

UnitTest

Практикум по программированию

```
import unittest

class TestStringMethods(unittest.TestCase):

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

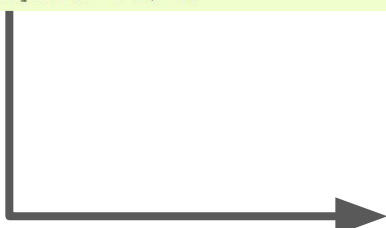
    def test_isupper(self):
        self.assertTrue('FOO'.isupper())
        self.assertFalse('Foo'.isupper())

    def test_split(self):
        s = 'hello world'
        self.assertEqual(s.split(), ['hello', 'world'])
        # Проверим, что s.split не работает, если разделитель - не строка
        with self.assertRaises(TypeError):
            s.split(2)

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

Подтесты (subTest())

```
class NumbersTest(unittest.TestCase):  
  
    def test_even(self):  
        """  
        Test that numbers between 0 and 5 are all even.  
        """  
        for i in range(0, 6):  
            with self.subTest(i=i):  
                self.assertEqual(i % 2, 0)
```



```
=====
```

```
FAIL: test_even (__main__.NumbersTest) (i=1)
```

```
-----
```

```
Traceback (most recent call last):  
  File "subtests.py", line 32, in test_even  
    self.assertEqual(i % 2, 0)  
AssertionError: 1 != 0
```

```
=====
```

```
FAIL: test_even (__main__.NumbersTest) (i=3)
```

```
-----
```

```
Traceback (most recent call last):  
  File "subtests.py", line 32, in test_even  
    self.assertEqual(i % 2, 0)  
AssertionError: 1 != 0
```

```
=====
```

```
FAIL: test_even (__main__.NumbersTest) (i=5)
```

```
-----
```

```
Traceback (most recent call last):  
  File "subtests.py", line 32, in test_even  
    self.assertEqual(i % 2, 0)  
AssertionError: 1 != 0
```

Виды проверок

<code>assertEqual(a, b)</code>	<code>a == b</code>
<code>assertNotEqual(a, b)</code>	<code>a != b</code>
<code>assertTrue(x)</code>	<code>bool(x) is True</code>
<code>assertFalse(x)</code>	<code>bool(x) is False</code>
<code>assertIsNone(x)</code>	<code>x is None</code>
<code>assertIsNotNone(x)</code>	<code>x is not None</code>
<code>assertIn(a, b)</code>	<code>a in b</code>
<code>assertNotIn(a, b)</code>	<code>a not in b</code>

<code>assertGreater(a, b)</code>	<code>a > b</code>
<code>assertGreaterEqual(a, b)</code>	<code>a >= b</code>
<code>assertLess(a, b)</code>	<code>a < b</code>
<code>assertLessEqual(a, b)</code>	<code>a <= b</code>
<code>assertCountEqual(a, b)</code>	а и b содержат те же элементы в одинаковых количествах, но порядок не важен