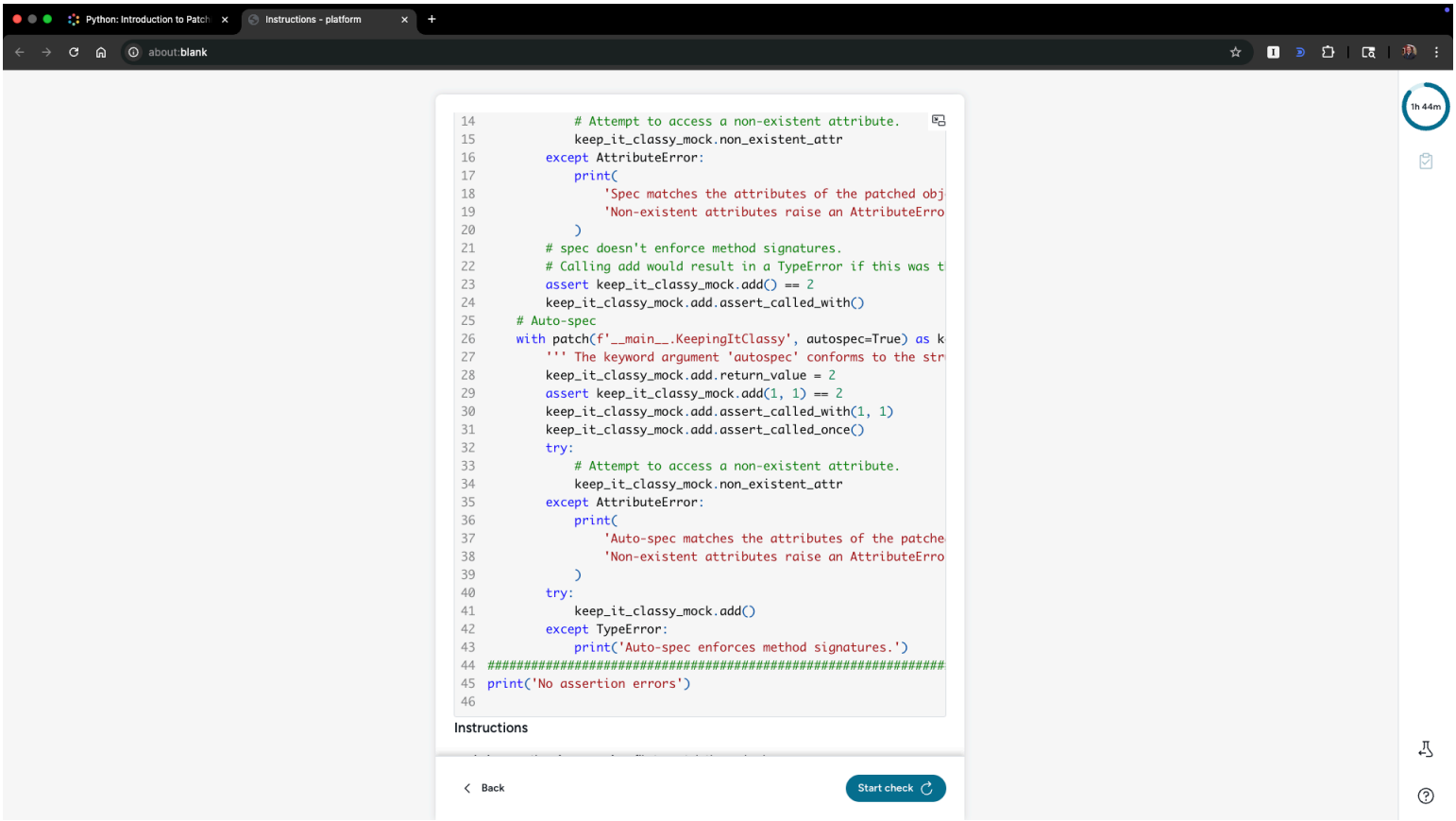
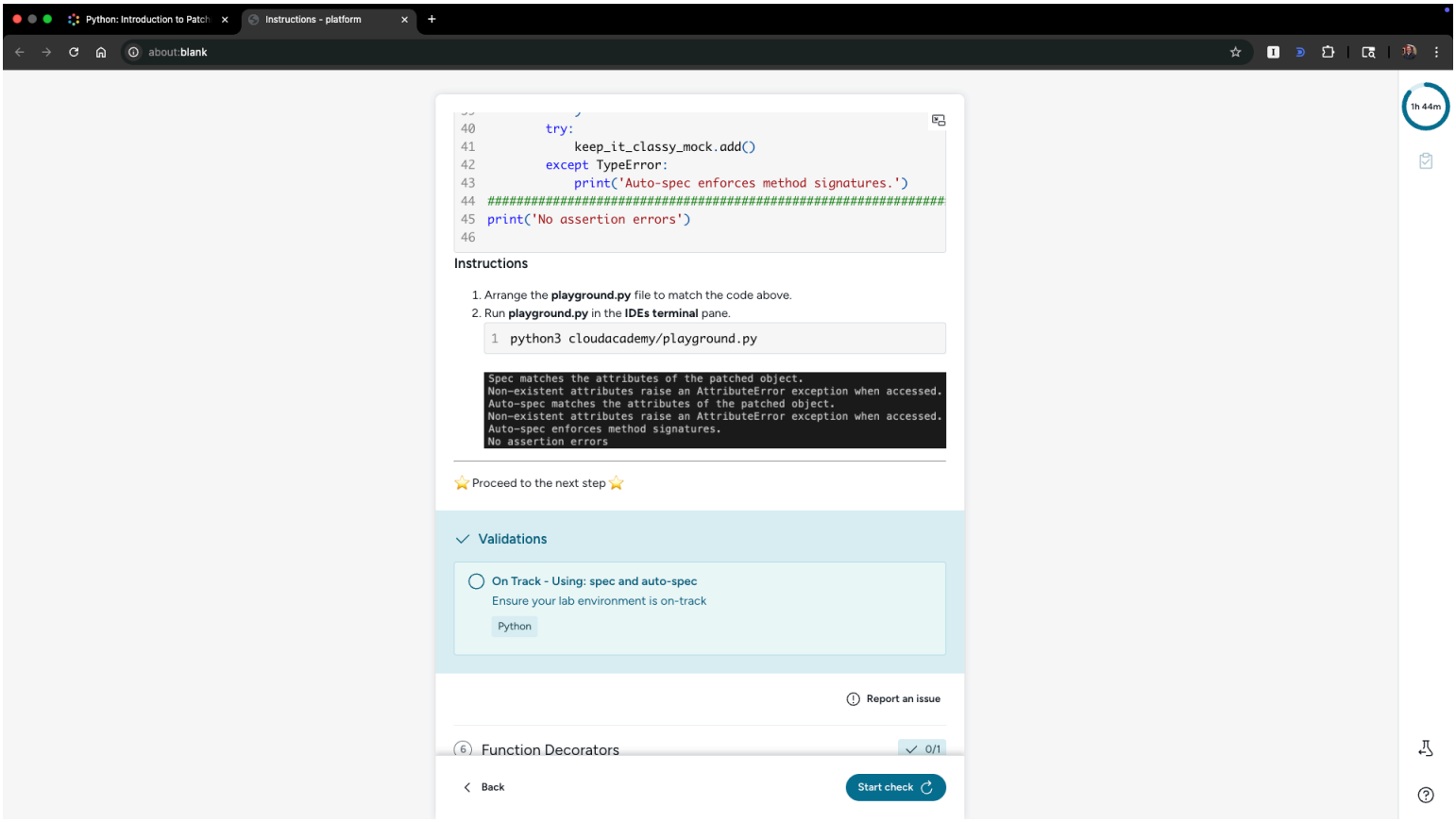

Python: Introduction to Patch
Instructions - platform
about:blank
5 Using: spec and auto-spec
Using: spec and auto-spec
Mock objects create attributes and methods on-demand giving them the ability to replace other objects. Allowing for maximum flexibility, mocks don't enforce restrictions on allowed 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 a mechanism called spec which is used to ensure mock objects adhere more closely to the object being mocked.
The spec keyword argument of the patch callable passes the argument through to the spec argument of the default mock callable.
Spec is useful for ensuring that mocks include the same attributes and methods as the provided object. However, spec allows callables to be called with any arguments without considering the defined parameters of the callable. The auto-spec feature enforces call signatures.
The below code demonstrates the difference between spec and autospec.
from unittest.mock import patch
class KeepingItClassy:
def add(self, a, b):
return self.a + self.b
if __name__ == '__main__':
# Spec
with patch(f'__main__.KeepingItClassy', spec=KeepingItClass
''' The keyword argument 'spec' passes the argument to
keep_it_classy_mock.add.return_value = 2
assert keep_it_classy_mock.add(1, 1) == 2
keep_it_classy_mock.add.assert_called_with(1, 1)
keep_it_classy_mock.add.assert_called_once()
try:
# Attempt to access a non-existent attribute.
Back
Start check
1h 44m


Python: Introduction to Patch
Instructions - platform
about:blank
1h 44m
14 # Attempt to access a non-existent attribute.
15 keep_it_classy_mock.non_existent_attr
16 except AttributeError:
17 print(
18 'Spec matches the attributes of the patched obj
19 'Non-existent attributes raise an AttributeErro
20)
21 # spec doesn't enforce method signatures.
22 # Calling add would result in a TypeError if this was t
23 assert keep_it_classy_mock.add() == 2
24 keep_it_classy_mock.add.assert_called_with()
25 # Auto-spec
26 with patch(f'__main__.KeepingItClassy', autospec=True) as k
27 ''' The keyword argument 'autospec' conforms to the str
28 keep_it_classy_mock.add.return_value = 2
29 assert keep_it_classy_mock.add(1, 1) == 2
30 keep_it_classy_mock.add.assert_called_with(1, 1)
31 keep_it_classy_mock.add.assert_called_once()
32 try:
33 # Attempt to access a non-existent attribute.
34 keep_it_classy_mock.non_existent_attr
35 except AttributeError:
36 print(
37 'Auto-spec matches the attributes of the patche
38 'Non-existent attributes raise an AttributeErro
39)
40 try:
41 keep_it_classy_mock.add()
42 except TypeError:
43 print('Auto-spec enforces method signatures.')
44 ##
45 print('No assertion errors')
46
Instructions
Back
Start check


Python: Introduction to Patch
Instructions - platform
about:blank
1h 44m
40 try:
41 keep_it_classy_mock.add()
42 except TypeError:
43 print('Auto-spec enforces method signatures.')
44 ##
45 print('No assertion errors')
46
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
Spec matches the attributes of the patched object.
Non-existent attributes raise an AttributeError exception when accessed.
Auto-spec matches the attributes of the patched object.
Non-existent attributes raise an AttributeError exception when accessed.
Auto-spec enforces method signatures.
No assertion errors
Proceed to the next step
Validations
On Track - Using: spec and auto-spec
Ensure your lab environment is on-track
Python
Report an issue
6 Function Decorators
0/1
Back
Start check