Python как первый язык

День 6

Инфраструктура

часть 2

Качество кода

Качество кода

- Автоматическая проверка стиля

Качество кода

- Автоматическая проверка стиля
- Проверка на синтаксические ошибки

Качество кода

- Автоматическая проверка стиля
- Проверка на синтаксические ошибки
- Юнит-тестирование

Качество кода

- Автоматическая проверка стиля
- Проверка на синтаксические ошибки
- Юнит-тестирование
- Автоматическое e2e тестирование

Unit-тестирование

Модуль unittest

Unit-тестирование

Модуль unittest

```
01: import unittest
02:
03: class MyTestCase(unittest.TestCase):
04:     def test_case(self):
05:         self.assertEqual(True, test_fn())
06:
07: unittest.main()
```

Набор тестов - внутри одного класса

Unit-тестирование

Модуль unittest

```
01: import unittest
02:
03: class MyTestCase(unittest.TestCase):
04:     def test_case(self):
05:         self.assertEqual(True, test_fn())
06:
07: unittest.main()
```

Наследуется от unittest.TestCase

Unit-тестирование

Модуль unittest

```
01: import unittest
02:
03: class MyTestCase(unittest.TestCase):
04:     def test_case(self):
05:         self.assertEqual(True, test_fn())
06:
07: unittest.main()
```

Тесты - просто методы, которые начинаются с test_

Unit-тестирование

Модуль unittest

```
01: import unittest
02:
03: class MyTestCase(unittest.TestCase):
04:     def test_case(self):
05:         self.assertEqual(True, test_fn())
06:
07: unittest.main()
```

Используются методы TestCase для сравнения значений

Unit-тестирование

Модуль unittest

```
01: import unittest
02:
03: class MyTestCase(unittest.TestCase):
04:     def test_case(self):
05:         self.assertEqual(True, test_fn())
06:
07: unittest.main()
```

Сравниваем ожидаемое и полученное значение

Unit-тестирование

Модуль unittest

```
01: import unittest
02:
03: class MyTestCase(unittest.TestCase):
04:     def test_case(self):
05:         self.assertEqual(True, test_fn())
06:
07: unittest.main()
```

Запускаем все тест-кейсы

Unit-тестирование

Модуль unittest

```
$ python test.py

.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
```

Unit-тестирование

Модуль unittest

```
01: import unittest
02:
03: class MyTestCase(unittest.TestCase):
04:     def test_case(self):
05:         self.assertEqual(True, False)
06:
07: unittest.main()
```

Unit-тестирование

Модуль unittest

```
$ python test.py

F
======================================================================
FAIL: test_alpha (__main__.Test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test.py", line 6, in test_alpha
    self.assertEqual(True, False)
AssertionError: True != False

----------------------------------------------------------------------
Ran 1 test in 0.001s

FAILED (failures=1)
```

Unit-тестирование

Модуль unittest

```
$ python test.py

F
======================================================================
FAIL: test_alpha (__main__.Test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test.py", line 6, in test_alpha
    self.assertEqual(True, False)
AssertionError: True != False

----------------------------------------------------------------------
Ran 1 test in 0.001s

FAILED (failures=1)
```

Линтинг кода

Библиотека pylint

Web и Python

часть 2

GET и POST запросы

```
01: from django.http import HttpResponse
02:
03: def home(req):
04:     return HttpResponse('Hello')
```

GET и POST запросы

```
01: from django.http import HttpResponse
02:
03: def home(req):
04:     return HttpResponse('Hello')
```

req - это объект HttpRequest

GET и POST запросы

```
01: from django.http import HttpResponse
02:
03: def home(req):
04:     return HttpResponse('Hello')
```

req - это объект HttpRequest

```
req.method # метод запроса GET или POST
```

GET и POST запросы

```
01: from django.http import HttpResponse
02:
03: def home(req):
04:     return HttpResponse('Hello')
```

req - это объект HttpRequest

```
req.method # метод запроса GET или POST
req.body # данные запроса (строка формате JSON)
```

GET и POST запросы

```
01: from django.http import HttpResponse
02:
03: def home(req):
04:     return HttpResponse('Hello')
```

req - это объект HttpRequest

```
req.method # метод запроса GET или POST
req.body # данные запроса (строка формате JSON)
req.path # url запроса
```

Создание запросов

Библиотека requests

```
01: from requests import get, post
```

Создание запросов

Библиотека requests

```
01: from requests import get, post
02: import json
03:
04: def call(url, method='GET', data={}, headers={}):
05:     if method == 'GET':
06:         req = get(url, params=data, headers=headers)
07:     elif method == 'POST':
08:         req = post(url,
                   params=json.dumps(data),
                   headers=headers)
09:
10:     req.encoding = 'utf-8'
11:
12:     return req.text
```

Создание запросов

Библиотека requests

```
01: from requests import get, post
02: import json
03:
04: def call(url, method='GET', data={}, headers={}):
05:     if method == 'GET':
06:         req = get(url, params=data, headers=headers)
07:     elif method == 'POST':
08:         req = post(url,
                   params=json.dumps(data),
                   headers=headers)
09:
10:     req.encoding = 'utf-8'
11:
12:     return req.text
```

Создание запросов

Библиотека requests

```
01: from requests import get, post
02: import json
03:
04: call('http://yandex.ru')
```

curl

```
$ curl -XGET “http://yandex.ru”
```

curl

```
$ curl -XPOST \
   -H "Content-Type: application/json" \
   "http://127.0.0.1:8000" \
   -d ‘{ \
       "message" : { \
       ”text": “Hello", \
       ”chat": {"id": “172862922”}} \
   }'
```