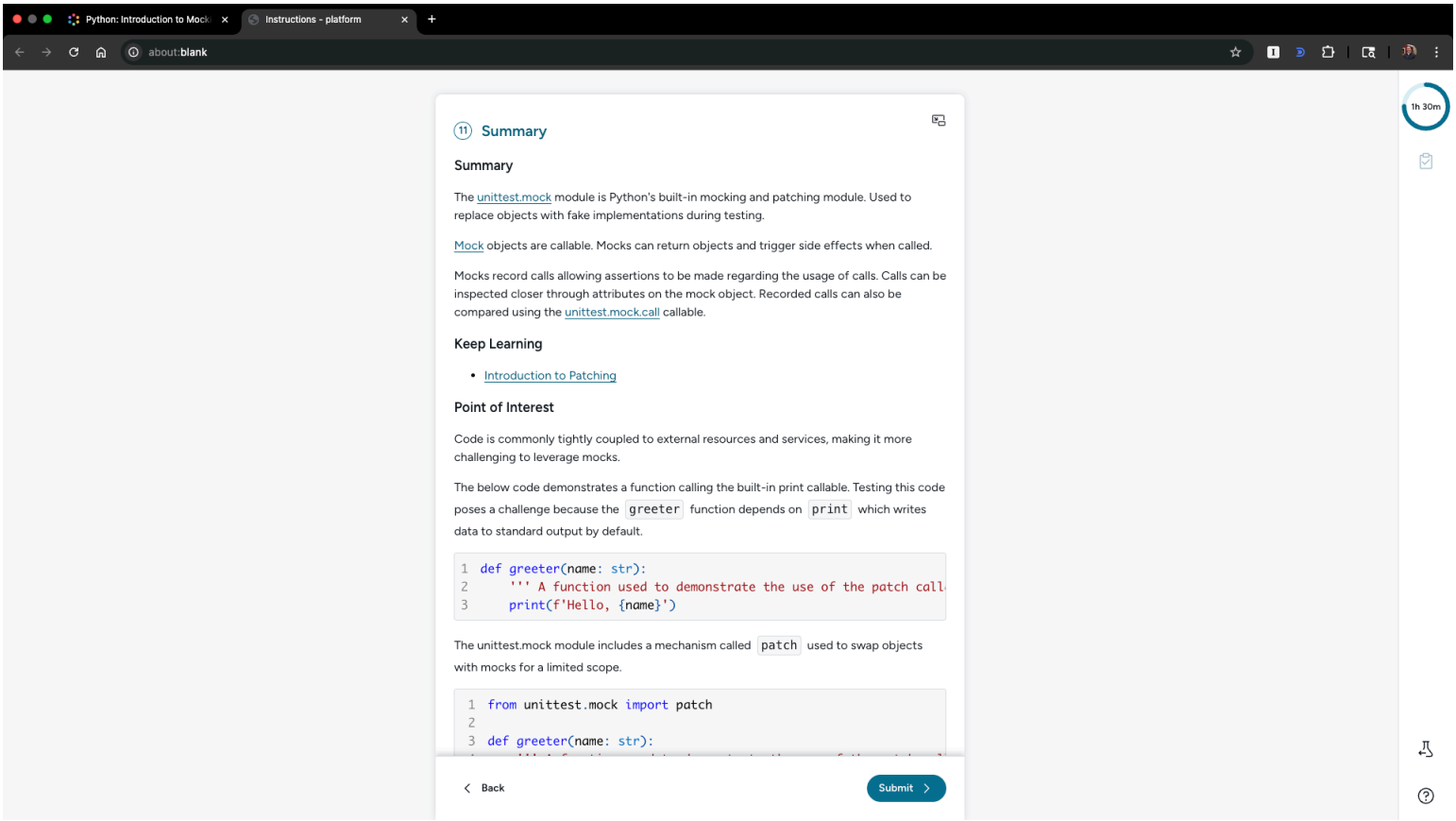
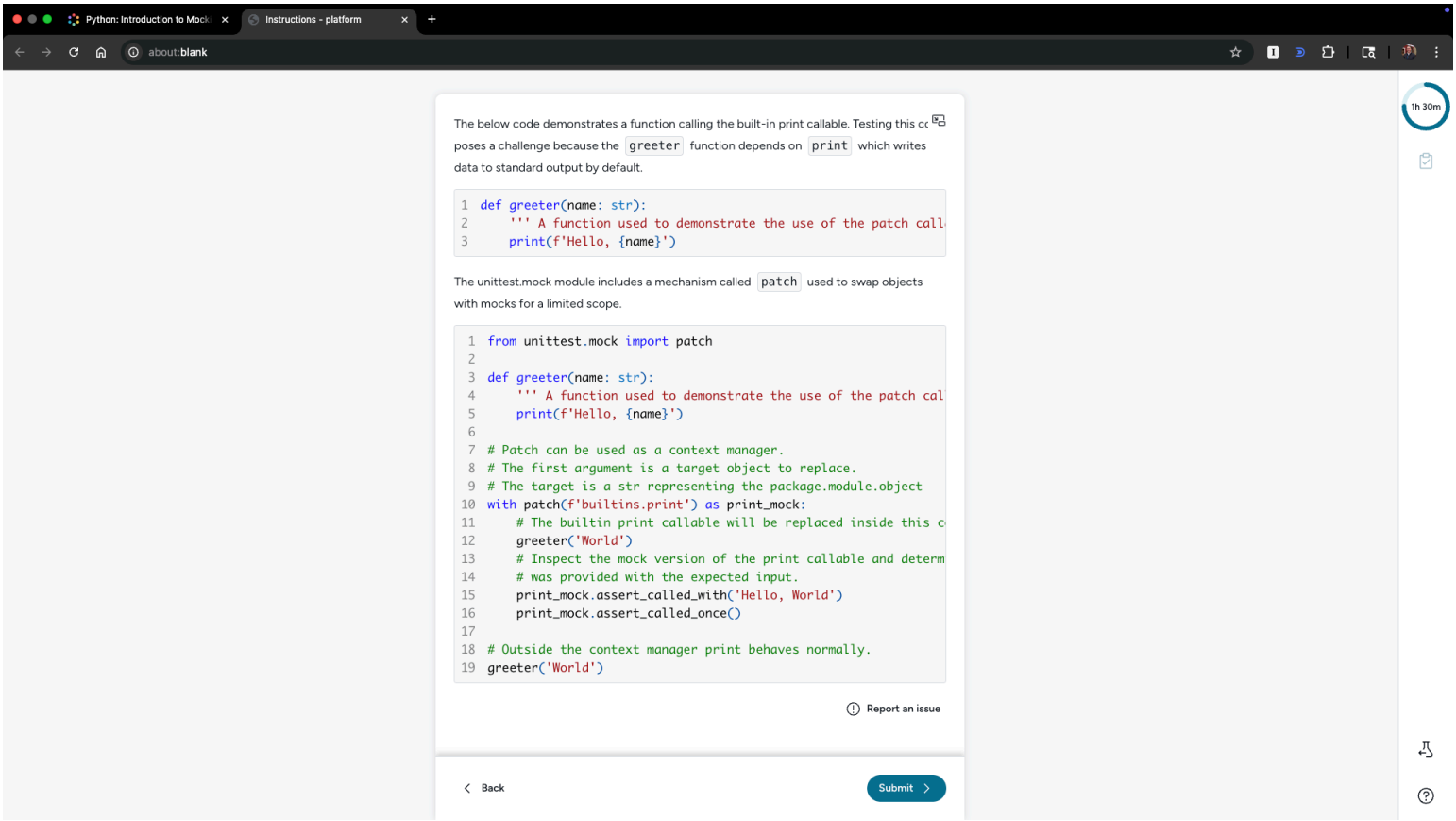

Python: Introduction to Mock
Instructions - platform
about:blank
1h 30m
11 Summary
Summary
Summary
The unittest.mock module is Python's built-in mocking and patching module. Used to replace objects with fake implementations during testing.
Mock objects are callable. Mocks can return objects and trigger side effects when called.
Mocks record calls allowing assertions to be made regarding the usage of calls. Calls can be inspected closer through attributes on the mock object. Recorded calls can also be compared using the unittest.mock.call callable.
Keep Learning
Introduction to Patching
Point of Interest
Code is commonly tightly coupled to external resources and services, making it more challenging to leverage mocks.
The below code demonstrates a function calling the built-in print callable. Testing this code poses a challenge because the greeter function depends on print which writes data to standard output by default.
1 def greeter(name: str):
2 ''' A function used to demonstrate the use of the patch call
3 print(f'Hello, {name}')
The unittest.mock module includes a mechanism called patch used to swap objects with mocks for a limited scope.
1 from unittest.mock import patch
2
3 def greeter(name: str):
Back
Submit


Python: Introduction to Mock
Instructions - platform
about:blank
The below code demonstrates a function calling the built-in print callable. Testing this cc
poses a challenge because the greeter function depends on print which writes
data to standard output by default.
1 def greeter(name: str):
2 ''' A function used to demonstrate the use of the patch call
3 print(f'Hello, {name}')
The unittest.mock module includes a mechanism called patch used to swap objects
with mocks for a limited scope.
1 from unittest.mock import patch
2
3 def greeter(name: str):
4 ''' A function used to demonstrate the use of the patch cal
5 print(f'Hello, {name}')
6
7 # Patch can be used as a context manager.
8 # The first argument is a target object to replace.
9 # The target is a str representing the package.module.object
10 with patch(f'builtins.print') as print_mock:
11 # The builtin print callable will be replaced inside this c
12 greeter('World')
13 # Inspect the mock version of the print callable and determ
14 # was provided with the expected input.
15 print_mock.assert_called_with('Hello, World')
16 print_mock.assert_called_once()
17
18 # Outside the context manager print behaves normally.
19 greeter('World')
Report an issue
Back
Submit
1h 30m