

**Московский государственный технический
университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №5
«Модульное тестирование в Python.»

Выполнил:
студент группы ИУ5-34Б:
Такташова Дарья Юрьевна
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2022 г.

Задание:

Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4.

1. Модифицируйте код таким образом, чтобы он был пригоден для модульного тестирования.
2. Разработайте модульные тесты. В модульных тестах необходимо применить следующие технологии:
 - TDD - фреймворк (не менее 3 тестов).
 - BDD - фреймворк (не менее 3 тестов).
 - Создание Моск-объектов (необязательное дополнительное задание).

Код программы:

1_pytest/test_field/test_field.py

```
1_pytests > test_field > test_field.py > ...
1  import pytest
2
3  def field(items, *args):
4      result = {}
5      assert len(args) > 0
6      for d in items:
7          for i,j in d.items():
8              if i in args:
9                  result[i] = j
10             if len(result) == 1:
11                 s = result.popitem()
12                 s = "" + str(s[1]) + ""
13                 yield s
14             else :
15                 yield result
16
17 @pytest.fixture
18 def goods():
19     goods = [{'title': 'Ковер', 'price': 2000, 'color': 'green'},
20             {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}]
21     return goods
22
23
24 def test_field_one_arg(goods):
25     t = list(field(goods, 'title'))
26     res = ["Ковер", "Диван для отдыха"]
27     assert res == t
28
29 def test_field_two_arg(goods):
30     t = field(goods, 'price', 'title')
31     assert next(t) == {'title': 'Ковер', 'price': 2000} and next(t) == {'title': 'Диван для отдыха', 'price': 5300}
32
```

1_pytest/test_sort/test_sort.py

```
1_pytests > test_sort > test_sort.py > ...
1  import pytest
2
3  def my_sort1(data):
4      return sorted(data, key = abs, reverse = True)
5
6  def my_sort2(data):
7      return sorted(data, key = lambda n: -abs(n))
8
9  def test_my_sort1():
10     data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
11     res = [123, 100, -100, -30, 4, -4, 1, -1, 0]
12     assert res == my_sort1(data)
13
14  def test_my_sort2():
15     data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
16     res = [123, 100, -100, -30, 4, -4, 1, -1, 0]
17     assert res == my_sort2(data)
```

1_pytest/test_unique/test_unique.py

1_pytests > test_unique > test_unique.py > ...

```
1 import pytest
2 import gen_random
3
4 class Unique(object):
5     def __init__(self, items, **kwargs):
6         self.seen = []
7         for i in items:
8             if len(kwargs) > 0 and kwargs["ignore_case"]:
9                 flag = True
10                 for j in self.seen:
11                     if j.lower() == i.lower():
12                         flag = False
13                 if flag:
14                     (self.seen).append(i)
15             else:
16                 if i in self.seen:
17                     continue
18                 self.seen.append(i)
19
20     def __next__(self):
21         if len(self.seen) == 0:
22             raise StopIteration
23         item = self.seen[0]
24         del self.seen[0]
25         return item
26
27     def __iter__(self):
28         return self
29
30 @pytest.fixture
31 def data():
32     d = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
33     return d
34
35 def test_unique_ignore_case_True(data):
36     t = list(Unique(data, ignore_case = True))
37     res = ['a', 'b']
38     assert res == t
39
40 def test_unique_ignore_case_False(data):
41     t = list(Unique(data))
42     res = ['a', 'A', 'b', 'B']
43     assert res == t
44
45 def test_unique_numbers():
46     data = [1,1,1,1,1,1,2,2,2,2,2]
47     t = list(Unique(data))
48     res = [1, 2]
49     assert res == t
50
```

Пример выполнения:

```

dasha@dasha-B0HB-WAX9:~/Desktop/BIKT/BIKT/1_pytests$ python3 -m pytest
===== test session starts =====
platform linux -- Python 3.10.6, pytest-7.2.0, pluggy-1.0.0
rootdir: /home/dasha/Desktop/BIKT/BIKT/1_pytests
plugins: bdd-6.1.0, anyio-3.6.2
collected 7 items

test_field/test_field.py ..                                [ 28%]
test_sort/test_sort.py ..                                [ 57%]
test_unique/test_unique.py ...                            [100%]

===== 7 passed in 0.04s =====

```

2_pytest_bdd/features:

f.feature

```

2_pytest_bdd > features > f.feature
1  Feature: Sorting elements by abs with function sorted
2
3  Scenario: Data need to be sorted by abs with function sorted
4  Given some data
5  When data get sorted with my_sort1
6  Then data is sorted
7

```

f2.feature

```

2_pytest_bdd > features > f2.feature
1  Feature: Sorting elements by abs with lambda expression
2
3  Scenario: Data need to be sorted by abs with lambda expression
4  Given some data
5  When data get sorted with my_sort2
6  Then data is sorted

```

f3.feature

```

2_pytest_bdd > features > f3.feature
1  Feature: Get only unique elements of list
2
3  Scenario: Getting list of unique elements
4  Given some data
5  When data is getting unique
6  Then data is unique

```

2_pytest_bdd/steps:

test_sort.py

```

2_pytest_bdd > steps > test_sort.py > ...
1  from func import my_sort1, my_sort2
2  import pytest
3  from pytest_bdd import scenario, given, when, then
4
5  @scenario('/home/dasha/Desktop/BIKT/BIKT/2_pytest_bdd/features/f.feature', 'Data need to be sorted by abs with function sorted')
6  def testing_my_sort1():
7      pass
8
9  @scenario('/home/dasha/Desktop/BIKT/BIKT/2_pytest_bdd/features/f2.feature', 'Data need to be sorted by abs with lambda expression')
10 def testing_my_sort2():
11     pass
12
13 @given('some data', target_fixture= 'data')
14 def data():
15     return [4, -30, 100, -100, 123, 1, 0, -1, -4]
16
17 @when('data get sorted with my_sort1', target_fixture='sorted_data')
18 def sorted_data(data):
19     return my_sort1(data)
20
21 @then('data is sorted')
22 def sorted_data(sorted_data):
23     assert sorted_data == [123, 100, -100, -30, 4, -4, 1, -1, 0]
24
25 @when('data get sorted with my_sort2', target_fixture='sorted_data')
26 def sorted_data(data):
27     return my_sort2(data)
28

```

test_unique.py

```
2_pytest_bdd > steps > test_unique.py > ...
1 from func import Unique
2 import pytest
3 from pytest_bdd import scenario, given, when, then
4
5 @scenario('/home/dasha/Desktop/BIKT/BIKT/2_pytest_bdd/features/f3.feature', 'Getting list of unique elements')
6 def testing_unique():
7     pass
8
9 @given('some data', target_fixture='data')
10 def data():
11     return ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
12
13 @when('data is getting unique', target_fixture='unique_data')
14 def unique_data(data):
15     return list(Unique(data, ignore_case = True))
16
17 @then('data is unique')
18 def res_data(unique_data):
19     assert unique_data == ['a', 'b']
20
```

Пример выполнения:

```
dasha@dasha-BOHB-WAX9:~/Desktop/BIKT/BIKT/2_pytest_bdd$ python3 -m pytest  
===== test session starts =====  
platform linux -- Python 3.10.6, pytest-7.2.0, pluggy-1.0.0  
rootdir: /home/dasha/Desktop/BIKT/BIKT/2_pytest_bdd  
plugins: bdd-6.1.0, anyio-3.6.2  
collected 3 items  
  
steps/test_sort.py .. [ 66%]  
steps/test_unique.py . [100%]  
  
===== 3 passed in 0.02s =====
```