Python Fundamentals Capstone

horizontal line

**About the Author**

|  |
| --- |
| Brian Lipp is a full-time Data Scientist, Data Engineer and Software Developer with 5+ years of experience with software development. |

**Lesson Name**: **Capstone**

*Task name :* Unit Testing Rest API’s

*Estimated time:*  ***5 hours***

**Scenario:**

REST API’s are ubiquitous, and python is the go to tool for testing REST APIs. The goal of this capstone is to mimic a real-world scenario of API testing. You must write several pytest unit tests that validates an example website’s endpoints. To guarantee stability over time, several JSON files will be present on a github account to represent the website under test. The student will combine the skills learned in the course with some new skills. The Application will use the requests package and pytest for basic unit testing.

Mock API’s and Schema:

API URL:

<https://raw.githubusercontent.com/TrainingByPackt/PythonFundamentals/master/>Capstone/API/

A. capstone/api/getCustomers/s

Expected schema:

{

"id": X,

"first\_name": "X",

"last\_name": "X",

"email": "X",

"ip\_address": "X",

"address": "X"

}

B. capstone/api/getSites,

Expected schema:

{

"id": X,

"address": "X",

"ThirdParty": "X",

"admin": "X"

}

C. capstone/api/getBillingInfo:

{

"id": 1,

"FirstName": "X",

"LastName": "X",

"city": "X",

"state": "X",

"Lang": "X",

"SSN": "X"

}

Project Scaffolding:

MyApplication

├── README.md

├── requirements.txt

├── functions

│ └── AccessApi.py

└── tests

└── test\_company\_api.py

The functions and tests folder should have an empty file names \_\_init\_\_.py in each folder.

To run your test you must install and use pytest and run the command:

python -m pytest <test file>

Pro Tips:

1.

Use annotations to give type hints. Python is not a strongly typed language, but with version 3 a developer can give type hints that all IDE’s will use and mimic a strongly typed language. This increases readability, and reduces errors in your code with only minor functionality being given up.

Examples:

def function(arg: <type>) -> <return\_type>:

my\_variable: <type> = <value>

2.

Docstrings are a very helpful tool in increasing readability of code, which can come up after its been put down for some time. When possible always try to add useful information, but do not over document.

3.

**Take Home Assignment**

**Lesson Name:** Supporting Class

**Scenario:** In order to access external REST API’s you must use the requests package to communicate in the HTTP protocol.

Create a class called AccessApi with 4 main components.

1. A constructor that requires the developer to input a base url as a string that will host the REST API endpoint. Example: “http://google.com”
2. A method to get the current URL base.
3. A method to set the current URL base.
4. A method to test that the URL is responding to GET requests to allow for a simple alive test.
5. A method to input an endpoint and have that endpoint concatenated to the base url and then send a GET request using the requests package to the combined string. Then return the JSON sent as a list.
6. A method to input an endpoint and have that endpoint concatenated to the base url and then send a GET request using the requests package to the combined string. Then return the status code.
7. A method to input an endpoint and have that endpoint concatenated to the base url and then send a GET request using the requests package to the combined string. Then return the total elapsed time used for the GET request.

.

**Take Home Assignment**

**Lesson Name:** Testing Process

**Scenario:** Now that the AccessApi class has been created we must created tests for each of the API endpoints.

Create the following 5 tests for each of the api endpoints.

1. valid the HTTP status code is 200

2. valid schema matches the one provided on a very simple level. Determine that the key’s are correct.

3. Validate the accuracy of the data by randomly picking an element for all the data and do a reasonable check. For example a SSN should be all int’s, in a format of XXX-XX-XXXX.

4. validate that the response time should be less than one minute.

Endpoints:

* Billing
* Sites
* Customers

Resource: for now please use: <https://github.com/bclipp/APItesting>

A simple pytest looks like:

def <test\_name>(<args>):

<setup code for test>

assert <an expression that tests and has a True or False outcome>

a simple example is:

def testing\_my\_class(age:int):

assert age <= 50

**Take Home Assignment**

**Lesson Name:** Refactoring our tests

**Scenario:** We now have a testing platform and some useful tests for our companies website. The tests we have done are not maintainable and hard not very pythonic. The next task is to now refactor two of our tests so they use the parametrize functionality of pytest.

Example:

import pytest

@pytest.mark.parametrize("<first\_input>,<second\_input>,", [

("String1", true),

("String2", false),

("String3",true),

])

def test\_eval((<first\_input>,<second\_input>):

….

Tests to Refactor:

1. valid the HTTP status code is 200
2. Validate that the response time should be less than one minute