|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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.

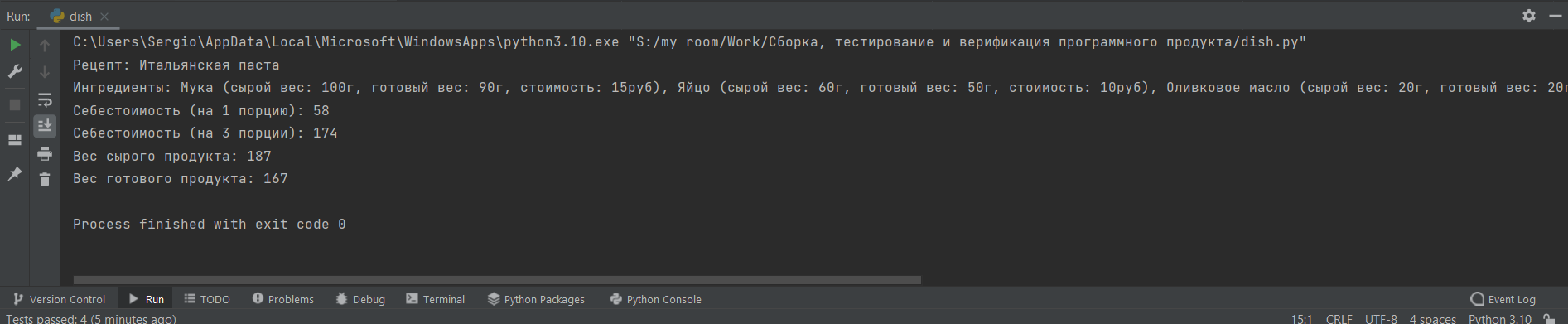


Рисунок 1. Вывод проверки методов файла dish.py

Далее создался файл test\_receipt.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\_receipt.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.receipt = 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.receipt.calc\_cost(), 58) # 15+10+30+1+2 = 58  self.assertEqual(self.receipt.calc\_cost(3), 174)   def test\_calc\_weight\_raw(self):  *"""Тест расчета общего сырого веса."""* self.assertEqual(self.receipt.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.receipt.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.

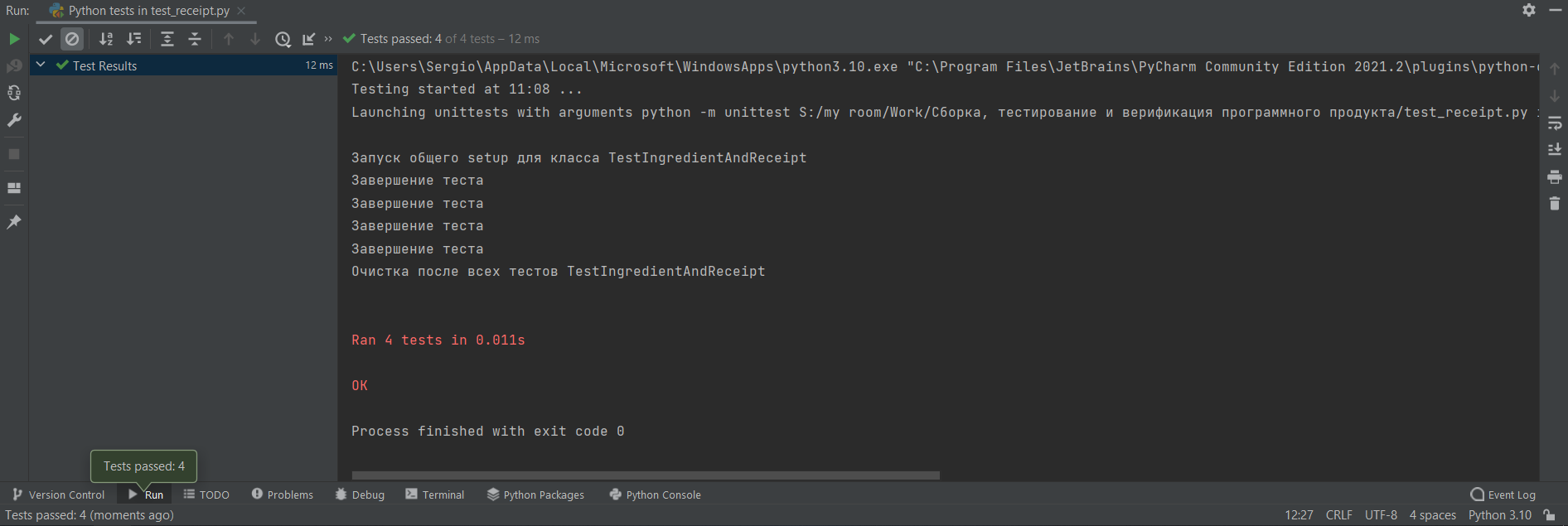


Рисунок 2 – Итог проведения тестов

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\_receipt.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.receipt\_1 = Receipt("Итальянская паста", self.ingredients\_list\_1)  self.receipt\_2 = Receipt("Спагетти карбонара", self.ingredients\_list\_2)   def tearDown(self):  print("Завершение теста")   def test\_calc\_cost\_receipt\_1(self):  *"""Тест расчета себестоимости Итальянской пасты на одну и несколько порций."""* self.assertEqual(self.receipt\_1.calc\_cost(), 58) # 15+10+30+1+2 = 58  self.assertEqual(self.receipt\_1.calc\_cost(3), 174)   def test\_calc\_cost\_receipt\_2(self):  *"""Тест расчета себестоимости Спагетти карбонара на одну и несколько порций."""* self.assertEqual(self.receipt\_2.calc\_cost(), 183) # Проверка расчета  self.assertEqual(self.receipt\_2.calc\_cost(2), 366)   def test\_calc\_weight\_raw\_receipt\_1(self):  *"""Тест расчета общего сырого веса Итальянской пасты."""* self.assertEqual(self.receipt\_1.calc\_weight(raw=True), 187) # 100+60+20+5+2 = 187   def test\_calc\_weight\_raw\_receipt\_2(self):  *"""Тест расчета общего сырого веса Спагетти карбонара."""* self.assertEqual(self.receipt\_2.calc\_weight(raw=True), 437) # Проверка веса   def test\_calc\_weight\_cooked\_receipt\_1(self):  *"""Тест расчета общего веса готового продукта Итальянской пасты."""* self.assertEqual(self.receipt\_1.calc\_weight(raw=False), 167) # 90+50+20+5+2 = 167   def test\_calc\_weight\_cooked\_receipt\_2(self):  *"""Тест расчета общего веса готового продукта Спагетти карбонара."""* self.assertEqual(self.receipt\_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() |

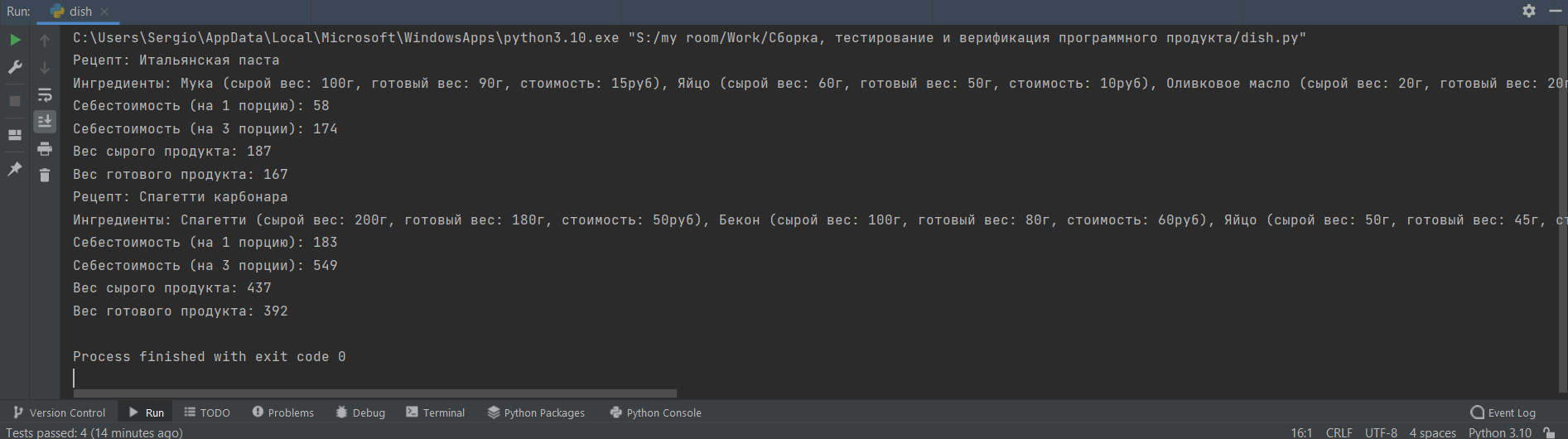


Рисунок 3 – Вывод файла dish.py

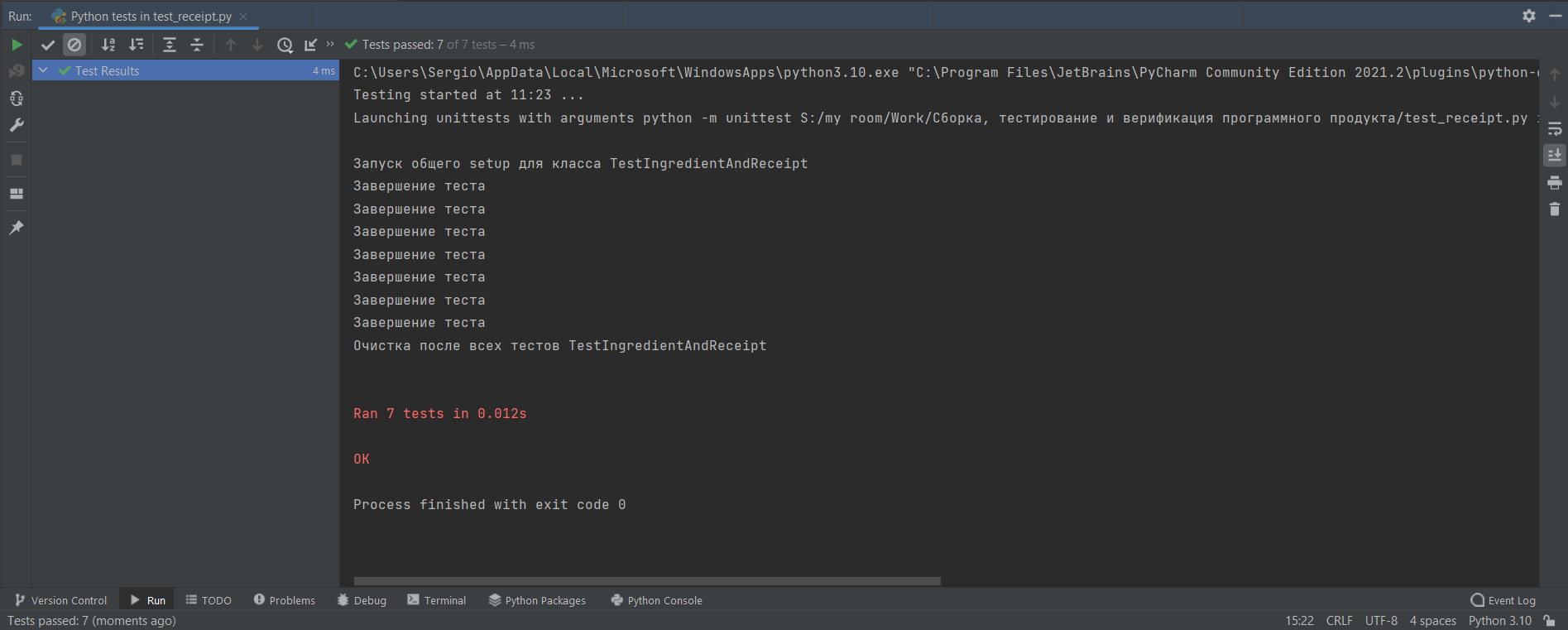


Рисунок 4 – Вывод файла test\_receipt.py