



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

**ОТЧЕТ**  
**По лабораторной работе №1**  
**По дисциплине «Парадигмы и конструкции**  
**языков программирования»**  
**«Вариант 9»**

Выполнил студент: Денефельд Валерия Максимовна  
*фамилия, имя, отчество*

Группа: ИБМЗ-34Б

Преподаватель

Гапанюк Ю.Е.

*подпись, дата*

2024 г.

## Основной файл с функциями и классами:

```
from operator import itemgetter

class OperatingSystem:
    """Класс Операционная система"""
    def __init__(self, id, name):
        self.id = id
        self.name = name

class Computer:
    """Класс Компьютер"""
    def __init__(self, id, model, os_id):
        self.id = id
        self.model = model
        self.os_id = os_id

class ComputerOS:
    """
    Класс для связи многие-ко-многим между компьютерами и операционными
    системами
    """
    def __init__(self, os_id, computer_id):
        self.os_id = os_id
        self.computer_id = computer_id

def one_to_many_mapping(computers, operating_systems):
    return [(c.model, o.name)
            for o in operating_systems
            for c in computers
            if c.os_id == o.id]

def many_to_many_mapping(computers, operating_systems, computers_os):
    many_to_many_temp = [(o.name, co.os_id, co.computer_id)
                          for o in operating_systems
                          for co in computers_os
                          if o.id == co.os_id]

    return [(c.model, os_name)
            for os_name, os_id, computer_id in many_to_many_temp
            for c in computers
            if c.id == computer_id]

def task_1(one_to_many):
    """Все операционные системы и связанные с ними компьютеры,
    отсортированные по ОС"""
    return sorted(one_to_many, key=itemgetter(1))

def task_2(one_to_many, operating_systems):
    """Количество компьютеров для каждой операционной системы"""
    res = []
    for o in operating_systems:
        # Список компьютеров для данной ОС
        o_computers = list(filter(lambda i: i[1] == o.name, one_to_many))
        # Количество компьютеров
        res.append((o.name, len(o_computers)))
```

```
    return sorted(res, key=itemgetter(1), reverse=True)

def task_3(many_to_many, operating_systems):
    """Все операционные системы, содержащие слово 'Windows', и их
    компьютеры"""
    res = {}
    for o in operating_systems:
        if "Windows" in o.name:
            o_computers = list(filter(lambda i: i[1] == o.name,
many_to_many))
            o_computers_names = [x[0] for x in o_computers]
            res[o.name] = o_computers_names
    return res
```

Тест №1: Проверяет функцию task\_1, которая возвращает список всех операционных систем и связанных с ними компьютеров, отсортированных по названию ОС.

```
import unittest
from program import OperatingSystem, Computer, one_to_many_mapping, task_1

class TestTask1(unittest.TestCase):
    def setUp(self):
        self.operating_systems = [
            OperatingSystem(1, "Windows 10"),
            OperatingSystem(2, "Ubuntu"),
            OperatingSystem(3, "macOS"),
        ]

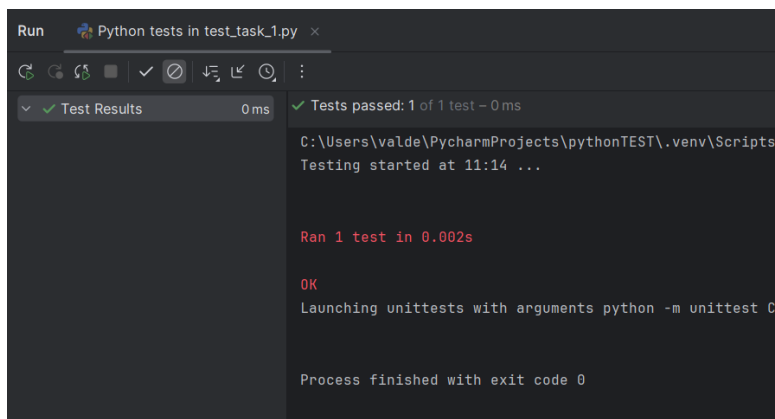
        self.computers = [
            Computer(1, "Dell XPS", 1),
            Computer(2, "MacBook Pro", 3),
            Computer(3, "HP Spectre", 1),
            Computer(4, "ThinkPad", 2),
            Computer(5, "Mac Mini", 3),
        ]

        self.one_to_many = one_to_many_mapping(self.computers,
self.operating_systems)

    def test_task_1(self):
        expected = [
            ("ThinkPad", "Ubuntu"),
            ("Dell XPS", "Windows 10"),
            ("HP Spectre", "Windows 10"),
            ("MacBook Pro", "macOS"),
            ("Mac Mini", "macOS"),
        ]
        result = task_1(self.one_to_many)
        self.assertEqual(result, expected)

if __name__ == "__main__":
    unittest.main()
```

Тест проведен:



Тест №2: Проверяет функцию task\_2, которая подсчитывает количество компьютеров, связанных с каждой операционной системой, и сортирует их по убыванию количества.

```
import unittest
from program import OperatingSystem, Computer, one_to_many_mapping, task_2

class TestTask2(unittest.TestCase):
    def setUp(self):
        self.operating_systems = [
            OperatingSystem(1, "Windows 10"),
            OperatingSystem(2, "Ubuntu"),
            OperatingSystem(3, "macOS"),
        ]

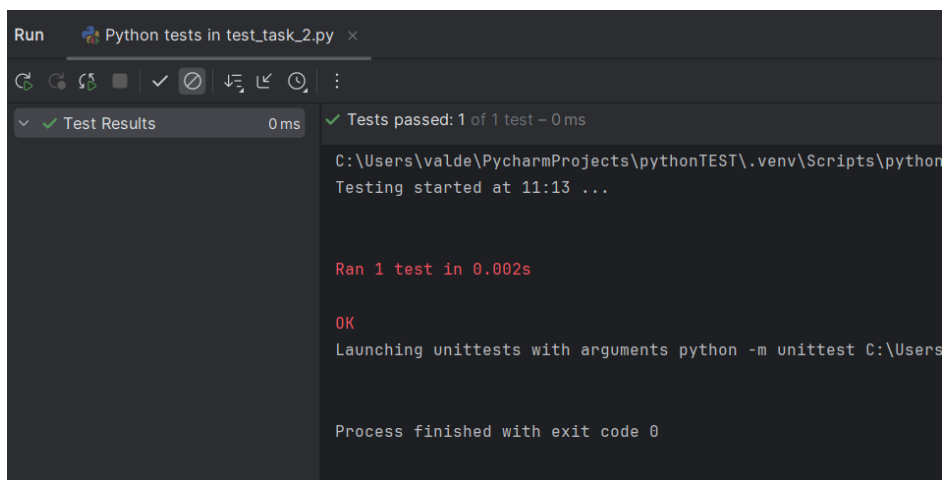
        self.computers = [
            Computer(1, "Dell XPS", 1),
            Computer(2, "MacBook Pro", 3),
            Computer(3, "HP Spectre", 1),
            Computer(4, "ThinkPad", 2),
            Computer(5, "Mac Mini", 3),
        ]

        self.one_to_many = one_to_many_mapping(self.computers,
self.operating_systems)

    def test_task_2(self):
        expected = [
            ("Windows 10", 2),
            ("macOS", 2),
            ("Ubuntu", 1),
        ]
        result = task_2(self.one_to_many, self.operating_systems)
        self.assertEqual(result, expected)

if __name__ == "__main__":
    unittest.main()
```

Тест проведен:



The screenshot shows the 'Run' window in PyCharm for a file named 'test\_task\_2.py'. The 'Test Results' tab is active, displaying a green checkmark and the text 'Tests passed: 1 of 1 test - 0 ms'. Below this, the test execution details are shown: 'C:\Users\valde\PycharmProjects\pythonTEST\.venv\Scripts\python', 'Testing started at 11:13 ...', 'Ran 1 test in 0.002s', 'OK', 'Launching unittests with arguments python -m unittest C:\Users\valde\PycharmProjects\pythonTEST\test\_task\_2.py', and 'Process finished with exit code 0'.

Тест №3: Проверяет функцию task\_3, которая фильтрует операционные системы, содержащие слово "Windows", и возвращает список связанных с ними компьютеров.

```
import unittest
from program import OperatingSystem, Computer, ComputerOS,
many_to_many_mapping, task_3

class TestTask3(unittest.TestCase):
    def setUp(self):
        self.operating_systems = [
            OperatingSystem(1, "Windows 10"),
            OperatingSystem(2, "Ubuntu"),
            OperatingSystem(3, "macOS"),
        ]

        self.computers = [
            Computer(1, "Dell XPS", 1),
            Computer(2, "MacBook Pro", 3),
            Computer(3, "HP Spectre", 1),
            Computer(4, "ThinkPad", 2),
            Computer(5, "Mac Mini", 3),
        ]

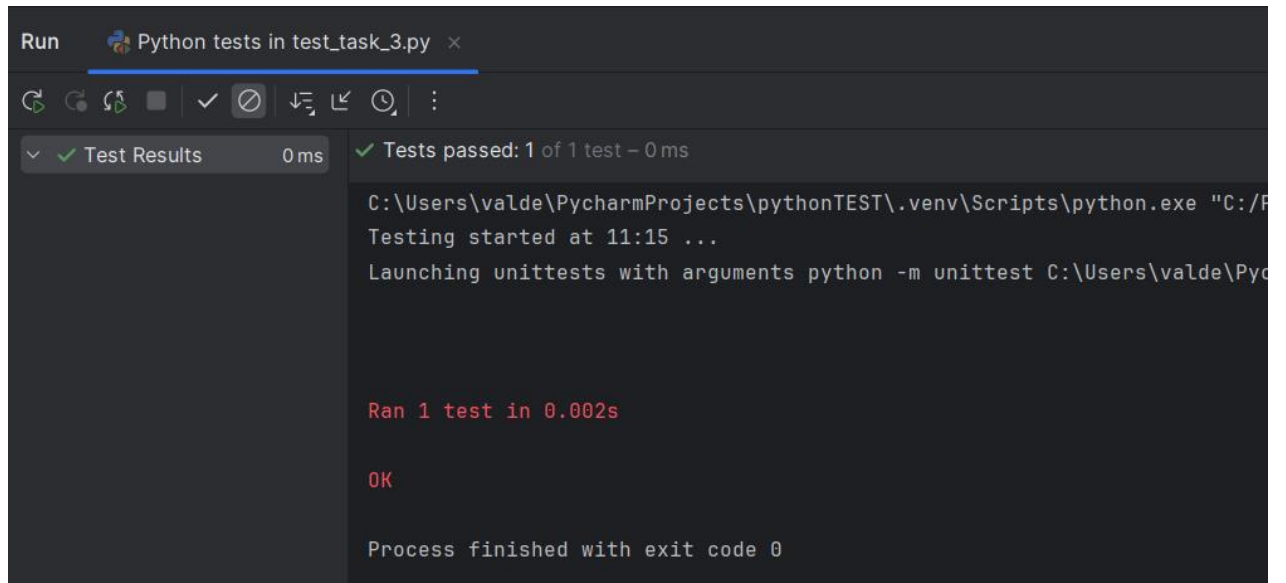
        self.computers_os = [
            ComputerOS(1, 1),
            ComputerOS(1, 3),
            ComputerOS(2, 4),
            ComputerOS(3, 2),
            ComputerOS(3, 5),
        ]

        self.many_to_many = many_to_many_mapping(self.computers,
self.operating_systems, self.computers_os)

    def test_task_3(self):
        expected = {
            "Windows 10": ["Dell XPS", "HP Spectre"]
        }
        result = task_3(self.many_to_many, self.operating_systems)
        self.assertEqual(result, expected)

if __name__ == "__main__":
    unittest.main()
```

Тест проведен:



The screenshot shows the 'Run' window in PyCharm. The title bar indicates 'Python tests in test\_task\_3.py'. The toolbar contains icons for running, debugging, and other actions. The 'Test Results' tab is active, showing a summary of 'Tests passed: 1 of 1 test - 0 ms'. The console output displays the command used to run the tests, the start time, and the successful completion of the test.

```
Run Python tests in test_task_3.py x
C:\Users\valde\PycharmProjects\pythonTEST\.venv\Scripts\python.exe "C:/P
Testing started at 11:15 ...
Launching unittests with arguments python -m unittest C:\Users\valde\Pyd

Ran 1 test in 0.002s

OK

Process finished with exit code 0
```