Hands-on Lab: Generating Steps with Behave

Estimated time needed: 15 minutes

Welcome to the **Generating Steps with Behave** lab. The first task is to create the steps to satisfy the statements in the `feature` file. To support this task, the **Behave** tool generates function stubs that serve as the skeleton code for you to begin your work.

In this lab, you will learn how to use the `behave` command to generate starter code for your steps files.

Learning Objectives

After completing this lab, you will be able to:

- Use the behave tool to generate recommendations for steps
- Take these recommendations and create a starter steps file from them.

About Theia

Theia is an open-source IDE (Integrated Development Environment) that can be run on a desktop or the cloud. You will use the Theia IDE to do this lab. When you log into the Theia environment, you will be presented with a dedicated computer on the cloud exclusively. This will be available as long as you work on the labs. Once you log off, this dedicated computer on the cloud and any files you may have created are deleted. So, it is a good idea to complete your labs in a single session. If you finish part of the lab and return to the Theia lab later, you may have to start from the beginning. Plan to work out all your Theia labs when you have the time so that you complete a lab in a single session.

Set Up the Lab Environment

You need to prepare the environment before you start the lab. First, open a terminal and install some system and Python dependencies.

Open a Terminal

Open a terminal window using the editor's menu: Terminal > New Terminal.

In the terminal, if you are not already in the `/home/projects` folder, change to your project folder now.

```
cd /home/project
```

Clone the Code Repo

Now, get the code that you need to test. To do this, use the `git clone` command to clone the git repository:

```
git clone https://github.com/ibm-developer-skills-network/duwjx-tdd_bdd_PracticeCode.git
```

Change into the Repo Folder

Then, you need to switch to the Repo directory that contains the lab files.

```
cd /home/project/duwjx-tdd_bdd_PracticeCode
```

Install Lab Dependencies

Once you have cloned the repository, you need to install some prerequisite software into the development environment.

```
bash ./bin/setup.sh
```

Change into the Lab Folder

Next, you need to switch to the lab directory that contains the lab files.

```
cd /home/project/duwjx-tdd_bdd_PracticeCode/labs/11_generating_steps
```

Install Python Dependencies

The final preparation step is to use `pip` to install the Python packages needed for the lab:

```
pip install -r requirements.txt
```

You are now ready to start the lab.

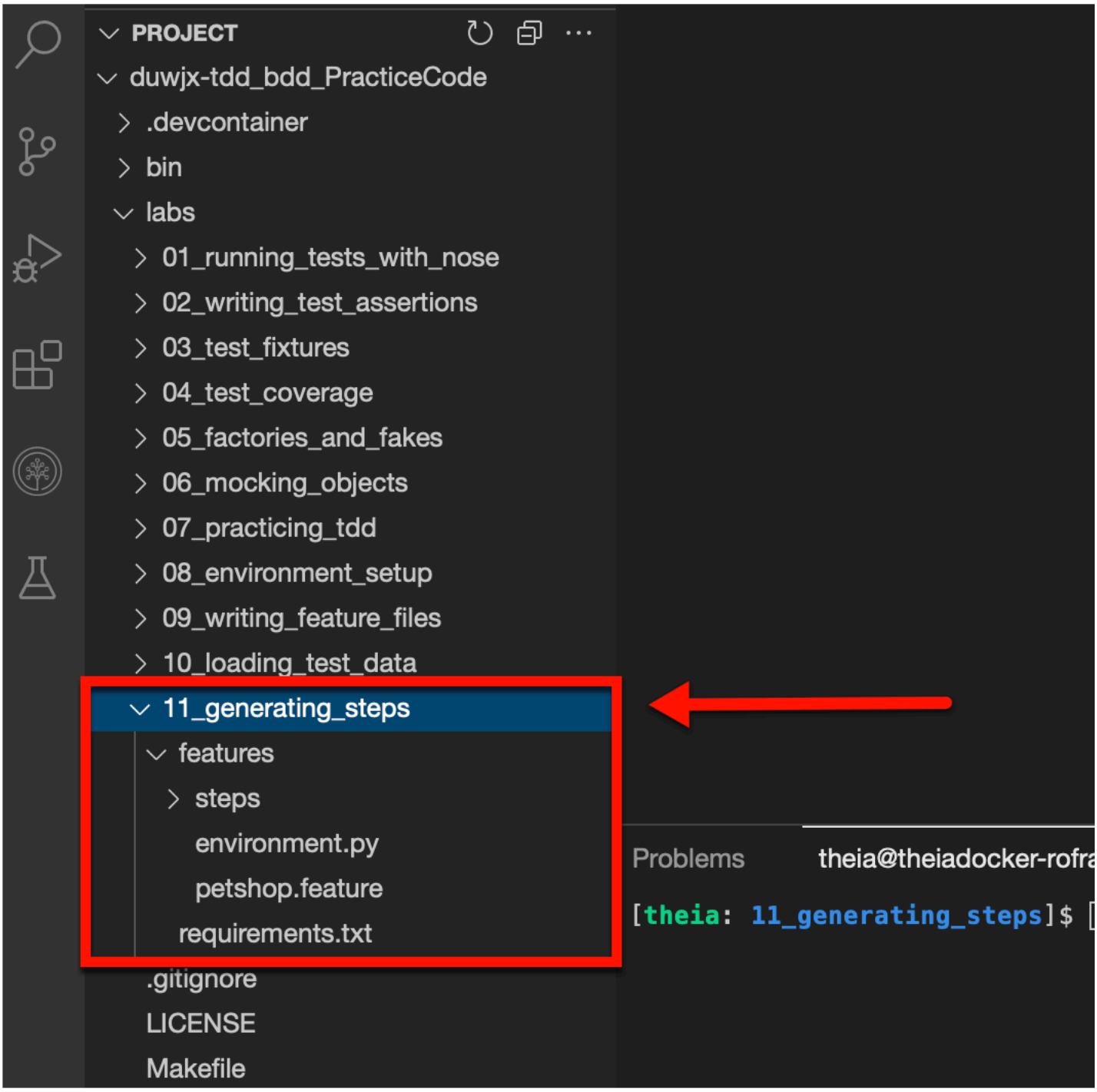
Optional

If working on the terminal becomes difficult because of the length of the command prompt, you can shorten the prompt using the following command:

```
export PS1="[\[\033[01;32m\]\u\[\033[00m\]: \[\033[01;34m\]\W\[\033[00m\]]\$ "
```

Navigate to the Code

In the IDE, navigate to the `duwjx-tdd_bdd_PracticeCode/labs/11_generating_steps` folder. This folder contains all the source code required for this lab.



Step 1: Generate Step Recommendations

When you first run the `behave` command, it will look in the `./features` folder for files with the extension `.feature` and try to find steps in the Python files in the `./features/steps` folder to match them. If any steps are missing, it will recommend the step definitions that need to be created.

When you first create the feature file, most of the steps are missing. You can generate the function definition stubs for your initial steps.

Your Task

Run the `behave` command. Then edit the `./features/steps/web_steps.py` file and paste the recommendations into the file along with some required imports, and save it.

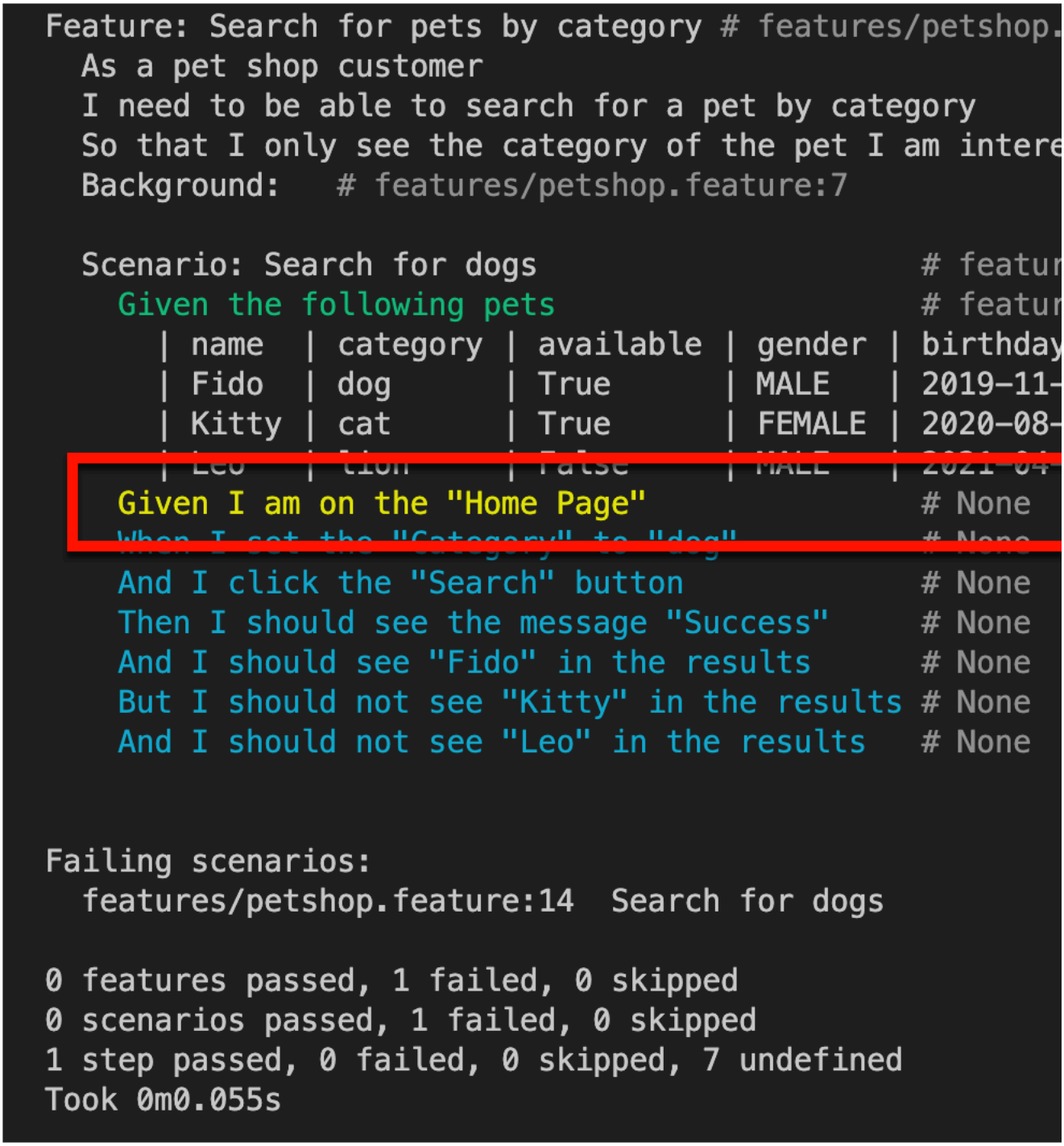
Open the `features/steps/web_steps.py` file in the IDE editor. You will work in this file for the remainder of the lab.

Open **web_steps.py** in IDE

1. Open a Terminal shell and run the `behave` command to see what's missing:

```
behave
```

Notice that the first line after the data table is yellow because it could not find a step that matched.



2. Copy the code definitions in yellow at the bottom of the output and paste them into the `web_steps.py` file:

Note: You may have to right-click inside the Terminal and select the **Copy** command to copy the steps.

```
Failing scenarios:
  features/petshop.feature:14  Search for dogs

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
1 step passed, 0 failed, 0 skipped, 7 undefined
Took 0m0.066s

You can implement step definitions for undefined steps
```

```
@given(u'I am on the "Home Page"')
def step_impl(context):
    raise NotImplementedError(u'STEP: Given I am on the


@when(u'I set the "Category" to "dog"')
def step_impl(context):
    raise NotImplementedError(u'STEP: When I set the "C


@when(u'I click the "Search" button')
def step_impl(context):
    raise NotImplementedError(u'STEP: When I click the


@then(u'I should see the message "Success"')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should see


@then(u'I should see "Fido" in the results')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should see


@then(u'I should not see "Kitty" in the results')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should not


@then(u'I should not see "Leo" in the results')
def step_impl(context):
```

3. Add the following imports to the top of the `web_steps.py` file that will be needed to resolve dependencies.

```
from behave import given, when, then
from selenium.webdriver.common.by import By
```

4. Save the file.

Solution

You can check below that you created the file correctly.

▼ Click here for the solution.

This is the solution for creating the initial steps:

```
from behave import given, when, then
from selenium.webdriver.common.by import By
@given(u'I am on the "Home Page"')
def step_impl(context):
    raise NotImplementedError(u'STEP: Given I am on the "Home Page"')
@when(u'I set the "Category" to "dog"')
def step_impl(context):
    raise NotImplementedError(u'STEP: When I set the "Category" to "dog"')
@when(u'I click the "Search" button')
def step_impl(context):
    raise NotImplementedError(u'STEP: When I click the "Search" button')
@then(u'I should see the message "Success"')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should see the message "Success"')
@then(u'I should see "Fido" in the results')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should see "Fido" in the results')
@then(u'I should not see "Kitty" in the results')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should not see "Kitty" in the results')
@then(u'I should not see "Leo" in the results')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should not see "Leo" in the results')
```

Step 2: Run Behave

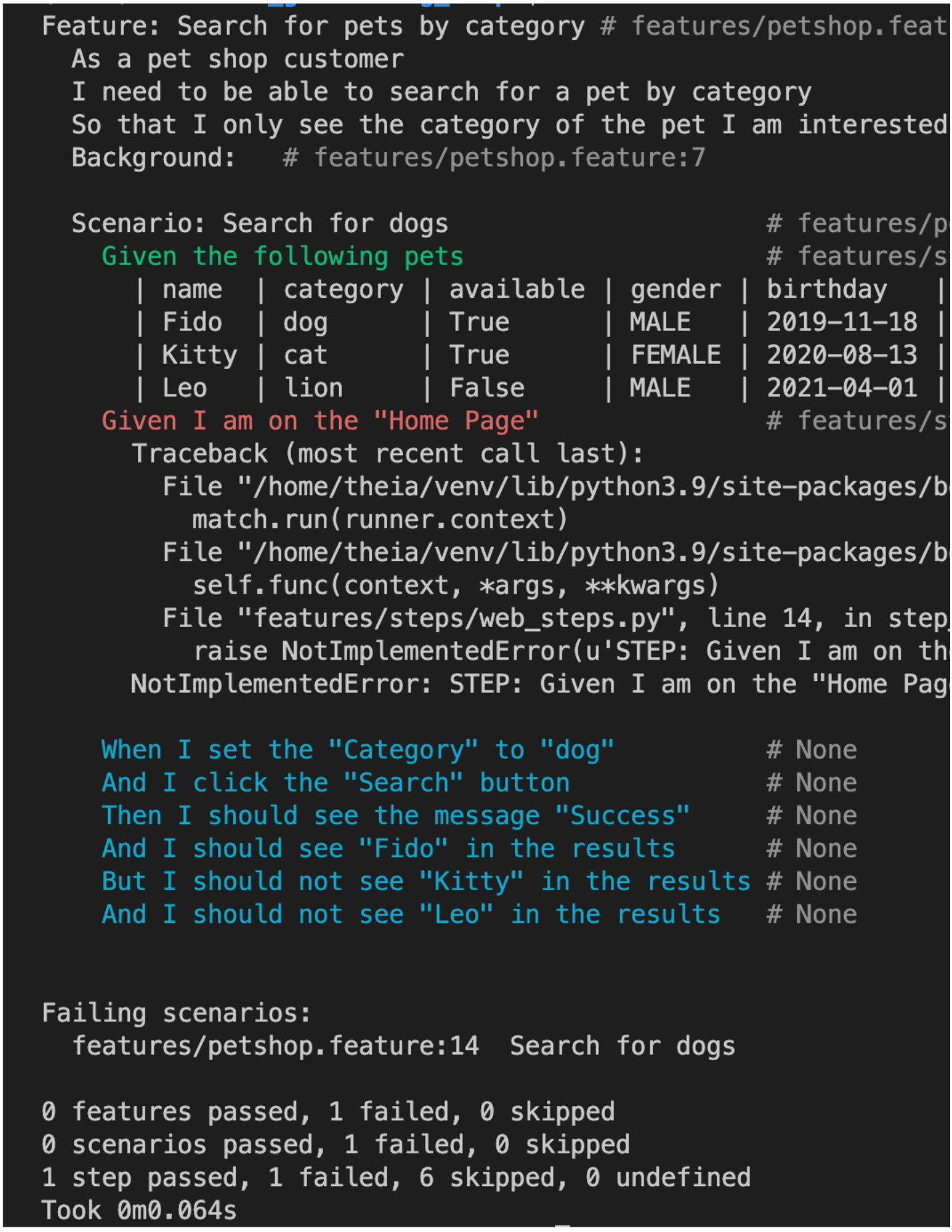
You are now ready to run the `behave` command to see if it finds the web steps you created.

Your Task

1. Run the `behave` command in a terminal shell

```
behave
```

The output should look like the image below. Notice the first statement is now red, signifying that it found the step, but the step failed because it raised a `NotImplementedError` which is no surprise. Since the steps have not been implemented, they will all give this error until you add the actual code to perform the step.



You are now ready to proceed to the next lab.

Conclusion

Congratulations! You just completed the **Generating Steps with Behave** lab. You should now have the skills to create the initial steps to implement the tests for a BDD `feature` file. You can use the `behave` tool to generate function stubs, which you can fill in with the real code.

In the next lab, you will learn how to write the steps.

Author(s)

John J. Rofrano

Contributors