

Intro to Unit Testing and Continuous Integration



Mariatta Wijaya

[@mariatta](#)



Mariatta Wijaya

Python Core Developer
Staff Software Engineer @ Uplight

Vancouver, Canada

she/her UTC - 7



@mariatta

Unit Testing



Unit Testing



tldr; do it

Why Unit Test?

Does the code work?

add_number.py

```
def add_two_numbers(first, second):  
    """Adds up both numbers and return the sum."""  
    return first + second
```

```
>>> print(add_two_numbers(1, 2))  
3
```


Why Unit Test?

To answer: “Does the code work?”

test_add_number.py

```
import unittest
from add_number import add_two_numbers
```

```
class TestAddNumbers(unittest.TestCase):
    def test_add_two_numbers(self):
        result = add_two_numbers(1, 2)
        self.assertEqual(result, 3)
```

```
>>> python -m unittest test_add_number
```

.

Ran 1 test in 0.000s

OK

Why Unit Test?

Test your code against various inputs and scenarios

```
def test_add_two_numbers(self):  
    result = add_two_numbers(1, 2)  
    self.assertEqual(result, 12)
```

```
>>> python -m unittest test_add_number
```

F

=====

FAIL: test_add_two_numbers (test_add_number.TestAddNumbers)

Traceback (most recent call last):

```
File "test_add_number.py", line 9, in test_add_two_numbers  
    self.assertEqual(result, 12)
```

AssertionError: 3 != 12

Ran 1 test in 0.000s

FAILED (failures=1)

Why Unit Test?

Test your code against various inputs and scenarios

```
def test_add_two_numbers(self):  
    result = add_two_numbers(1, 2)  
    self.assertEqual(result, 3)
```

```
def test_add_two_negative_numbers(self):  
    result = add_two_numbers(-1, -2)  
    self.assertEqual(result, -3)
```

```
def test_add_decimals_numbers(self):  
    result = add_two_numbers(10.5, 3.25)  
    self.assertEqual(result, 13.75)
```


Why Unit Test?

Find bugs early

```
def test_add_letters(self):  
    result = add_two_numbers("A", "B")  
    self.assertEqual(result, "AB")
```

Why Unit Test?

Find bugs early

```
def test_add_letters(self):  
    result = add_two_letters("A", "B")  
    self.assertEqual(result, "AB")
```

```
def test_add_letter_and_number(self):  
    result = add_two_numbers(1, "B")  
    self.assertEqual( )
```

Why Unit Test?

Find bugs early

```
def add_two_numbers(first, second):  
    """Adds up both numbers and return the sum.  
    Input values must be numbers."""  
    if not isinstance(first, (int, float)) \  
        or not (isinstance(second, (int, float))):  
        raise ValueError("Inputs must be numbers.")  
    return first + second
```

Why Unit Test?

Find bugs early

```
def add_two_numbers(first, second):  
    """Adds up both numbers and return the sum.  
    Input values must be numbers."""  
    if not isinstance(first, (int, float)) \  
        or not (isinstance(second, (int, float))):  
        raise ValueError("Inputs must be numbers.")  
    return first + second
```

```
def test_add_letters(self):  
    with self.assertRaises(ValueError):  
        add_two_numbers("A", "B")
```

```
def test_add_letter_and_number(self):  
    with self.assertRaises(ValueError):  
        add_two_numbers(1, "B")
```

Why Unit Test?

Provides documentation and
code example

```
def test_add_two_numbers(self):  
    ...  
  
def test_add_two_negative_numbers(self):  
    ...  
  
def test_add_decimals_numbers(self):  
    ...  
  
def test_add_letters(self):  
    """Will raise error, only numbers allowed."""
```

Why Unit Test?

Facilitate future modification

- Compatibility against different Python/library version

```
>>> python3.8 -m unittest test_add_number
```

```
>>> python3.9 -m unittest test_add_number
```

```
>>> python3.10 -m unittest test_add_number
```


Why Unit Test?

Facilitate future modification

- Catch deprecation warnings

```
from collections import Mapping # Deprecated in Python 3.10
```

```
>>> python3.9 -Wd -m unittest test_add_number
```

```
DeprecationWarning: Using or importing the ABCs from  
'collections' instead of from 'collections.abc' is  
deprecated since Python 3.3, and in 3.10 it will stop  
working
```

Why Unit Test?

Continuous Integration (CI)

The practice of using automation to frequently integrating (aka merging) code changes from various contributors into a single project.

Continuous Integration



Continuous Integration



tldr; do it

@mariatta

Why CI

**Integrating Code is Costly
(and Risky)**

@mariatta

Why CI

Integrating Code is Costly (and Risky)



Does the code work?

Why CI

Integrating Code is Costly (and Risky)



Does the code work?



Does it do what it's supposed to do?

Why CI

Integrating Code is Costly (and Risky)



Does the code work?



Does it do what it's supposed to do?



What are the constraints?

Why CI

Integrating Code is Costly (and Risky)



Does the code work?



Does it do what it's supposed to do?



What are the constraints?



Does it handle exceptions?

Why CI

Integrating Code is Costly (and Risky)



Does the code work?



Does it do what it's supposed to do?



What are the constraints?



Does it handle exceptions?



Is it compatible with the rest of the software?

Merging Code Without CI



Get a copy of the code to be merged

Merging Code Without CI



Get a copy of the code to be merged



Set up local environments to test the code

Merging Code Without CI



Get a copy of the code to be merged



Set up local environments to test the code



Run the tests

Merging Code with CI



Get a copy of the code to be merged **automatically**





Set up local environments to test the code **automatically**




Run the tests **automatically**

Merging Code with CI



**All checks have passed**
2 successful checks

[Show all checks](#)

**This branch has no conflicts with the base branch**
Merging can be performed automatically.

Squash and merge

▼

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Adding CI

- GitHub Actions (<https://github.com/features/actions>)
- GitLab Pipelines (<https://docs.gitlab.com/pipelines>)
- Circle CI (<https://circleci.com>)
- Travis CI (<https://travis-ci.org>)
- etc

GitHub Actions

Docs: <https://github.com/features/actions>

Configured by adding a YAML file on your repo under .github/workflows directory

The YAML file contains a set of instructions to tell the CI how to run the tests.

GitHub Actions

Running Unittest

Example Repo: https://github.com/Mariatta/sample_ci_repo

.github/workflows/ci.yml

jobs:

test:

name: test w/ Python \${ matrix.python-version }

runs-on: ubuntu-latest

strategy:

matrix:

python-version: ["3.8", "3.9"]

steps:

- uses: actions/checkout@v2
- name: Install Dependencies
run: python3 -m pip install -U pip coverage
- name: Run Tests
run: python -m unittest test_add_number

GitHub Actions

Check Code Style

Example Repo: https://github.com/Mariatta/sample_ci_repo

.github/workflows/ci.yml

jobs:

black-formatting:

name: code autoformatting with black

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2
- name: Install Dependencies
run: python3 -m pip install -U pip black
- name: Check code with black
run: black --check .

GitHub Actions

Check Code Coverage

Example Repo: https://github.com/Mariatta/sample_ci_repo

.github/workflows/ci.yml

jobs:

code-coverage:

name: Code coverage

runs-on: ubuntu-latest


steps:

- uses: actions/checkout@v2
- name: Install Dependencies
run: python3 -m pip install -U pip coverage
- name: Run test and produce coverage report
run: |
coverage run test_add_number.py
coverage report

GitHub Actions

Example Repo: https://github.com/Mariatta/sample_ci_repo

Add more commits by pushing to the **change-it** branch on **Mariatta/sample_ci_repo**.








✓

All checks have passed

5 successful checks

Hide all checks

✓	 Unittests / test w/ Python 3.8 (pull_request)	Successful in 12s	Details
✓	 Unittests / test w/ Python 3.9 (pull_request)	Successful in 9s	Details
✓	 Unittests / code autoformatting with black (pull_request)	Successful in 8s	Details
✓	 codecov/patch — Coverage not affected		Details
✓	 codecov/project — No report found to compare against		Details

✓

This branch has no conflicts with the base branch

Merging can be performed automatically.

Squash and merge

▼

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Advanced Topics on Testing

unittest module documentation

<https://docs.python.org/3/library/unittest.html>

Python Tutorial: Unit Testing Your Code with the unittest Module, by Corey Schafer

<https://youtu.be/6tNS--WetLI>

Getting Started With Testing in Python, Real Python

<https://realpython.com/python-testing/>

Advanced Topics on Testing

mock object library

<https://docs.python.org/3/library/unittest.mock.html>

Understanding the Python Mock Object Library, Real Python

<https://realpython.com/python-mock-library/>

Demystifying the Patch Function, PyCon US talk by Lisa Roach

<https://www.youtube.com/watch?v=ww1UsGZV8fQ>

Advanced Topics on Testing

pytest framework: <https://docs.pytest.org/>

Introduction to Unit Testing in Python with Pytest, PyCon US Tutorial by Michael Tom-Wing and Christie Wilson

<https://youtu.be/UPanUFVFfzY>

Advanced Topics on Testing

End-to-end Testing

Testing the workflow of your software application from start to finish.

What is End-to-End Testing and When Should You Use It?, freeCodeCamp

<https://www.freecodecamp.org/news/end-to-end-testing-tutorial/>

Selenium with Python

<https://selenium-python.readthedocs.io/>

Unit Testing & CI



do it



THANK YOU

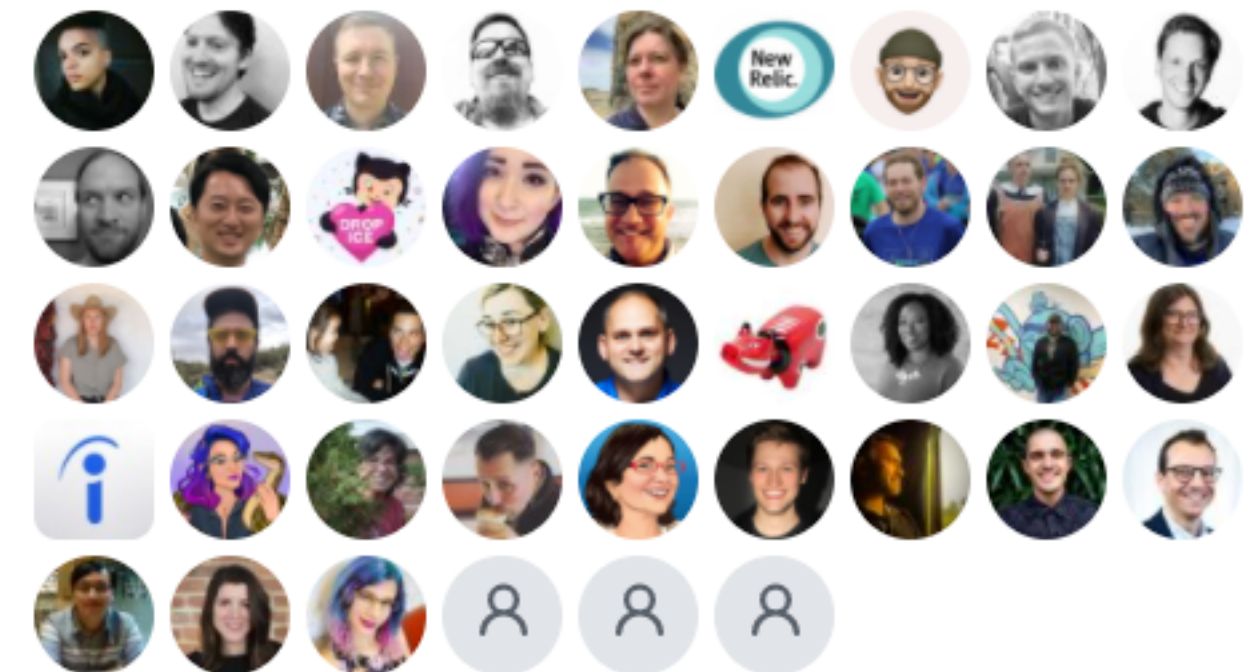


Sponsor on GitHub

<https://github.com/sponsors/python>

<https://github.com/sponsors/Mariatta>

42 sponsors are funding Mariatta's work.





THANK YOU



Sponsor on GitHub

<https://github.com/sponsors/python>

<https://github.com/sponsors/Mariatta>

42 sponsors are funding Mariatta's work.

