PyUnit (unittest)

1. Введение

- Модуль **unittest** (изначально проект назывался **PyUnit**) входит в состав стандартной библиотеки Python.
- Относится к архитектуре **xUnit**, похож на JUnit, NUnit, PHPUnit и другие.
- Также имеет объектно-ориентированную основу.
- Дистрибутив Python содержит полную документацию по модулю в разделе unittest (module), также доступна в соответствующем разделе на официальном сайте (ссылка для Python 3.8)
- Тесты обычно помещаются в отдельный файл (модуль) с расширением .py
- Запуск тестов из под командной строки:

```
python -m unittest файл-модуля
```

- В основе имеет 4 основных понятия:
  - **Test fixture** - конкретный тест, вместе с инициализацией и последующей очисткой
  - **Test Case** - "тестовый случай" - класс с тестами
  - **Test Suite** - набор тестовый случая
  - **Test Runner** - программа, осуществляющая выполнение тестов

2. Создание тестов

- Считаем, что тестируемый модуль и тесты к нему находятся в одной директории, в виде файлов:
  - **Color.py**—тестируемый модуль (класс «Цвет»)
  - **ColorTest.py**—файл с тестами
- Для использования модуль **unittest** нужно импортировать:

```
import unittest
```

- Также не забываем импортировать сам тестируемый модуль, либо его содержимое:

```
from Color import *
```

- Для создания тест кейса (класса с тестами) нужно создать класс, наследующий от unittest.**TestCase**:

```
class ColorTest(unittest.TestCase):
```

- Отдельные тесты (fixture) - методы, начинающиеся с текста 'test':

```
def testFromHtml(self):
    html_color = "#7f5f3f"
    color = Color(0,0,0,0);
    color.FromHTML(html_color)
```

- Функции проверок имеют виды self.**assertX**(), где X—вид проверяемого условия:

```
self.assertEqual(color.Red, 0x7f)
self.assertEqual(color.Green, 0x5f)
self.assertEqual(color.Blue, 0x3f)
self.assertEqual(color.Alpha, 0xff)
```

- Наиболее ходовые проверки:

Метод	Описание (убедиться, что...)
assertEqual(first, second)	Значения равны
assertTrue(condition)	Условие выполняется (верно)
assertFalse(condition)	Условие НЕ выполняется (ложно)
assertIs(first, second)	Ссылки на один и тот же объект
assertIsNot(first, second)	Ссылки на разные объекты
assertIsNone(expression)	Выражение даёт результат **None** (аналог null в Python)
assertIsNotNone(expression)	Выражение НЕ даёт результат **None**
assertIsInstance(object, class)	Объект является представителем указанного класса
assertNotIsInstance(object, class)	Объект НЕ является представителем указанного класса
assertIn(object, container)	Объект входит в контейнер
assertNotIn(object, container)	Объект НЕ входит в контейнер
assertRaises(exception, fn, *args)	Функция кидает (в терминологии Python «поднимает») исключение

- **assertRaises**() поддерживает оператор **with**, которым более удобно вызвать функцию:

```
with self.assertRaises(ZeroDivisionError):
  calc.Divide(1,0)
```

- Все функции типа **assertX**() также принимают аргумент msg, где можно указать сообщение, выводимое в случае провала проверки.
- Для инициализации перед каждым тестом используется **setUp**(), для приборки **tearDown**().
- Механизм по типу data provider в базовом модуле **unittest** отсутствует.

- Приходим к следующему виду файла **ColorTest.py**:

```
import unittest
from Color import *

class ColorTest(unittest.TestCase):
  def testFromHtml(self):
    html_color = "#7f5f3f"
    color = Color(0,0,0,0);
    color.FromHTML(html_color)

    self.assertEqual(color.Red, 0x7f)
    self.assertEqual(color.Green, 0x5f)
    self.assertEqual(color.Blue, 0x3f)
    self.assertEqual(color.Alpha, 0xff)
```

- Открываем в директории (где находятся файлы) командную строку, запускаем:

```
python -m unittest ColorTest.py
```

- Получаем следующую картину: