**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №5

«Модульное тестирование в Python.»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-31Б |  | преподаватель каф. ИУ5 |
| Орлова Анна |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2024 г.

### Задание:

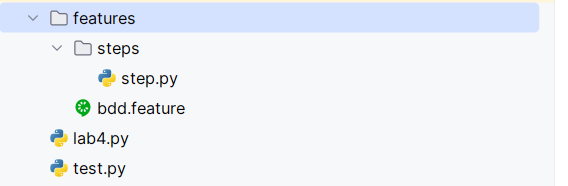
### Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4.

Модифицируйте код таким образом, чтобы он был пригоден для модульного тестирования.

Разработайте модульные тесты. В модульных тестах необходимо применить следующие технологии:

TDD - фреймворк (не менее 3 тестов).

BDD - фреймворк (не менее 3 тестов).

Текст программы

lab4.py

def field(items, \*args):  
 if not args:  
 return  
  
 for j in items:  
 if len(args) == 1:  
 if args[0] in j and j[args[0]] is not None:  
 yield j[args[0]]  
 else:  
 ans = {}  
 for i in args:  
 if i in j and j[i] is not None:  
 ans[i] = j[i]  
 if ans:  
 yield ans

test.py

*# test\_field.py*import unittest  
from lab4 import field  
  
class TestFieldFunction(unittest.TestCase):  
 def setUp(self):  
 self.goods = [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}  
 ]  
  
 def test\_extract\_single\_field(self):  
 titles = list(field(self.goods, 'title'))  
 self.assertEqual(titles, ['Ковер', 'Диван для отдыха'])  
  
 def test\_extract\_multiple\_fields(self):  
 prices = list(field(self.goods, 'title', 'price'))  
 expected = [  
 {'title': 'Ковер', 'price': 2000},  
 {'title': 'Диван для отдыха', 'price': 5300}  
 ]  
 self.assertEqual(prices, expected)  
  
 def test\_extract\_non\_existent\_field(self):  
 result = list(field(self.goods, 'non\_existent\_field'))  
 self.assertEqual(result, [])  
  
 def test\_get\_all\_list(self):  
 result = list(field(self.goods, 'title', 'price','color'))  
 expected = [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}  
 ]  
 self.assertEqual(result, expected)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

bdd.feature

Feature: Field extraction  
  
 Scenario: Extracting a single field  
 Given a list of goods  
 When extract the title field  
 Then a list  
  
 Scenario: Extracting multiple fields  
 Given a list of goods  
 When extract the title and price fields  
 Then a list of goods with title and price  
  
 Scenario: Extracting a non-existent field  
 Given a list of goods  
 When extract a non-existent field  
 Then an empty list  
  
 Scenario: Getting all fields  
 Given a list of goods  
 When extract the title, price, color fields  
 Then a list of goods with title, price, color

step.py

from behave import given, when, then, step  
from lab4 import field  
@given('a list of goods')  
def step\_given\_list\_of\_goods(context):  
 context.goods = [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}  
 ]  
  
@when('extract the title field')  
def step\_when\_extract\_title(context):  
 context.result = list(field(context.goods, 'title'))  
  
@then('a list')  
def step\_then\_result\_titles(context):  
 assert context.result == ['Ковер', 'Диван для отдыха']  
  
@when('extract the title and price fields')  
def step\_when\_extract\_title(context):  
 context.result = list(field(context.goods, 'title','price'))  
  
@then('a list of goods with title and price')  
def step\_then\_result\_titles(context):  
 expected = [{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}]  
 assert context.result == expected  
  
  
@when('extract a non-existent field')  
def step\_when\_extract\_title(context):  
 context.result = list(field(context.goods, 'qwerty'))  
  
  
@then('an empty list')  
def step\_then\_result\_titles(context):  
 assert context.result == []  
  
  
@when('extract the title, price, color fields')  
def step\_when\_extract\_title(context):  
 context.result = list(field(context.goods, 'title','price','color'))  
  
@then('a list of goods with title, price, color')  
def step\_then\_result\_titles(context):  
 expected = [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}  
 ]  
 assert context.result == expected

экранные формы с примерами выполнения программы.

