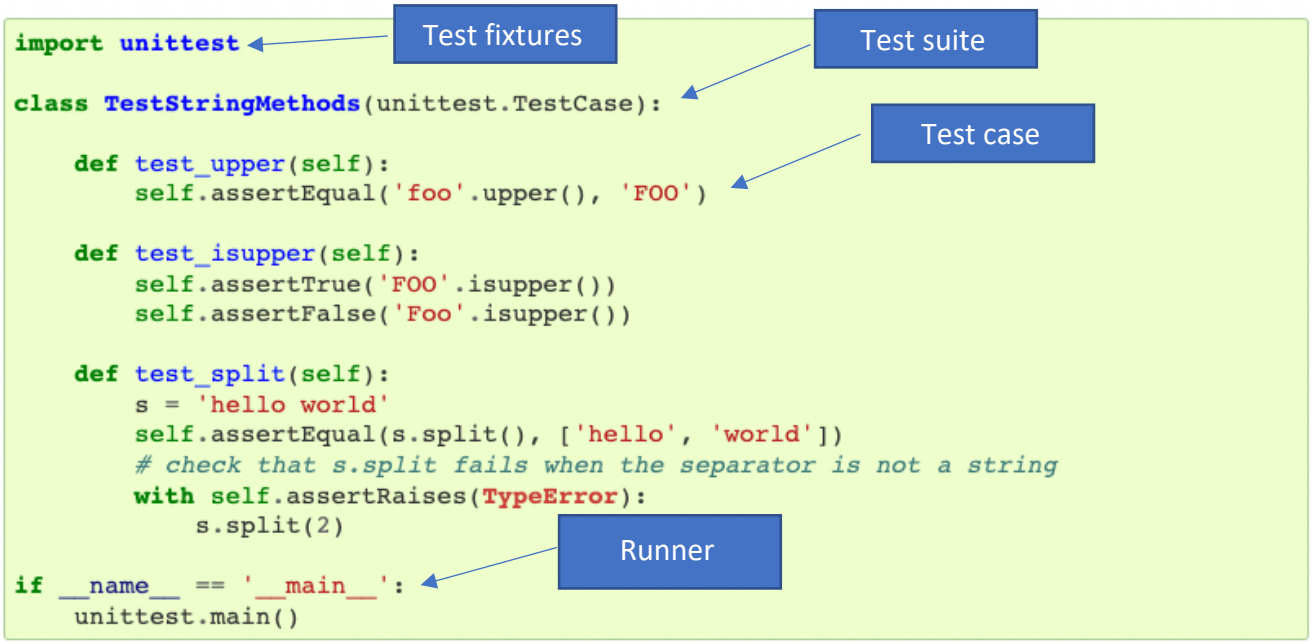
Unit Tests in Python and the Integration of Python Unit Test in CI Pipeline – GitLab

Unit Tests in Python

1. Test module: unittest
    a. Unit testing framework that is inspired by Junit using the same assert methods and similar construction
    b. It supports test automation, sharing of setup, and shutdown code for tests.
    c. Object-oriented
    d. Components:
        i. Test fixtures: preparation necessary to preform and cleanup tests
        ii. Test case: Base class TestCase, where all of the unit tests are contained. Note: unit test methods should be named with "test" at the beginning.
        iii. Test suite: Collection of test cases (each class is a suite, and all of the suites make a suite as well)
        iv. Test runner: execution of the tests




```
import unittest

class TestStringMethods(unittest.TestCase):

    def test_upper(self):
        self.assertEqual('foo'.upper(), 'FOO')

    def test_isupper(self):
        self.assertTrue('FOO'.isupper())
        self.assertFalse('Foo'.isupper())

    def test_split(self):
        s = 'hello world'
        self.assertEqual(s.split(), ['hello', 'world'])
        # check that s.split fails when the separator is not a string
        with self.assertRaises(TypeError):
            s.split(2)

if __name__ == '__main__':
    unittest.main()
```

    e. How to execute:
        i. python <name of test file>
        ii. Note: you can move the runner to a separate file and execute the tests from there. This is helpful if the import hierarchy is not convenient for executing tests from the test folder. This will likely be necessary for this project as the src folder contains a lot of the files necessary for testing, so having the runner in a file in the src folder will make it easier to import the files necessary for testing.

f. Assert methods:

Method	Checks that	New in
`assertEqual(a, b)`	`a == b`	
`assertNotEqual(a, b)`	`a != b`	
`assertTrue(x)`	`bool(x) is True`	
`assertFalse(x)`	`bool(x) is False`	
`assertIs(a, b)`	`a is b`	3.1
`assertIsNot(a, b)`	`a is not b`	3.1
`assertIsNone(x)`	`x is None`	3.1
`assertIsNotNone(x)`	`x is not None`	3.1
`assertIn(a, b)`	`a in b`	3.1
`assertNotIn(a, b)`	`a not in b`	3.1
`assertIsInstance(a, b)`	`isinstance(a, b)`	3.2
`assertNotIsInstance(a, b)`	`not isinstance(a, b)`	3.2

Note: https://docs.python.org/3/library/unittest.html provides more information on unittest, this is just a brief summary.

CI Pipeline in GitLab

1. CI Pipeline: contains jobs that are meant to be executed repetitively. Unit tests are great examples of repetitive tasks/execution, so having a pipeline that can run them all automatically would be beneficial.
2. Creating a pipeline:
    a. Must add a .gitlab-ci.yml file to the repo
    b. Sections of the file:
        i. Default: defines the steps necessary prior to the execution of tests, and docker image
            1. Image: the docker image to pull
            2. Before_script: list of steps needed before execution – installations, directory changes
        ii. Stages: the job execution order
            1. Script: define the commands needed to be executed to run tests
        iii. Final job (can be named anything): the last job executed
    c. Once the file has been added, the default is for the pipeline to be ran every time a commit/push occurs in the repo. This can be updated in the scheduling part of the pipeline.
3. Maintaining the pipeline:
    a. Maintaining the pipeline isn't a difficult task as it is likely that nothing in the pipeline will need to change. Most of the maintenance will need to be done on unit tests, if anything drastic in the code changes.
4. Debugging fails:

a. Most of the fails in the pipeline are due to changes in the code that are not reflected in the unit tests.
b. Sometimes fails are due to the wrong version of a language within the pipeline, etc. Python 3.8 vs. 3.10

Note: https://medium.com/swlh/automate-testing-with-gitlab-pipelines-4d35c72c18a outlines the development of a pipeline using a different form of unit tests, but is still useful for understanding pipelines in GitLab

Other source(s)

https://medium.com/techtofreedom/unit-testing-in-python-23b129add2b
https://docs.gitlab.com/ee/ci/pipelines/