



**МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

**Институт кибербезопасности и цифровых технологий**

Кафедра КБ-3 «Разработка программных решений и системного  
программирования»

## **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2**

по дисциплине: «Сборка, тестирование и верификация программного  
продукта»

на тему: «Рецепты и ингредиенты»

Выполнил: студент группы БСБО-10-21

Филёв С. А.

Проверил: преподаватель кафедры КБ-3

Ивакин С. Н.

**Москва, 2024 г**

## Ход работы

Был создан файл в котором описано блюдо «Итальянская паста», который представлен листингом 1. В нем прописаны классы `Ingredients` и `Receipt`. Более подробно о написанном есть в комментариях в коде.

### Листинг 1. Код файла `dish.py`

```
class Ingredient:
    """Класс для описания ингредиента с атрибутами: название, вес
    в сыром виде, вес в готовом виде, стоимость."""

    def __init__(self, name: str, raw_weight: int, weight: int,
cost: int) -> None:
        if not name or raw_weight <= 0 or weight <= 0 or cost <
0:
            raise ValueError("Неверные данные для ингредиента")

        self.name = name
        self.raw_weight = raw_weight # вес в сыром виде (граммы)
        self.weight = weight # вес после приготовления (граммы)
        self.cost = cost # стоимость (в рублях)

    def __str__(self) -> str:
        return f"{self.name} (сырой вес: {self.raw_weight}г,
готовый вес: {self.weight}г, стоимость: {self.cost}руб)"

class Receipt:
    """Класс для работы с рецептом, включая расчеты себестоимости
    и веса."""

    def __init__(self, name: str, ingredients: list[tuple[str,
int, int, int]]) -> None:
        self.name = name
        self.ingredients = [Ingredient(*ingredient) for
ingredient in ingredients]
```

```

def calc_cost(self, portions=1) -> int:
    """Рассчитывает себестоимость порции."""
    total_cost = sum(ingredient.cost for ingredient in
self.ingredients)
    return total_cost * portions

def calc_weight(self, portions=1, raw=True) -> int:
    """Рассчитывает общий вес ингредиентов. Можно выбрать
сырой или готовый вес."""
    total_weight = sum(ingredient.raw_weight if raw else
ingredient.weight for ingredient in self.ingredients)
    return total_weight * portions

def __str__(self) -> str:
    ingredients_str = ", ".join(str(ingredient) for
ingredient in self.ingredients)
    return f"Рецепт: {self.name}\nИнгредиенты:
{ingredients_str}"

# Пример рецепта
if __name__ == '__main__':
    receipt_from_api = {
        "title": "Итальянская паста",
        "ingredients_list": [
            ('Мука', 100, 90, 15),
            ('Яйцо', 60, 50, 10),
            ('Оливковое масло', 20, 20, 30),
            ('Соль', 5, 5, 1),
            ('Перец', 2, 2, 2),
        ],
    }

    receipt = Receipt(receipt_from_api['title'],

```

```

receipt_from_api['ingredients_list'])

# Проверка методов
print(receipt) # Вывод рецепта и ингредиентов
print("Себестоимость (на 1 порцию):", receipt.calc_cost())
print("Себестоимость (на 3 порции):", receipt.calc_cost(3))
print("Вес сырого продукта:", receipt.calc_weight(raw=True))
print("Вес готового продукта:",
receipt.calc_weight(raw=False))

```

Вывод проверки методов представлен на рисунке 1.

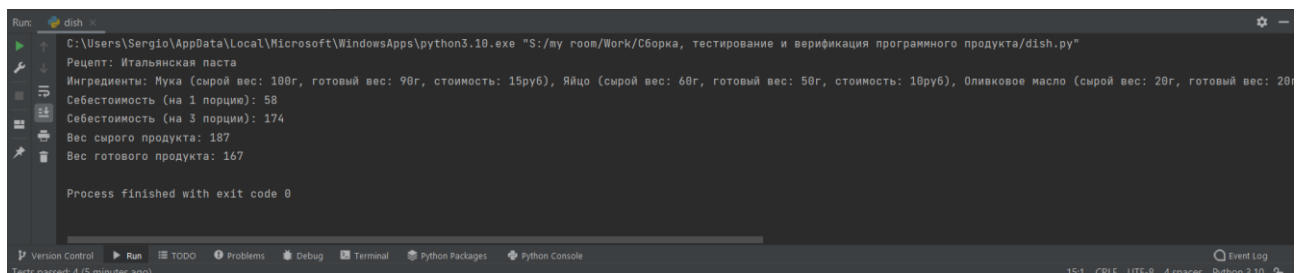


Рисунок 1. Вывод проверки методов файла dish.py

Далее создался файл test\_recipe.py для проведения тестов с помощью unittest, в котором используются методы setUp() и setUpClass(), tearDown(), и tearDownClass(). Код данного файла представлен листингом 2.

Листинг 2. Код файла test\_recipe.py

```

import unittest
from dish import Ingredient, Receipt

class TestIngredientAndReceipt(unittest.TestCase):

    @classmethod
    def setUpClass(cls):
        print("Запуск общего setup для класса
TestIngredientAndReceipt")

    @classmethod
    def tearDownClass(cls):
        print("Очистка после всех тестов

```

```
TestIngredientAndReceipt")
```

```
def setUp(self):
    """Инициализация ингредиентов и рецепта для тестов"""
    self.ingredients_list = [
        ('Мука', 100, 90, 15),
        ('Яйцо', 60, 50, 10),
        ('Оливковое масло', 20, 20, 30),
        ('Соль', 5, 5, 1),
        ('Перец', 2, 2, 2),
    ]
    self.recipe = Receipt("Итальянская паста",
self.ingredients_list)

def tearDown(self):
    print("Завершение теста")

def test_calc_cost(self):
    """Тест расчета себестоимости на одну и несколько
порций."""
    self.assertEqual(self.recipe.calc_cost(), 58) #
15+10+30+1+2 = 58
    self.assertEqual(self.recipe.calc_cost(3), 174)

def test_calc_weight_raw(self):
    """Тест расчета общего сырого веса."""
    self.assertEqual(self.recipe.calc_weight(raw=True), 187)
# 100+60+20+5+2 = 187

def test_calc_weight_cooked(self):
    """Тест расчета общего веса готового продукта."""
    self.assertEqual(self.recipe.calc_weight(raw=False),
167) # 90+50+20+5+2 = 167

def test_invalid_ingredient(self):
```

```

        """Тест обработки неверных данных ингредиента."""
        with self.assertRaises(ValueError):
            Ingredient("", -100, -90, -10)

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

```

Вывод проведения тестов представлен на рисунке 2.

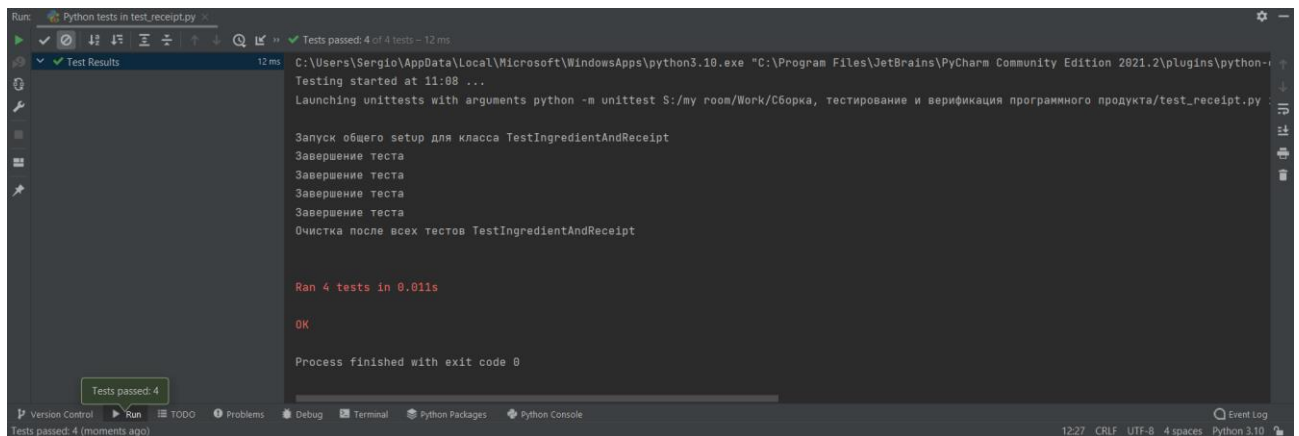


Рисунок 2 – Итог проведения тестов

Далее по аналогии произвелся подбор второго блюда на первую букву имени. Было выбрано блюдо «Спагетти карбонара». Изменению подверглись оба файла. Код обоих файлов представлен листингами 3 и 4 соответственно, а также их вывод при запуске на рисунках 3 и 4.

Листинг 3. Код файла dish.py

```

class Ingredient:
    """Класс для описания ингредиента с атрибутами: название, вес
    в сыром виде, вес в готовом виде, стоимость."""

    def __init__(self, name: str, raw_weight: int, weight: int,
cost: int) -> None:
        if not name or raw_weight <= 0 or weight <= 0 or cost <
0:
            raise ValueError("Неверные данные для ингредиента")

        self.name = name

```

```

        self.raw_weight = raw_weight # вес в сыром виде (граммы)
        self.weight = weight # вес после приготовления (граммы)
        self.cost = cost # стоимость (в рублях)

    def __str__(self) -> str:
        return f"{self.name} (сырой вес: {self.raw_weight}г,
готовый вес: {self.weight}г, стоимость: {self.cost}руб)"

class Receipt:
    """Класс для работы с рецептом, включая расчеты себестоимости
и веса."""

    def __init__(self, name: str, ingredients: list[tuple[str,
int, int, int]]) -> None:
        self.name = name
        self.ingredients = [Ingredient(*ingredient) for
ingredient in ingredients]

    def calc_cost(self, portions=1) -> int:
        """Рассчитывает себестоимость порции."""
        total_cost = sum(ingredient.cost for ingredient in
self.ingredients)
        return total_cost * portions

    def calc_weight(self, portions=1, raw=True) -> int:
        """Рассчитывает общий вес ингредиентов. Можно выбрать
сырой или готовый вес."""
        total_weight = sum(ingredient.raw_weight if raw else
ingredient.weight for ingredient in self.ingredients)
        return total_weight * portions

    def __str__(self) -> str:
        ingredients_str = ", ".join(str(ingredient) for
ingredient in self.ingredients)

```

```

        return f"Рецепт: {self.name}\nИнгредиенты:
{ingredients_str}"

# Пример рецепта
if __name__ == '__main__':
    receipt_from_api = {
        "title": "Итальянская паста",
        "ingredients_list": [
            ('Мука', 100, 90, 15),
            ('Яйцо', 60, 50, 10),
            ('Оливковое масло', 20, 20, 30),
            ('Соль', 5, 5, 1),
            ('Перец', 2, 2, 2),
        ],
    }

    receipt = Receipt(receipt_from_api['title'],
receipt_from_api['ingredients_list'])

# Проверка методов
print(receipt) # Вывод рецепта и ингредиентов
print("Себестоимость (на 1 порцию):", receipt.calc_cost())
print("Себестоимость (на 3 порции):", receipt.calc_cost(3))
print("Вес сырого продукта:", receipt.calc_weight(raw=True))
print("Вес готового продукта:",
receipt.calc_weight(raw=False))

# Второе блюдо: Спагетти карбонара

receipt_from_api_2 = {
    "title": "Спагетти карбонара",
    "ingredients_list": [
        ('Спагетти', 200, 180, 50),
        ('Бекон', 100, 80, 60),

```



```

        ('Яйцо', 50, 45, 10),
        ('Пармезан', 30, 30, 40),
        ('Сливки', 50, 50, 20),
        ('Соль', 5, 5, 1),
        ('Черный перец', 2, 2, 2),
    ],
}

receipt_2 = Receipt(receipt_from_api_2['title'],
receipt_from_api_2['ingredients_list'])

# Проверка методов
print(receipt_2) # Вывод рецепта и ингредиентов
print("Себестоимость (на 1 порцию):", receipt_2.calc_cost())
print("Себестоимость (на 3 порции):", receipt_2.calc_cost(3))
print("Вес сырого продукта:",
receipt_2.calc_weight(raw=True))
print("Вес готового продукта:",
receipt_2.calc_weight(raw=False))

```

#### Листинг 4. Код файла test\_recipe.py

```

import unittest
from dish import Ingredient, Receipt

class TestIngredientAndReceipt(unittest.TestCase):

    @classmethod
    def setUpClass(cls):
        print("Запуск общего setup для класса
TestIngredientAndReceipt")

    @classmethod
    def tearDownClass(cls):
        print("Очистка после всех тестов
TestIngredientAndReceipt")

```

```

def setUp(self):
    """Инициализация ингредиентов и рецепта для тестов"""
    self.ingredients_list_1 = [
        ('Мука', 100, 90, 15),
        ('Яйцо', 60, 50, 10),
        ('Оливковое масло', 20, 20, 30),
        ('Соль', 5, 5, 1),
        ('Перец', 2, 2, 2),
    ]
    self.ingredients_list_2 = [
        ('Спагетти', 200, 180, 50),
        ('Бекон', 100, 80, 60),
        ('Яйцо', 50, 45, 10),
        ('Пармезан', 30, 30, 40),
        ('Сливки', 50, 50, 20),
        ('Соль', 5, 5, 1),
        ('Черный перец', 2, 2, 2),
    ]
    self.recipe_1 = Recipe("Итальянская паста",
self.ingredients_list_1)
    self.recipe_2 = Recipe("Спагетти карбонара",
self.ingredients_list_2)

def tearDown(self):
    print("Завершение теста")

def test_calc_cost_recipe_1(self):
    """Тест расчета себестоимости Итальянской пасты на одну и
несколько порций."""
    self.assertEqual(self.recipe_1.calc_cost(), 58) #
15+10+30+1+2 = 58
    self.assertEqual(self.recipe_1.calc_cost(3), 174)

def test_calc_cost_recipe_2(self):

```

```

        """Тест расчета себестоимости Спагетти карбонара на одну
и несколько порций."""
        self.assertEqual(self.recipe_2.calc_cost(), 183) #
Проверка расчета
        self.assertEqual(self.recipe_2.calc_cost(2), 366)

    def test_calc_weight_raw_recipe_1(self):
        """Тест расчета общего сырого веса Итальянской пасты."""
        self.assertEqual(self.recipe_1.calc_weight(raw=True),
187) # 100+60+20+5+2 = 187

    def test_calc_weight_raw_recipe_2(self):
        """Тест расчета общего сырого веса Спагетти карбонара."""
        self.assertEqual(self.recipe_2.calc_weight(raw=True),
437) # Проверка веса

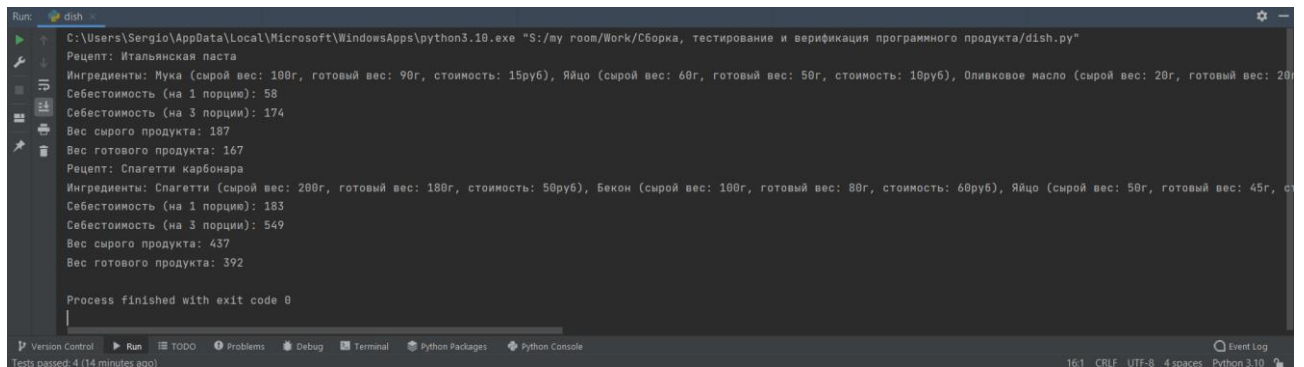
    def test_calc_weight_cooked_recipe_1(self):
        """Тест расчета общего веса готового продукта Итальянской
пасты."""
        self.assertEqual(self.recipe_1.calc_weight(raw=False),
167) # 90+50+20+5+2 = 167

    def test_calc_weight_cooked_recipe_2(self):
        """Тест расчета общего веса готового продукта Спагетти
карбонара."""
        self.assertEqual(self.recipe_2.calc_weight(raw=False),
392) # Проверка веса

    def test_invalid_ingredient(self):
        """Тест обработки неверных данных ингредиента."""
        with self.assertRaises(ValueError):
            Ingredient("", -100, -90, -10)

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

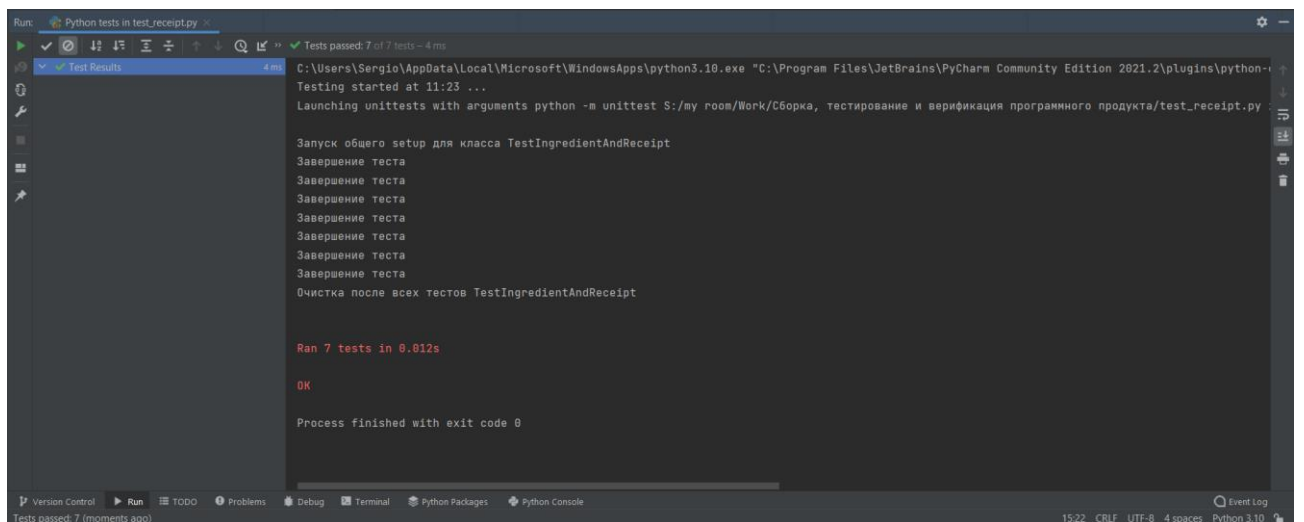
```



```
Run: dish.py
C:\Users\Sergio\AppData\Local\Microsoft\WindowsApps\python3.10.exe "S:/my room/Work/Сборка, тестирование и верификация программного продукта/dish.py"
Рецепт: Итальянская паста
Ингредиенты: Мука (сырой вес: 100г, готовый вес: 90г, стоимость: 15руб), Яйцо (сырой вес: 60г, готовый вес: 50г, стоимость: 10руб), Оливковое масло (сырой вес: 20г, готовый вес: 20г, стоимость: 10руб)
Себестоимость (на 1 порции): 58
Себестоимость (на 3 порции): 174
Вес сырого продукта: 187
Вес готового продукта: 167
Рецепт: Спагетти карбонара
Ингредиенты: Спагетти (сырой вес: 200г, готовый вес: 180г, стоимость: 50руб), Бекон (сырой вес: 100г, готовый вес: 80г, стоимость: 60руб), Яйцо (сырой вес: 50г, готовый вес: 45г, стоимость: 9руб)
Себестоимость (на 1 порции): 183
Себестоимость (на 3 порции): 549
Вес сырого продукта: 437
Вес готового продукта: 392

Process finished with exit code 0
```

Рисунок 3 – Вывод файла dish.py



```
Run: Python tests in test_recipe.py
Tests passed: 7 of 7 tests - 4 ms
Test Results
C:\Users\Sergio\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Program Files\JetBrains\PyCharm Community Edition 2021.2\plugins\python-test\python-test.py"
Testing started at 11:23 ...
Launching unittests with arguments python -m unittest S:/my room/Work/Сборка, тестирование и верификация программного продукта/test_recipe.py

Запуск общего setup для класса TestIngredientAndReceipt
Завершение теста
Завершение теста
Завершение теста
Завершение теста
Завершение теста
Завершение теста
Завершение теста
Очистка после всех тестов TestIngredientAndReceipt

Ran 7 tests in 0.012s

OK

Process finished with exit code 0
```

Рисунок 4 – Вывод файла test\_recipe.py