Тестирование

In [1]:
```
# test_python.py

import unittest


class TestPython(unittest.TestCase):
    def test_float_to_int_coercion(self):
        self.assertEqual(1, int(1.0))

    def test_get_empty_dict(self):
        self.assertIsNone({}.get('key'))

    def test_trueness(self):
        self.assertTrue(bool(10))
```

project/tests $> python3 -m unittest test_python.py

In [2]:
```
# test_division.py

import unittest


class TestDivision(unittest.TestCase):
    def test_integer_division(self):
        self.assertIs(10 / 5, 2)
```

project/tests $> python3 -m unittest test_division.py

In [7]:
```
import requests
```

```
class Asteroid:
    BASE_API_URL = 'https://api.nasa.gov/neo/rest/
v1/neo/{}?api_key=DEMO_KEY'

    def __init__(self, spk_id):
        self.api_url = self.BASE_API_URL.format(sp
k_id)

    def get_data(self):
        return requests.get(self.api_url).json()

    @property
    def name(self):
        return self.get_data()['name']

    @property
    def diameter(self):
        return int(self.get_data()['estimated_diam
eter']['meters']['estimated_diameter_max'])

    @property
    def closest_approach(self):
        closest = {
            'date': None,
            'distance': float('inf')
        }
        for approach in self.get_data()['close_app
roach_data']:
            distance = float(approach['miss_distan
ce']['lunar'])
            if distance < closest['distance']:
                closest.update({

                    'date': approach['close_approa
ch_date'],
                    'distance': distance
```

```
                })
        return closest
```

In [9]:
```
apophis = Asteroid(2099942)

print(f'Name: {apophis.name}')
print(f'Diameter: {apophis.diameter}m')
```

```
Name: 99942 Apophis (2004 MN4)
Diameter: 682m
```

In [ ]:

```
import json
import unittest
from unittest.mock import patch

from asteroid import Asteroid


class TestAsteroid(unittest.TestCase):
    def setUp(self):
        self.asteroid = Asteroid(2099942)

    def mocked_get_data(self):
        with open('apophis_fixture.txt') as f:
            return json.loads(f.read())

    @patch('asteroid.Asteroid.get_data', mocked_ge
t_data)
    def test_name(self):
        self.assertEqual(
            self.asteroid.name, '99942 Apophis (20
04 MN4)'
        )

    @patch('asteroid.Asteroid.get_data', mocked_ge
t_data)
    def test_diameter(self):
        self.assertEqual(self.asteroid.diameter, 6
82)
```

In [10]:
```
print(f'Date: {apophis.closest_approach["date"]}')
print(f'Distance: {apophis.closest_approach["distance"]:.2} LD')
```

```
Date: 2029-04-13
Distance: 0.099 LD
```