# Report for Assignment 1

## Project chosen

Name: Pyinstaller

URL: https://github.com/pyinstaller/pyinstaller.git

Number of lines of code and the tool used to count it: 60,825 (used lizard to count)

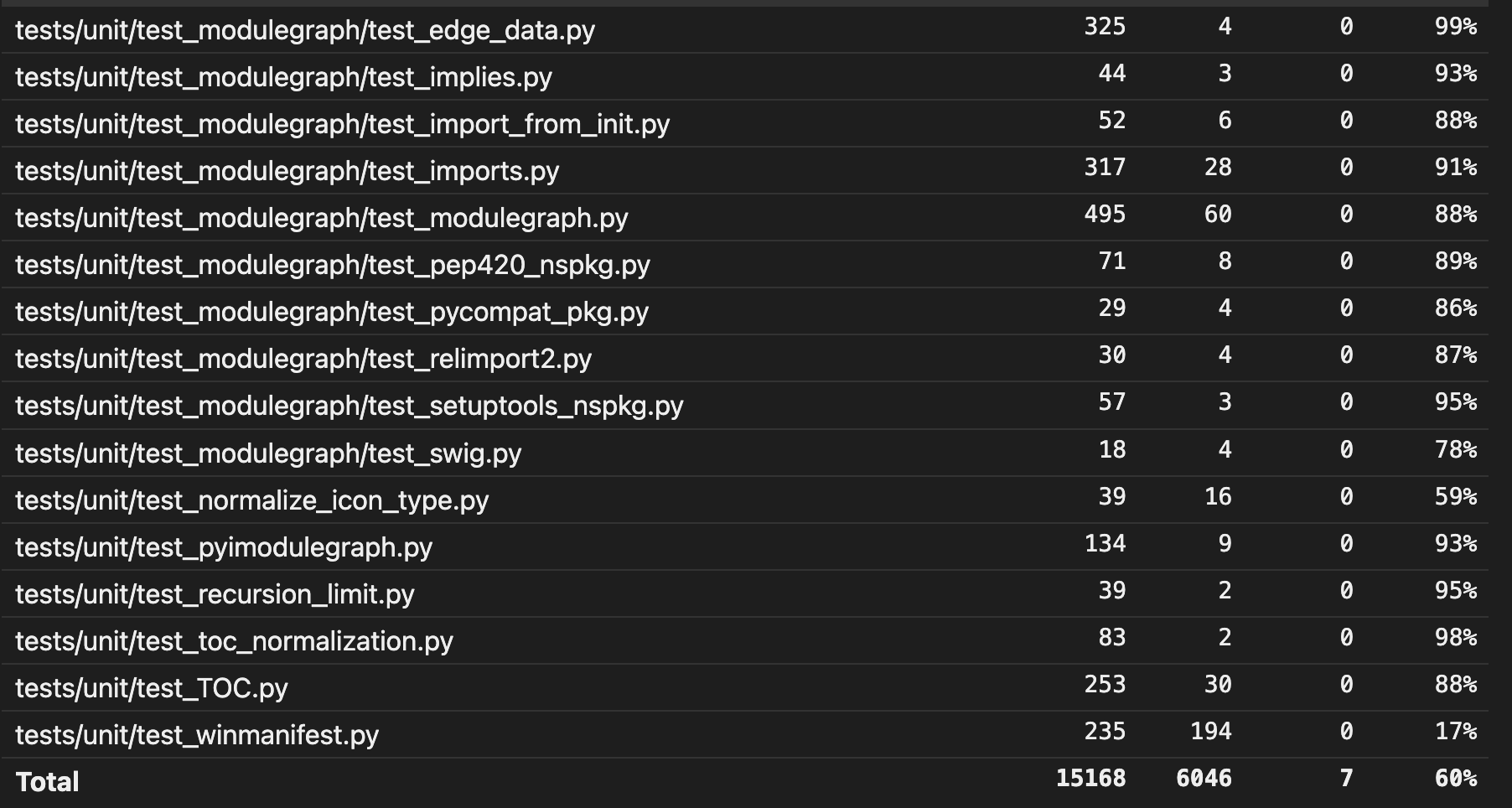
Programming language: Python

## Coverage measurement

### Existing tool

After downloading the necessary modules from the requirements.txt file using “pip install -r requirements.txt”, we used the Coverage.py command “coverage run --source=. -m pytest” to test the coverage of our code. We then translated the results into HTML to help us better visualise the coverage using “coverage html”:



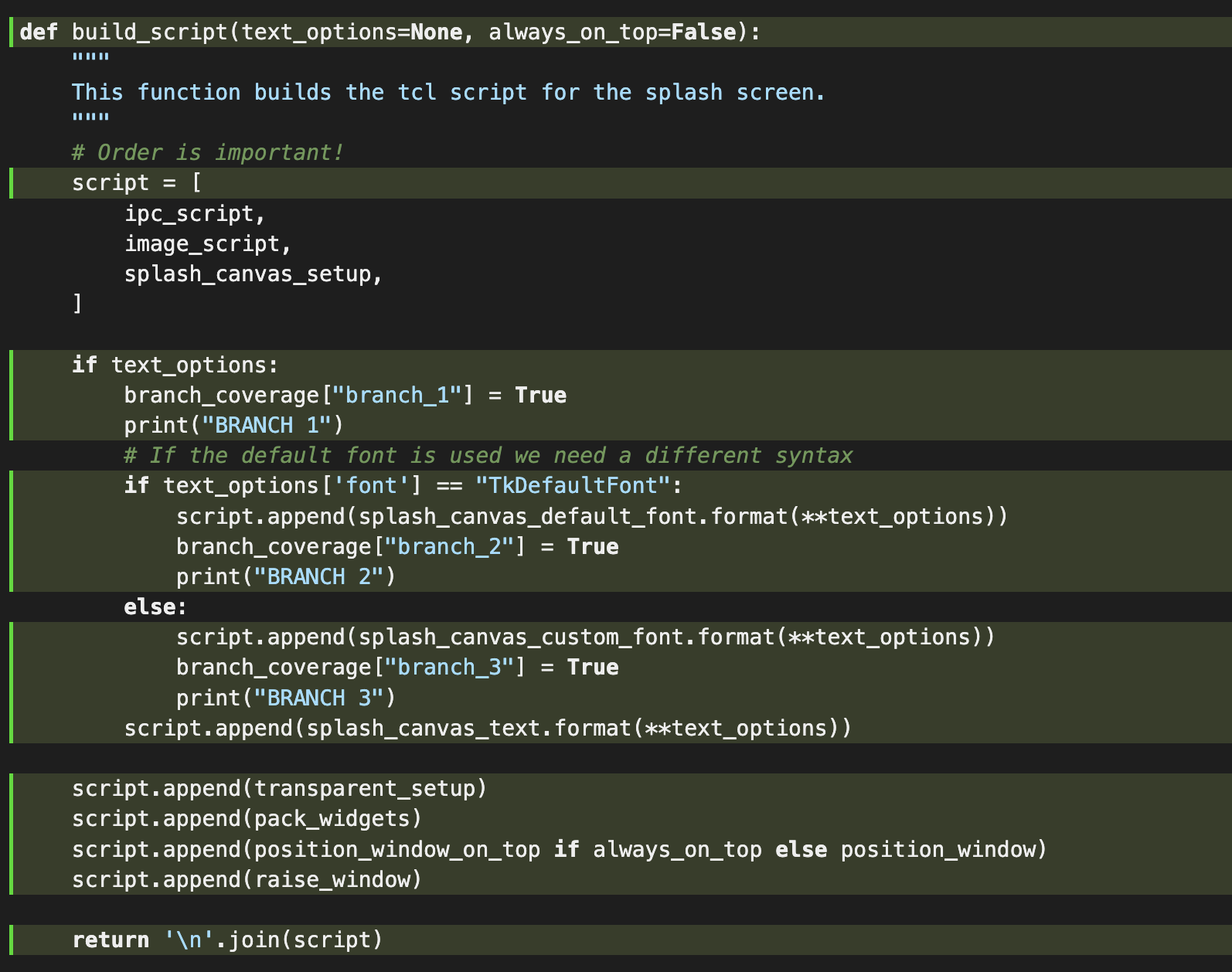


### Your own coverage tool

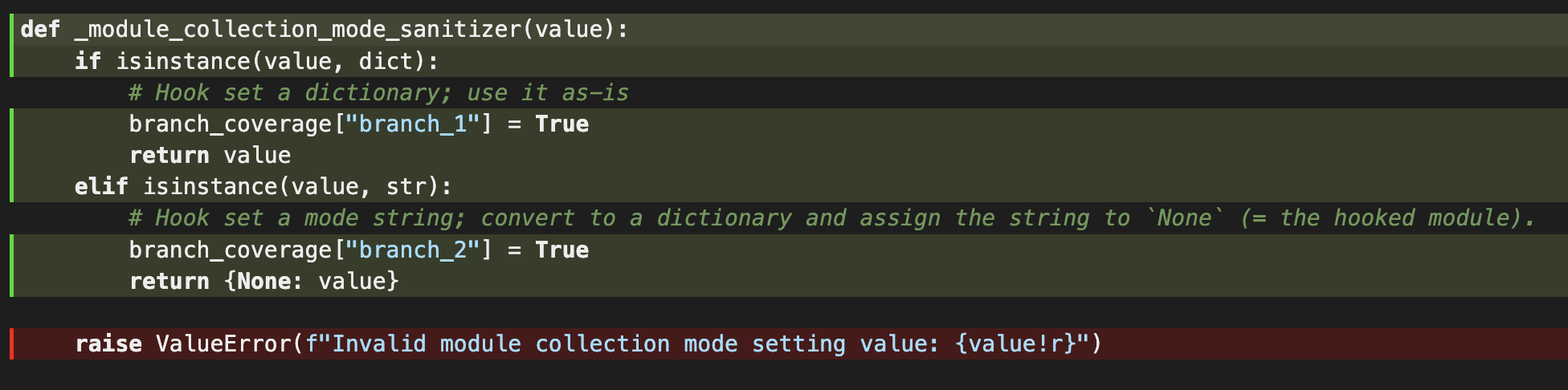
Future (Essam) Abdel Fattah:

Function 1 name: build\_script(text\_options=None, always\_on\_top=False)

<Show a patch (diff) or a link to a commit made in your forked repository that shows the instrumented code to gather coverage measurements>

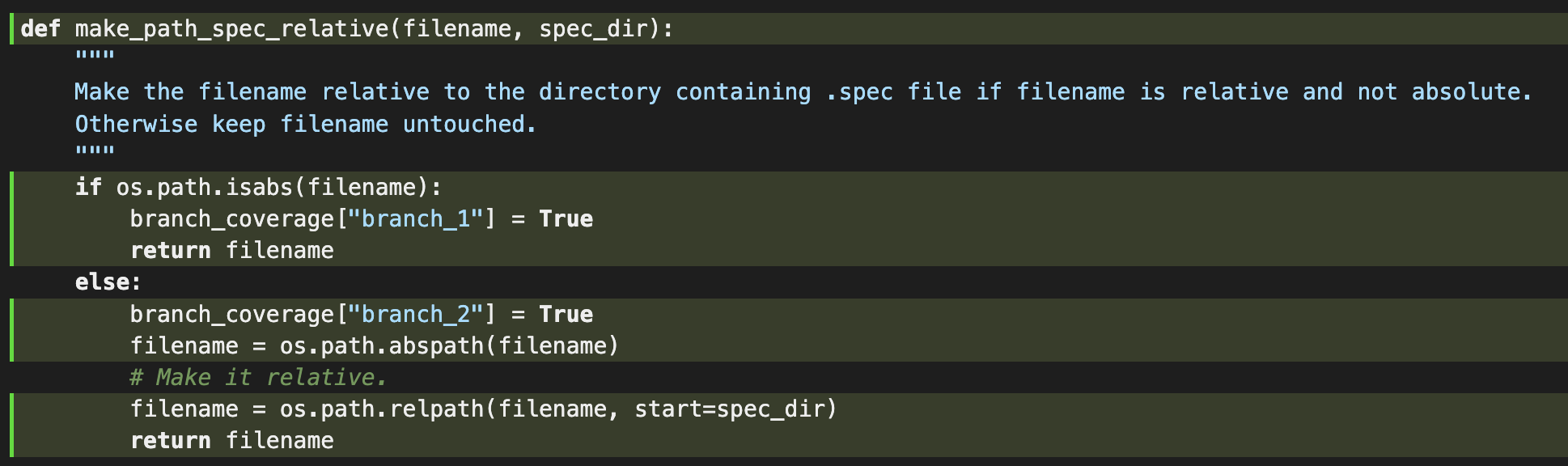


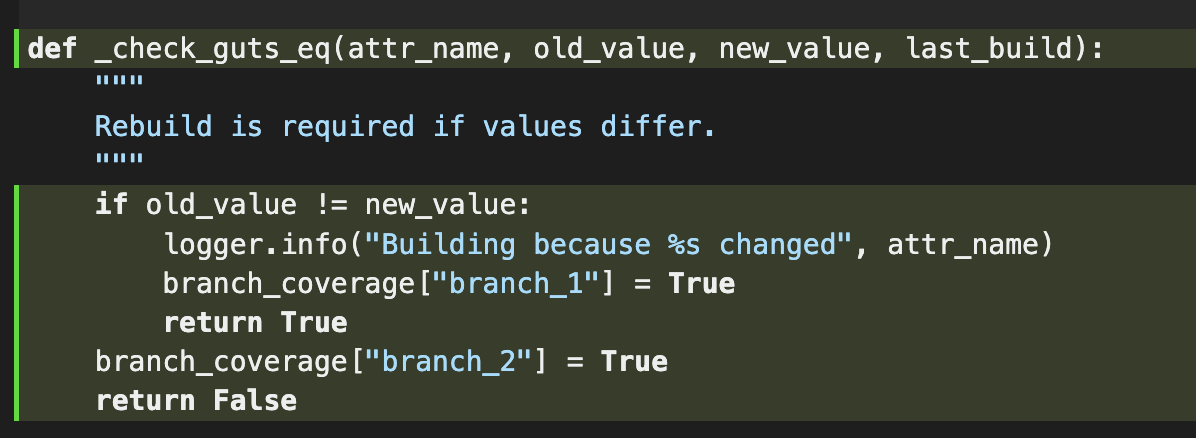
Function 2 name: \_module\_collection\_mode\_sanitizer(value)



Member 2 Seleem Wagdy:

Function 1 name: make\_path\_spec\_relative(filename, spec\_dir)



Function 2 name: \_check\_guts\_eq(attr\_name, old\_value, new\_value, last\_build)

## Coverage improvement

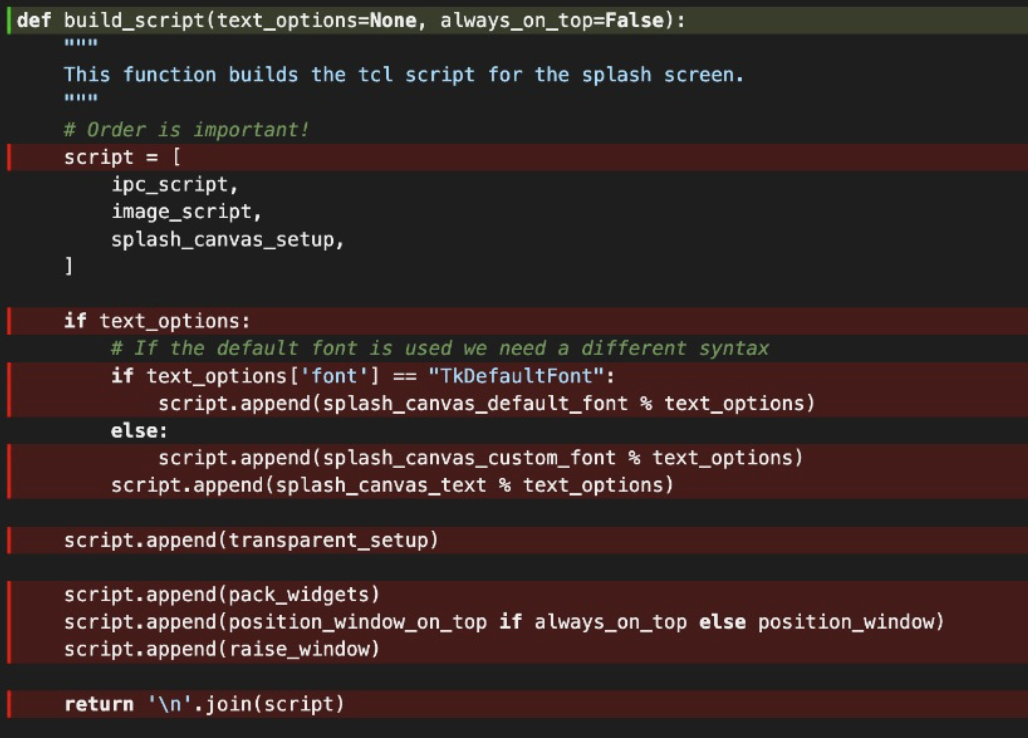
### Individual tests

Member 1 Future (Essam) Abdel Fattah:

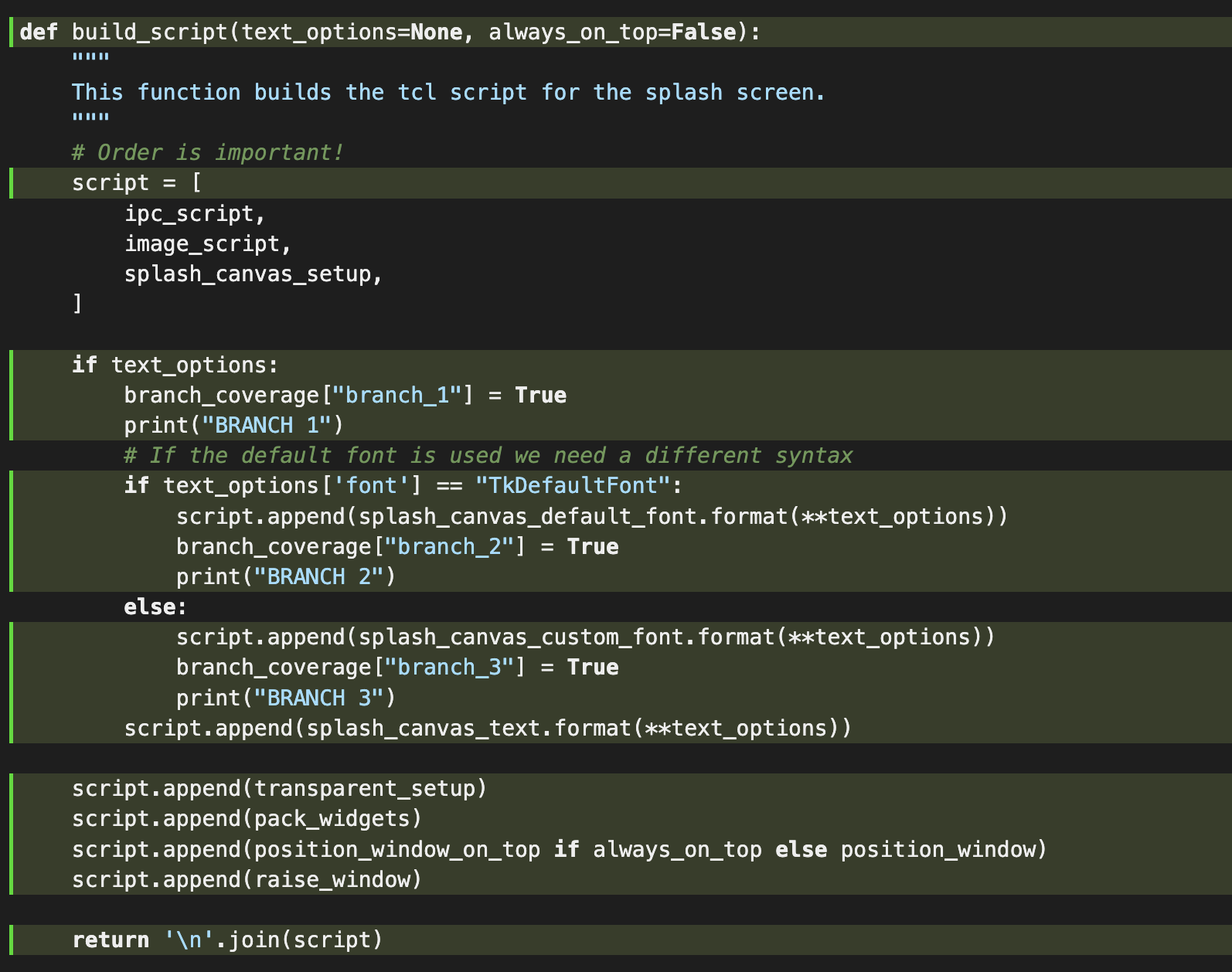
Test 1:

<Show a patch (diff) or a link to a commit made in your forked repository that shows the new/enhanced test>

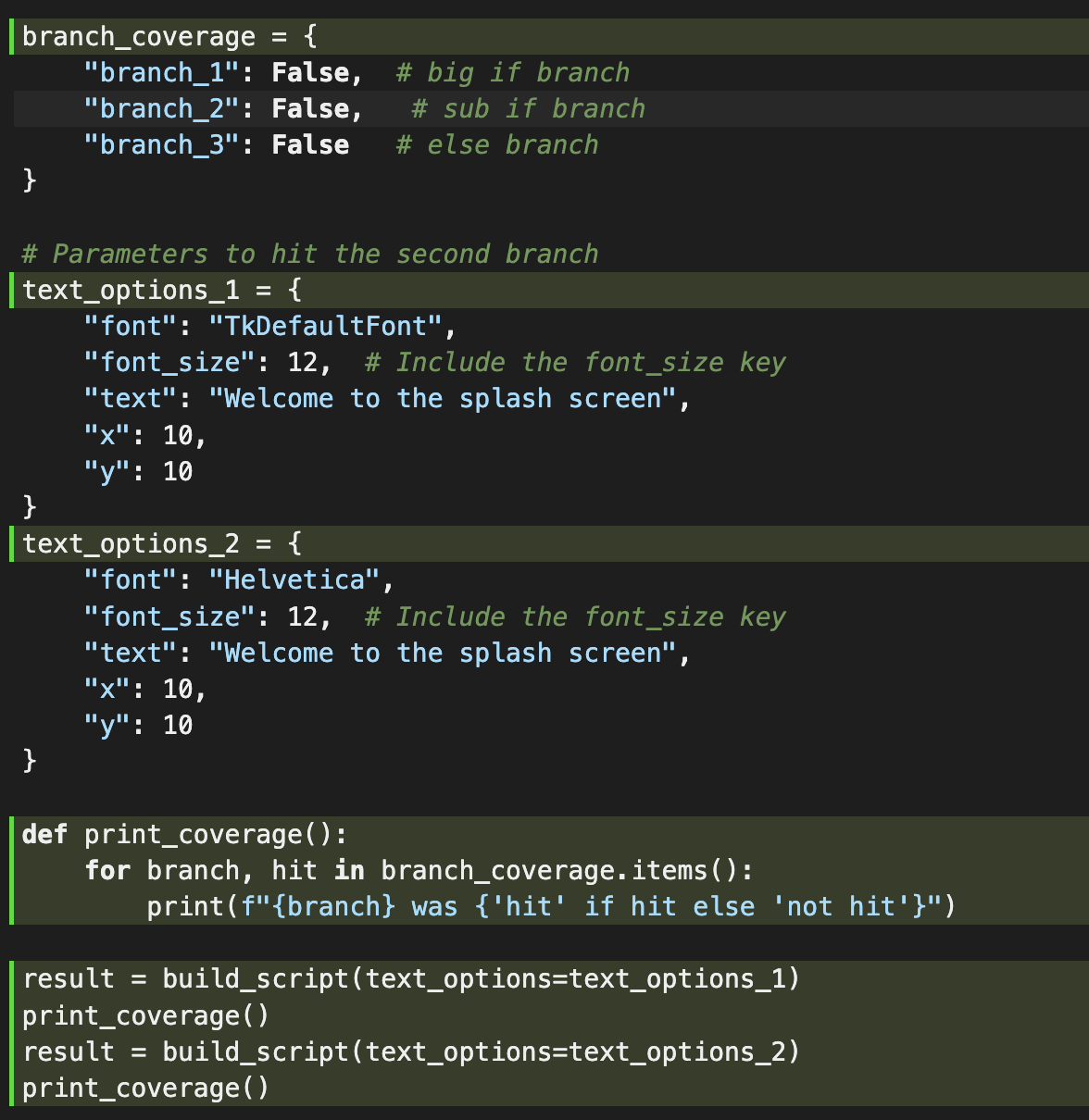
Old coverage results:



New coverage results:

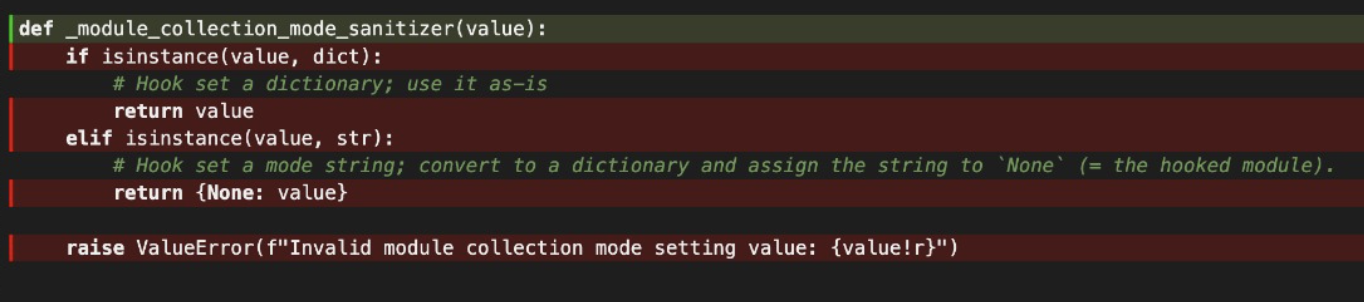


We created a test case where the font is “TKDefault” as well as a test case where the font is “Helvetica”, this made sure that the code in the conditional statements is covered. We can see this is the case because of the green highlight seen in the screenshot above.

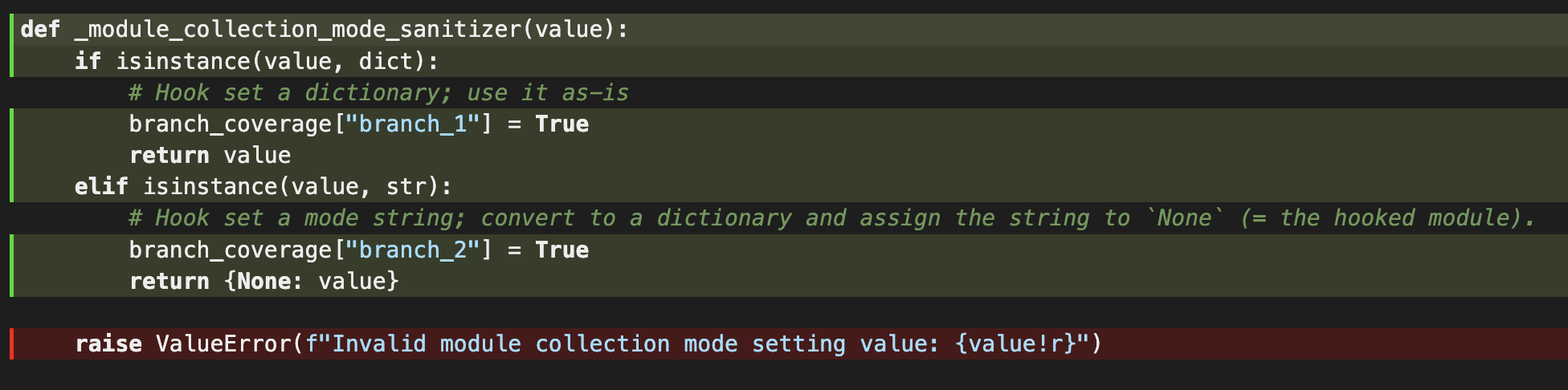


Test 2:

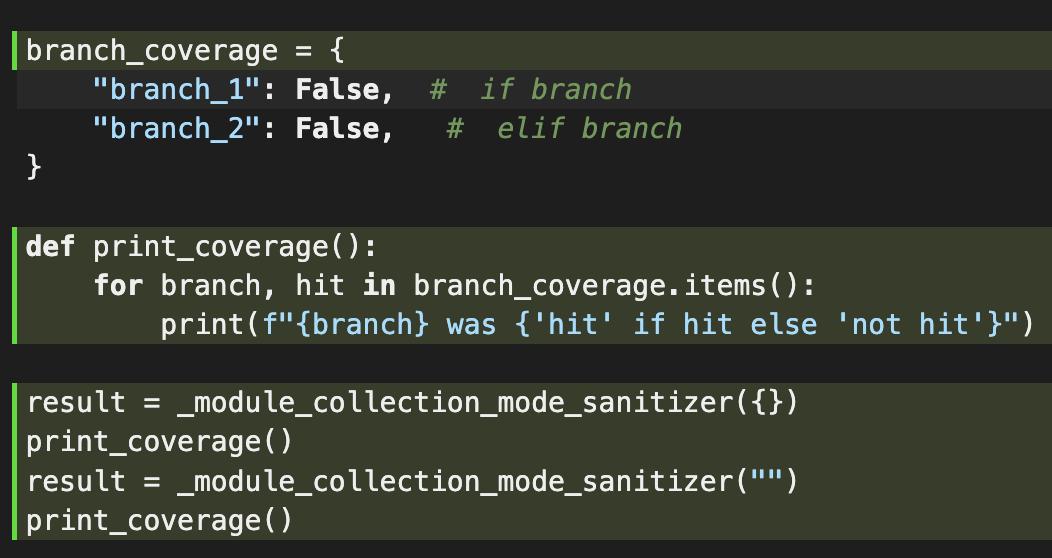
Old coverage results:



New coverage results:



We created 2 test cases to cover both of the branches seen in the code, the first test case tests when value is a dictionary variable, and the second test case tests when value is a string variable. This properly covers both available branches of the function.

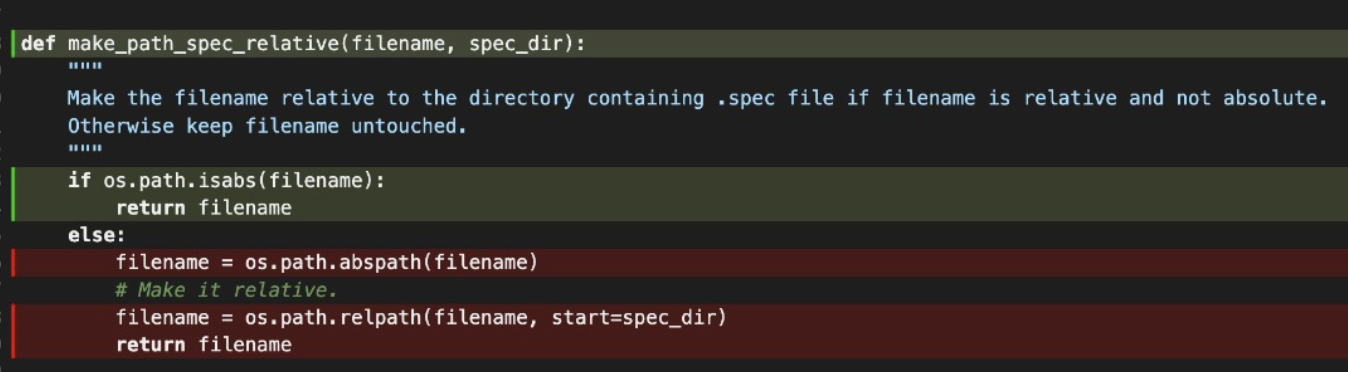


Member 2: Seleem Wagdy

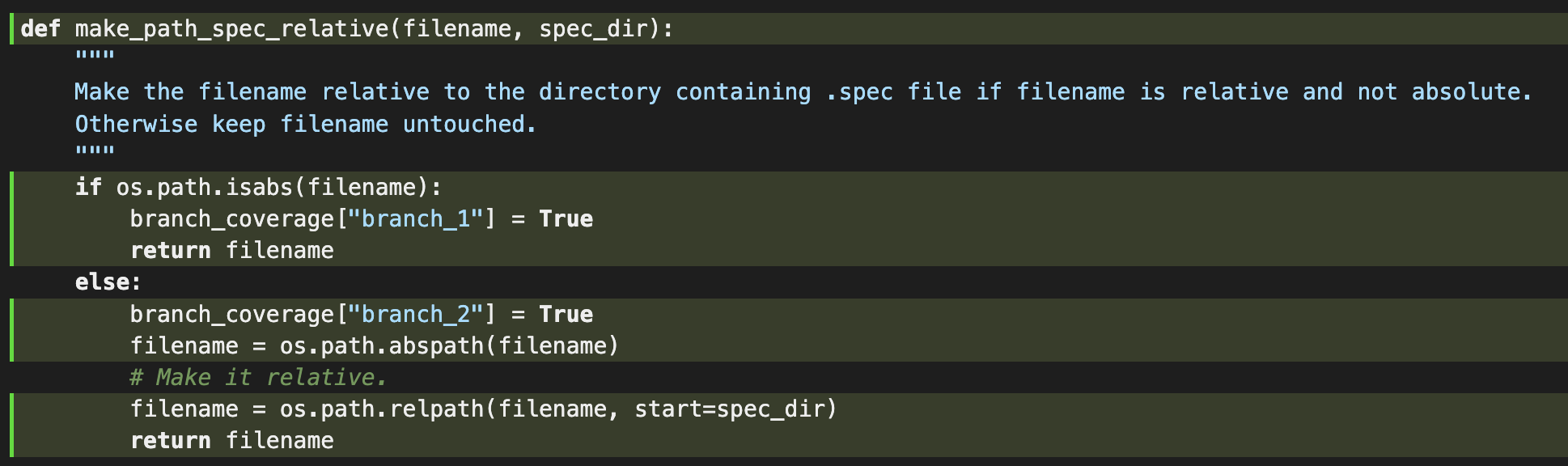
Test 1:

<Show a patch (diff) or a link to a commit made in your forked repository that shows the new/enhanced test>

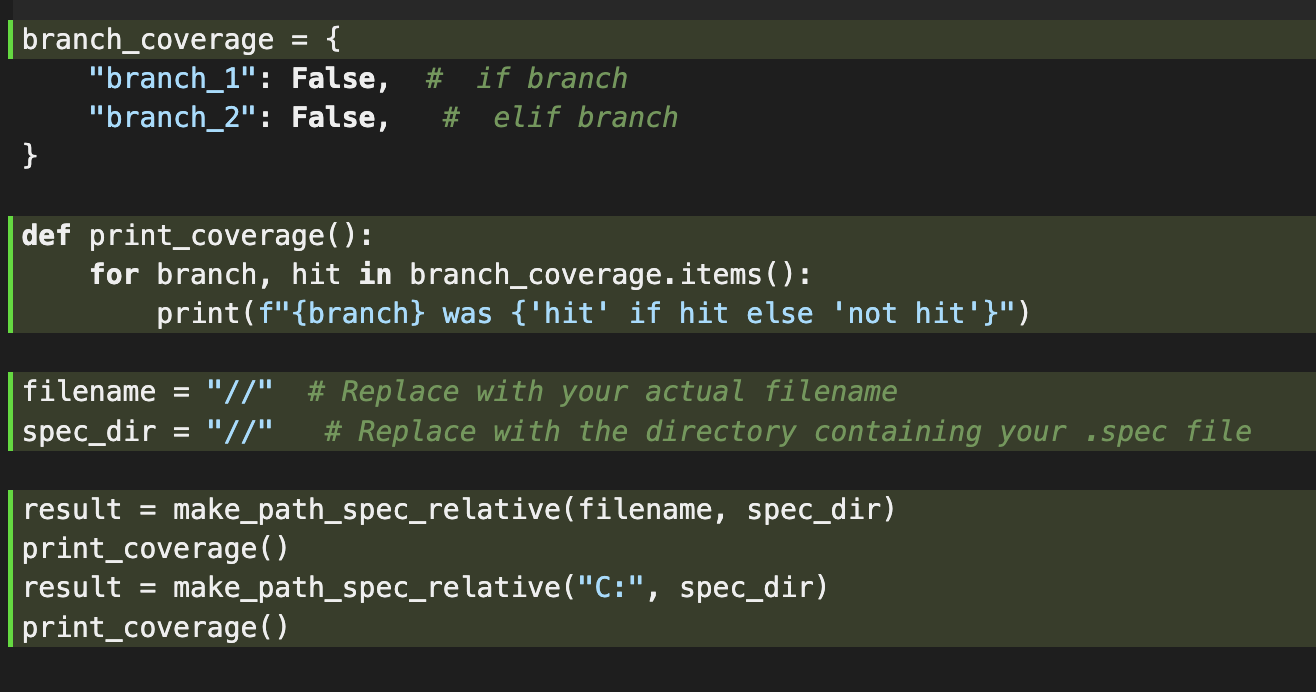
Old coverage:



New coverage:



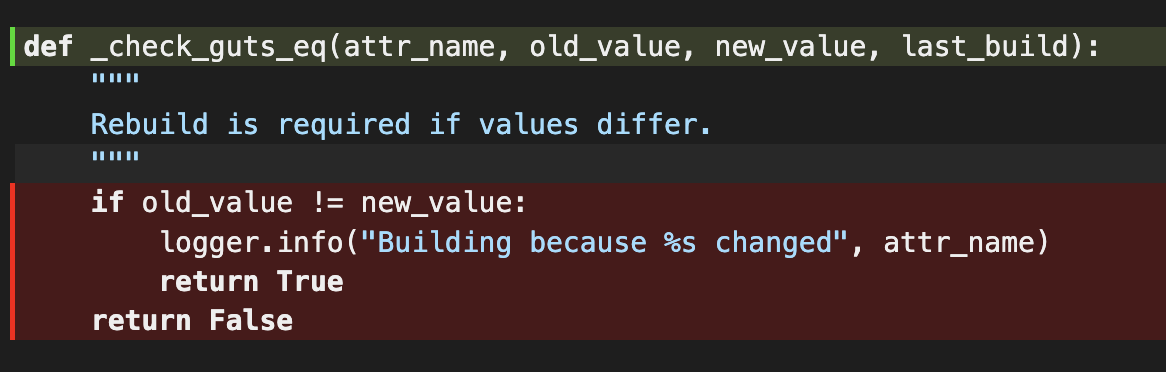
For this function, we created two test cases that would cover the available branches. The first case takes filename to be an absolute path (“//”), while the second test case takes to be a normal path (“C:”).



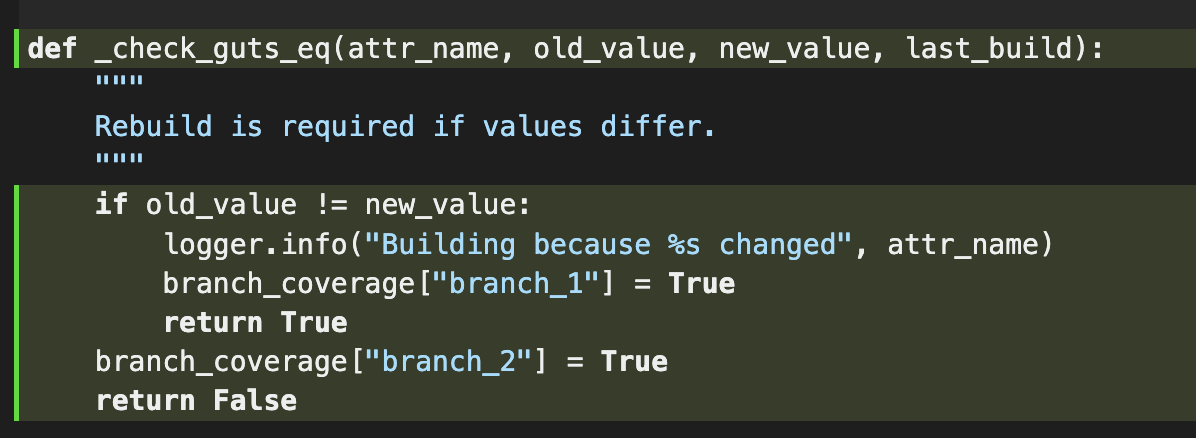
Task 2:

<Show a patch (diff) or a link to a commit made in your forked repository that shows the new/enhanced test>

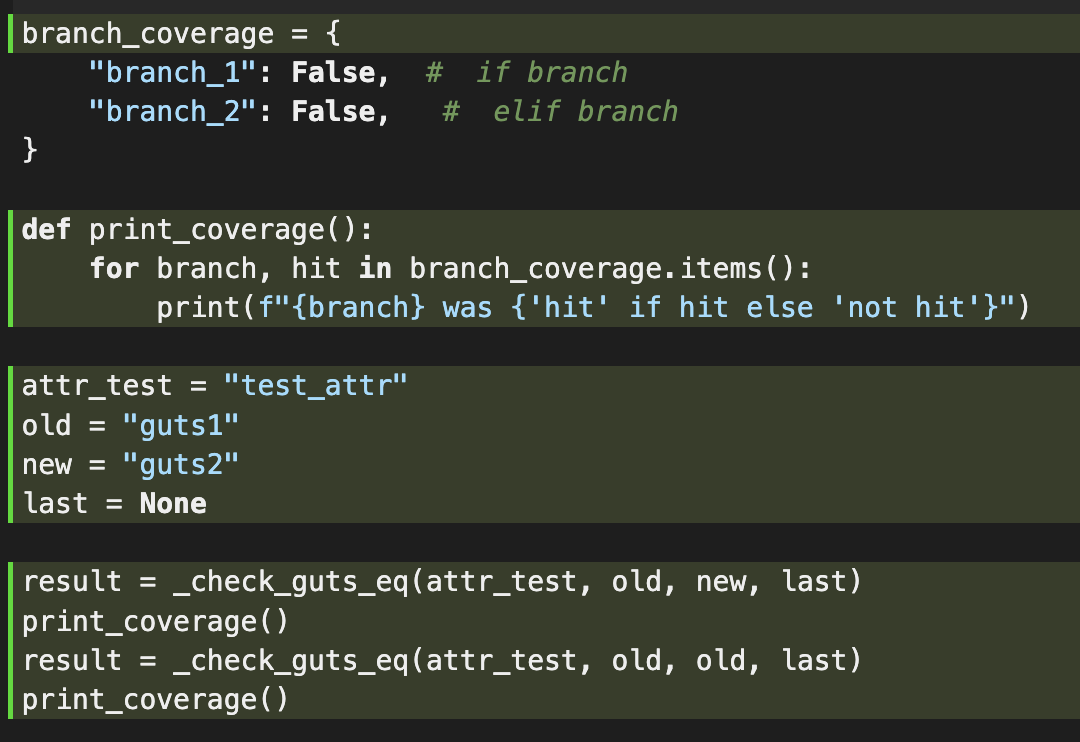
Old coverage:



New coverage:



We created two test cases to cover both branches present in this function, first we created a test cases where old\_value and new\_value are different, and a test case where they are the same.



### Overall



<Provide a screenshot of the new coverage results by running the existing tool using all test modifications made by the group>

## Statement of individual contributions

Looked for a suitable repository: Future and Seleem

Figured out how to run the coverage test: Future and Seleem

Took care of the forked repository: Seleem

Wrote and put together the documentation: Future

Wrote the coverage code for the build\_script and \_module\_collection\_mode\_sanitizer functions: Future

Wrote the coverage code for the make\_path\_spec\_relative and

\_check\_guts\_eq functions: Seleem