

# Unit testing

Unit testing is a method for testing code to make sure a set of test cases work as expected. Tests can also be used to help track and plan the functionality of your code.

## Example: Calculate the distance between two angles (in degrees)

Let's start by writing a simple version of our function:

```
In [1]: def angle_distance(a, b):  
        return abs(b - a)
```

## Writing our first test

Now we will write a test to see if this works as expected. For this example the test is in the same file as the original code, but in a real case the test should be contained in its own file or folder. As long as the filename has the word `test` in it calling the command `nosetests` will automatically find and run the file.

```
In [2]: import unittest  
  
class TestAngle(unittest.TestCase):  
    def test_small_angles(self):  
        '''Test distance between small angles'''  
        a = 10  
        b = 90  
        expected = 80  
        result = angle_distance(a, b)  
        self.assertEqual(result, expected)
```

## Running the test

To run the test inside the notebook we can use this block, typically this would be run from the console with the command `nosetests`.

```
In [3]: unittest.main(argv=['first-arg-is-ignored'], exit=False);
```

```
·  
-----  
Ran 1 test in 0.001s  
  
OK
```

## Add test for edge case 1

We can see that our test has worked as expected, but we have only tested one case, we need to also test some edge cases.

```
In [4]: class TestAngle(unittest.TestCase):
def test_small_angles(self):
    '''Test distance between small angles'''
    a = 10
    b = 90
    expected = 80
    result = angle_distance(a, b)
    self.assertEqual(result, expected)

def test_large_angles(self):
    '''Test distance between large angles'''
    a = 0
    b = 270
    expected = 90
    result = angle_distance(a, b)
    self.assertEqual(result, expected)
```

```
In [5]: unittest.main(argv=['first-arg-is-ignored'], exit=False);
```

```
F.
=====
FAIL: test_large_angles (__main__.TestAngle)
Test distance between large angles
-----
Traceback (most recent call last):
  File "/tmp/ipykernel_9930/4248502634.py", line 16, in test_large_angles
    self.assertEqual(result, expected)
AssertionError: 270 != 90

-----
Ran 2 tests in 0.001s

FAILED (failures=1)
```

## Fix the first bug

This test has failed, we have found a bug in our code! We have not accounted for the angle wrapping around, the largest distance two angles can be is 180 before they start getting closer again. Let's update our function:

```
In [6]: def angle_distance(a, b):
return abs(b - a) % 180
```

```
In [7]: unittest.main(argv=['first-arg-is-ignored'], exit=False);
```

```
..
-----
Ran 2 tests in 0.001s
```

## Add test for edge case 2

The code should be able to handle distances between angles on either side of the branch cut.

```
In [8]: class TestAngle(unittest.TestCase):
def test_small_angles(self):
    '''Test distance between small angles'''
    a = 10
    b = 90
    expected = 80
    result = angle_distance(a, b)
    self.assertEqual(result, expected)

def test_large_angles(self):
    '''Test distance between large angles'''
    a = 0
    b = 270
    expected = 90
    result = angle_distance(a, b)
    self.assertEqual(result, expected)

def test_wrapping_angles(self):
    '''Test distance around wrapping angle'''
    a = 1
    b = 359
    expected = 2
    result = angle_distance(a, b)
    self.assertEqual(result, expected)
```

```
In [9]: unittest.main(argv=['first-arg-is-ignored'], exit=False);
```

```
..F
```

```
=====
FAIL: test_wrapping_angles (__main__.TestAngle)
Test distance around wrapping angle
-----
```

```
Traceback (most recent call last):
```

```
  File "/tmp/ipykernel_9930/4238174130.py", line 24, in test_wrapping_angles
    self.assertEqual(result, expected)
AssertionError: 178 != 2
```

```
-----
Ran 3 tests in 0.001s
```

```
FAILED (failures=1)
```

## Fix the second bug

Looks like we have found another breaking point in our code. We will update it again:

```
In [10]: def angle_distance(a, b):  
         d = abs(b - a)  
         return min(360 - d, d)
```

```
In [11]: unittest.main(argv=['first-arg-is-ignored'], exit=False);
```

```
...
```

```
-----  
Ran 3 tests in 0.001s
```

```
OK
```

## Add test for edge case 3

What about the case where the original angles are larger than 360 ? We will write another test:

```
In [12]: class TestAngle(unittest.TestCase):  
         def test_small_angles(self):  
             '''Test distance between small angles'''  
             a = 10  
             b = 90  
             expected = 80  
             result = angle_distance(a, b)  
             self.assertEqual(result, expected)  
  
         def test_large_angles(self):  
             '''Test distance between large angles'''  
             a = 0  
             b = 270  
             expected = 90  
             result = angle_distance(a, b)  
             self.assertEqual(result, expected)  
  
         def test_wrapping_angles(self):  
             '''Test distance around wrapping angle'''  
             a = 1  
             b = 359  
             expected = 2  
             result = angle_distance(a, b)  
             self.assertEqual(result, expected)  
  
         def test_large_input_angles(self):  
             '''Test distance when input angles are above 360'''  
             a = 720  
             b = 270  
             expected = 90  
             result = angle_distance(a, b)  
             self.assertEqual(result, expected)
```

```
In [13]: unittest.main(argv=['first-arg-is-ignored'], exit=False);
```

```
.F..
```

```
=====
FAIL: test_large_input_angles (__main__.TestAngle)
Test distance when input angles are above 360
=====
```

```
Traceback (most recent call last):
```

```
  File "/tmp/ipykernel_9930/4200136150.py", line 32, in test_large_input_angles
```

```
    self.assertEqual(result, expected)
```

```
AssertionError: -90 != 90
```

```
-----
Ran 4 tests in 0.002s
```

```
FAILED (failures=1)
```

## Fix the third bug

To fix the bug for this case we need to make sure the input values are re-cast in the range `[0, 360)` :

```
In [14]: def angle_distance(a, b):
         d = abs((b % 360) - (a % 360))
         return min(360 - d, d)
```

```
In [15]: unittest.main(argv=['first-arg-is-ignored'], exit=False);
```

```
....
```

```
-----
Ran 4 tests in 0.001s
```

```
OK
```

## Cleaning up the tests

Looking at our tests now we can see that is really the same test four times over, just with different values. `unittest` allows us to create `subTests` for cases like this:

```
In [16]: class TestAngle(unittest.TestCase):
         def test_small_angles(self):
             '''Test distance between angles'''
             a = [10, 0, 1, 720]
             b = [90, 270, 359, 270]
             expected = [80, 90, 2, 90]
             for i, j, e in zip(a, b, expected):
                 with self.subTest(a=i, b=j):
                     result = angle_distance(i, j)
                     self.assertEqual(result, e)
```

```
In [17]: unittest.main(argv=['first-arg-is-ignored'], exit=False);
```

```
.
```

---

```
Ran 1 test in 0.001s
```

```
OK
```

If any of the sub-tests fail the error message will display all the keywords we passed into `subTest` (e.g. `(a=10, b=90)` )

## Things to keep in mind about writing tests

- Only write tests for code you have written
- Every function definition should have at least one test
- Every `if-else` block in a function should have a test (this is known as "test coverage" and can be calculated by running `nosetests --with-coverage` )
- If you find a bug in your code, write a test that reproduces the bug, then fix the bug
- Never remove test cases
- Github can be configured to run tests before allowing PR to be merged to ensure new code does not break existing code
- Write your tests first then write your code until all tests pass (this is known as "test driven development")

In [ ]: