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 рубежному контролю №2

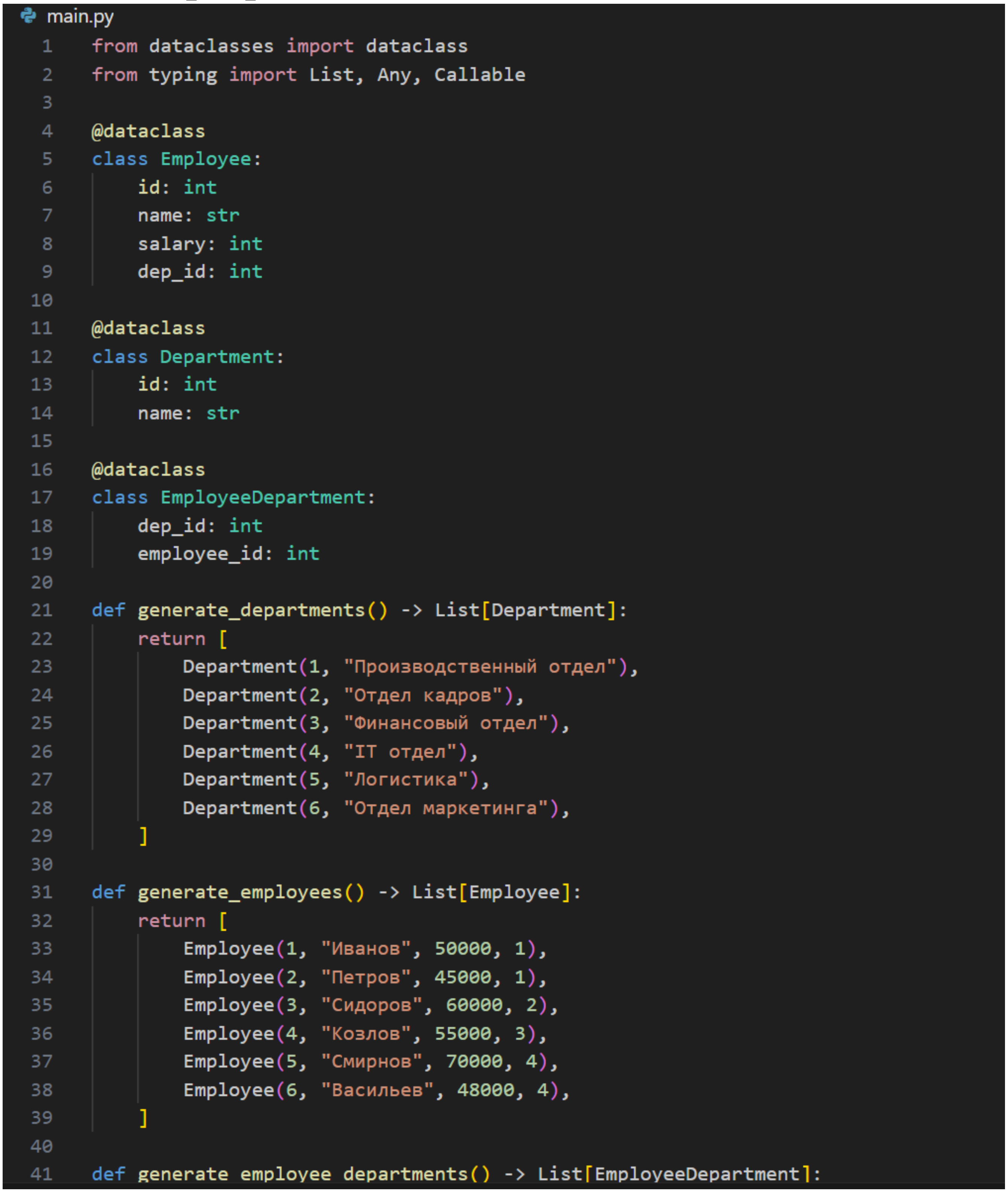
«**Модульное тестирование**»

Выполнил:
Студент группы ИУ5-31Б
Чехович Юрий

Проверил:
Гапанюк Ю. Е.

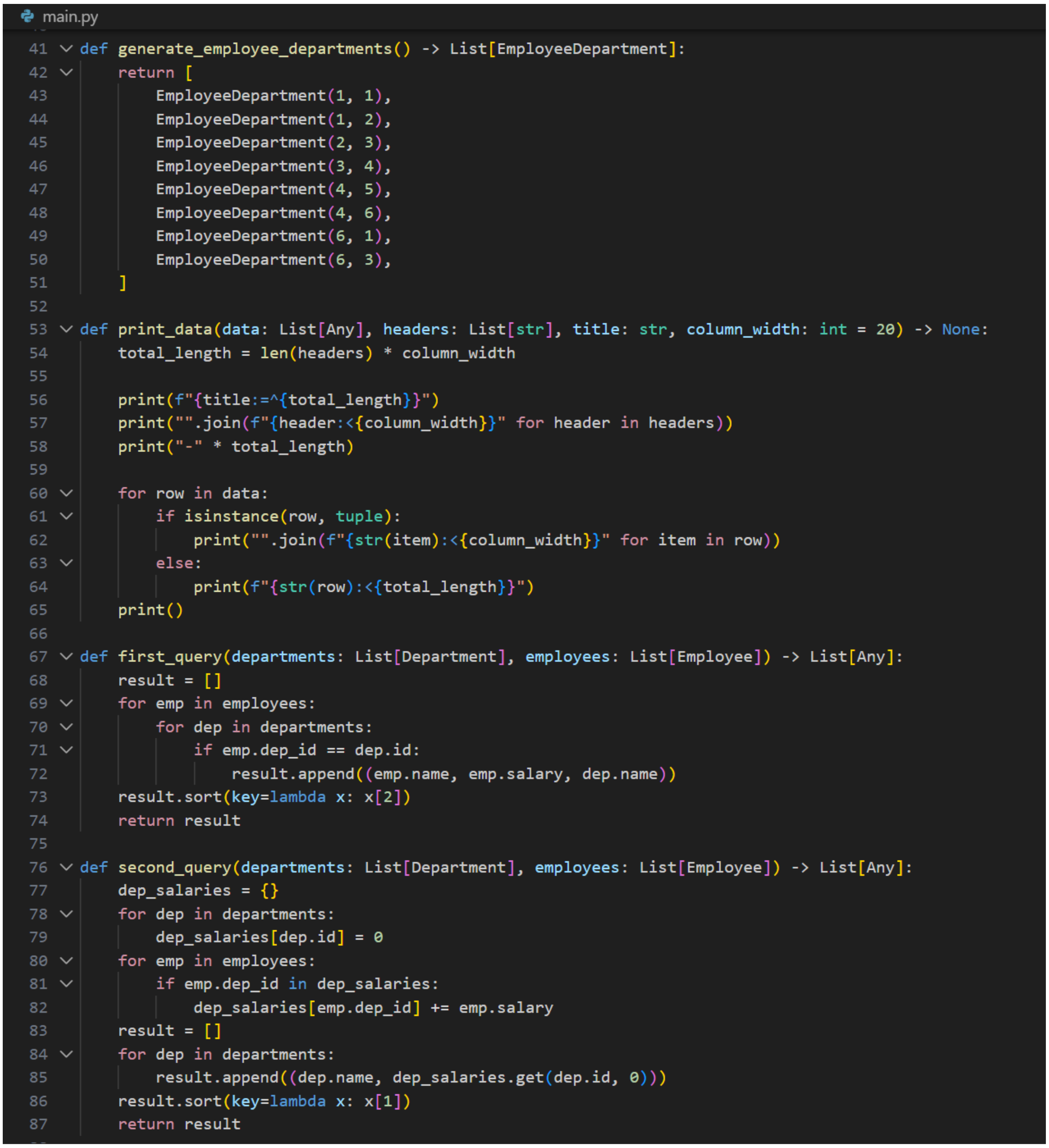
2025 г.

Листинг программы



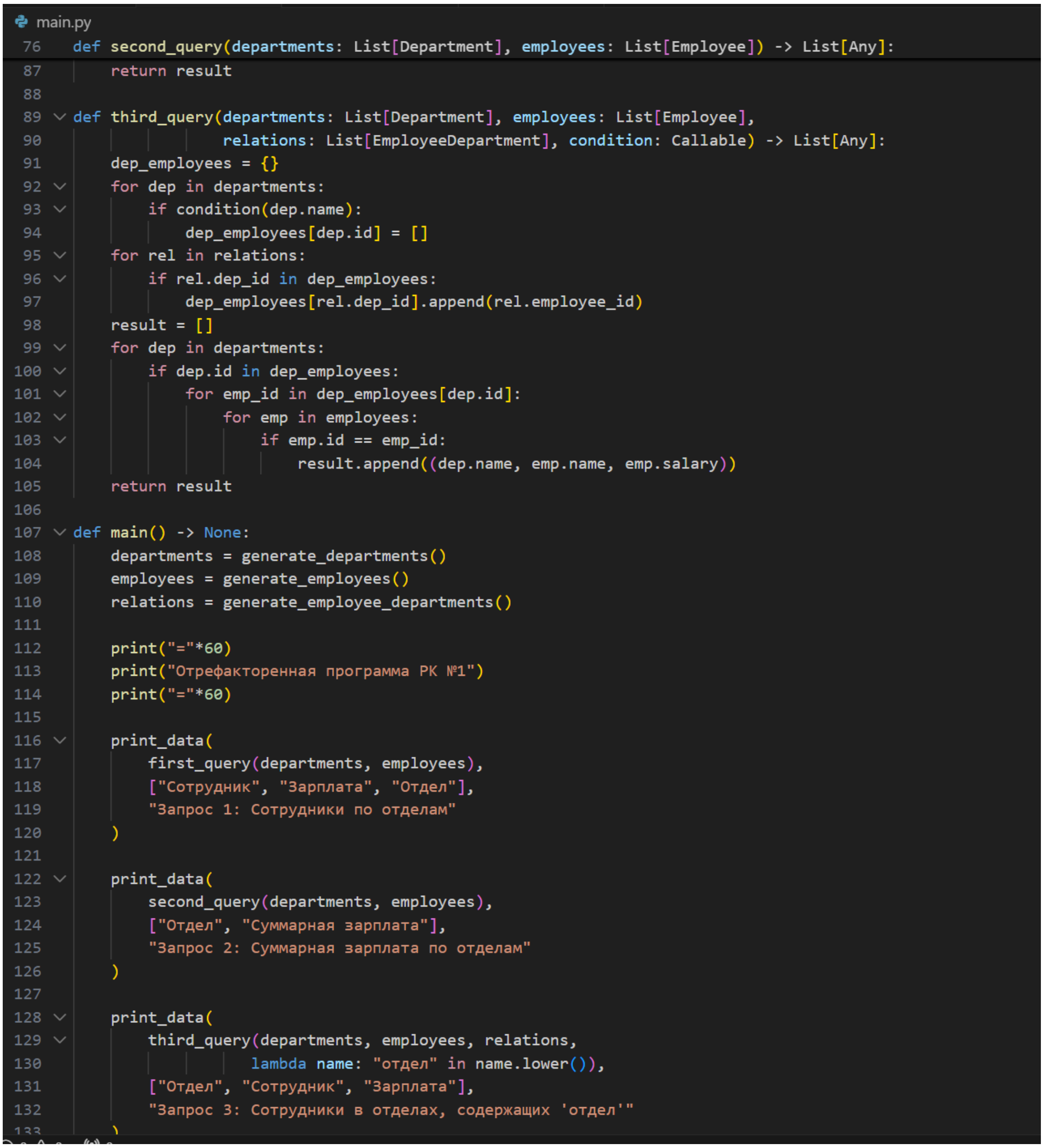
main.py

```python
from dataclasses import dataclass
from typing import List, Any, Callable

@dataclass
class Employee:
    id: int
    name: str
    salary: int
    dep_id: int

@dataclass
class Department:
    id: int
    name: str

@dataclass
class EmployeeDepartment:
    dep_id: int
    employee_id: int

def generate_departments() -> List[Department]:
    return [
        Department(1, "Производственный отдел"),
        Department(2, "Отдел кадров"),
        Department(3, "Финансовый отдел"),
        Department(4, "IT отдел"),
        Department(5, "Логистика"),
        Department(6, "Отдел маркетинга"),
    ]

def generate_employees() -> List[Employee]:
    return [
        Employee(1, "Иванов", 50000, 1),
        Employee(2, "Петров", 45000, 1),
        Employee(3, "Сидоров", 60000, 2),
        Employee(4, "Козлов", 55000, 3),
        Employee(5, "Смирнов", 70000, 4),
        Employee(6, "Васильев", 48000, 4),
    ]

def generate_employee_departments() -> List[EmployeeDepartment]:
```



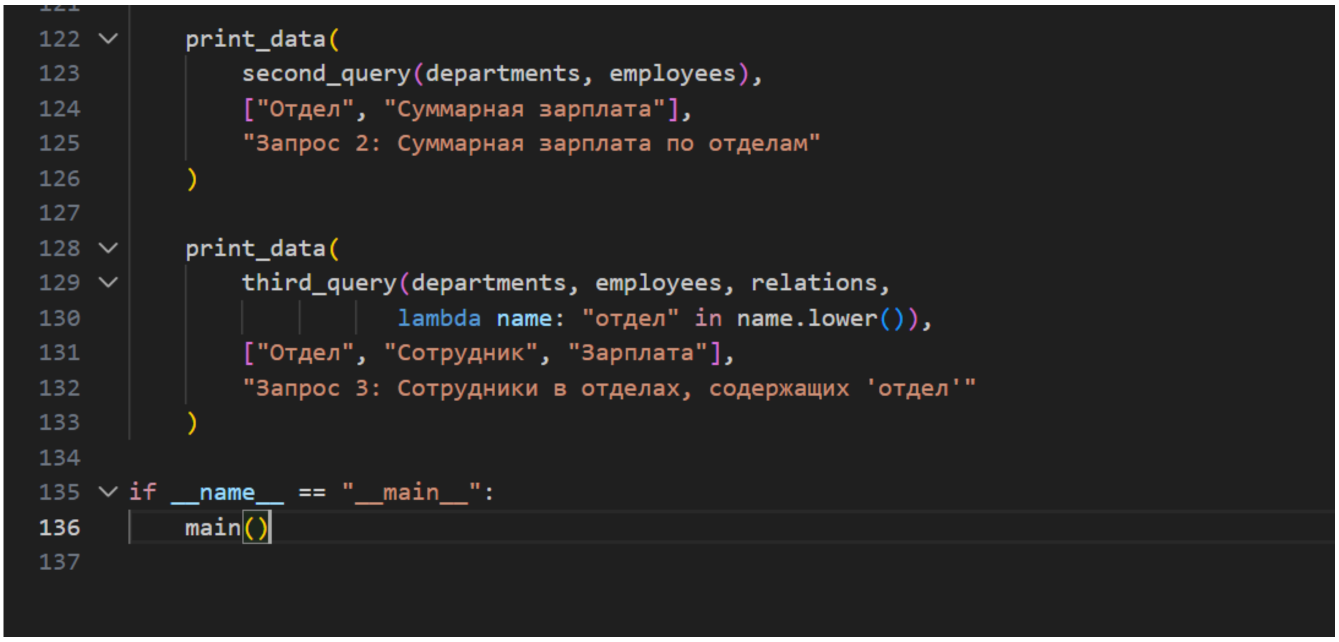
main.py

```python
def generate_employee_departments() -> List[EmployeeDepartment]:
    return [
        EmployeeDepartment(1, 1),
        EmployeeDepartment(1, 2),
        EmployeeDepartment(2, 3),
        EmployeeDepartment(3, 4),
        EmployeeDepartment(4, 5),
        EmployeeDepartment(4, 6),
        EmployeeDepartment(6, 1),
        EmployeeDepartment(6, 3),
    ]

def print_data(data: List[Any], headers: List[str], title: str, column_width: int = 20) -> None:
    total_length = len(headers) * column_width

    print(f"{title:=^{total_length}}")
    print("".join(f"{header:<{column_width}}" for header in headers))
    print("-" * total_length)

    for row in data:
        if isinstance(row, tuple):
            print("".join(f"{str(item):<{column_width}}" for item in row))
        else:
            print(f"{str(row):<{total_length}}")
    print()

def first_query(departments: List[Department], employees: List[Employee]) -> List[Any]:
    result = []
    for emp in employees:
        for dep in departments:
            if emp.dep_id == dep.id:
                result.append((emp.name, emp.salary, dep.name))
    result.sort(key=lambda x: x[2])
    return result

def second_query(departments: List[Department], employees: List[Employee]) -> List[Any]:
    dep_salaries = {}
    for dep in departments:
        dep_salaries[dep.id] = 0
    for emp in employees:
        if emp.dep_id in dep_salaries:
            dep_salaries[emp.dep_id] += emp.salary
    result = []
    for dep in departments:
        result.append((dep.name, dep_salaries.get(dep.id, 0)))
    result.sort(key=lambda x: x[1])
    return result
```

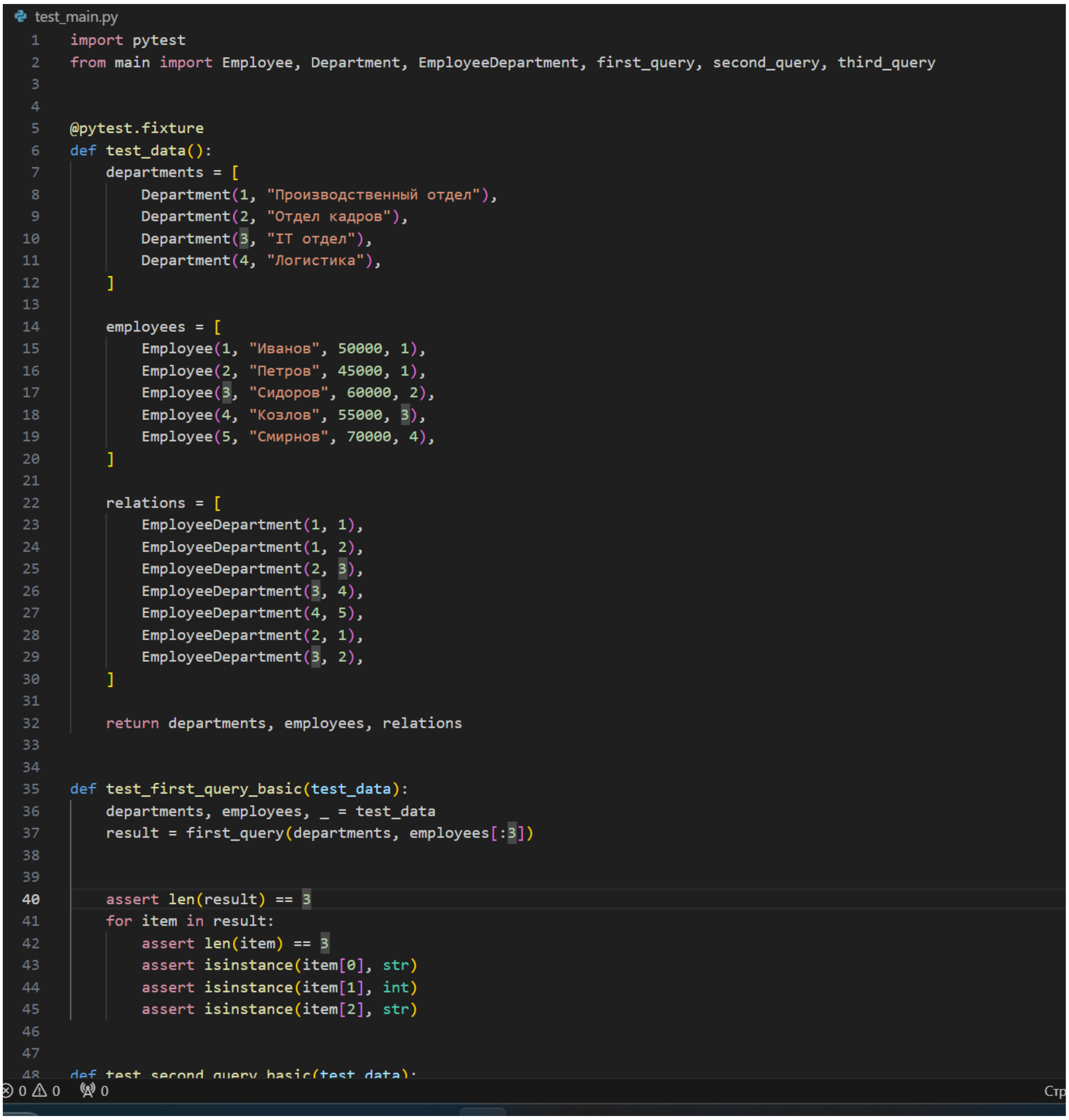


main.py

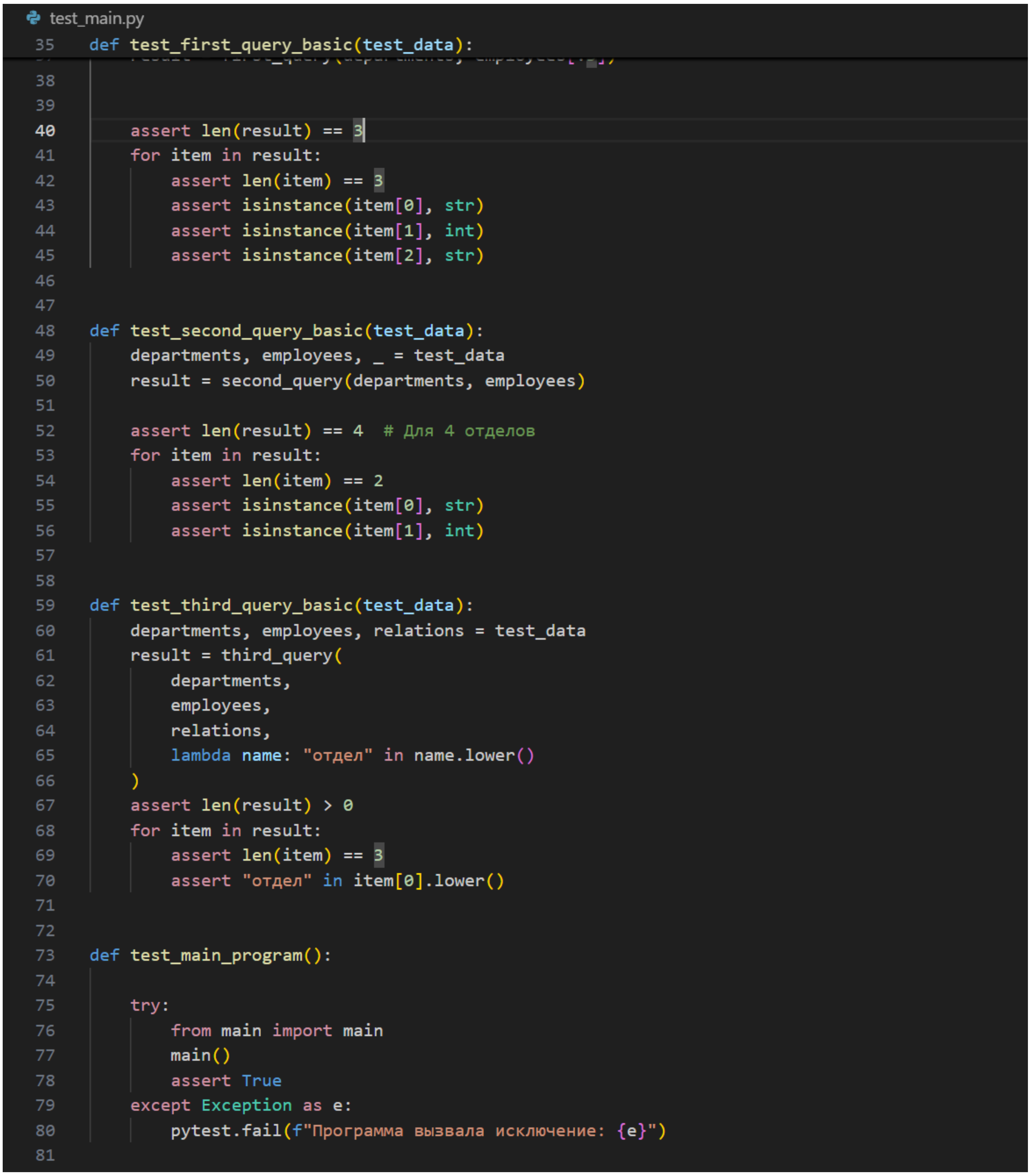
```python
def second_query(departments: List[Department], employees: List[Employee]) -> List[Any]:
    return result

def third_query(departments: List[Department], employees: List[Employee],
                relations: List[EmployeeDepartment], condition: Callable) -> List[Any]:
    dep_employees = {}
    for dep in departments:
        if condition(dep.name):
            dep_employees[dep.id] = []
    for rel in relations:
        if rel.dep_id in dep_employees:
            dep_employees[rel.dep_id].append(rel.employee_id)
    result = []
    for dep in departments:
        if dep.id in dep_employees:
            for emp_id in dep_employees[dep.id]:
                for emp in employees:
                    if emp.id == emp_id:
                        result.append((dep.name, emp.name, emp.salary))
    return result

def main() -> None:
    departments = generate_departments()
    employees = generate_employees()
    relations = generate_employee_departments()

    print("="*60)
    print("Отрефакторенная программа РК №1")
    print("="*60)

    print_data(
        first_query(departments, employees),
        ["Сотрудник", "Зарплата", "Отдел"],
        "Запрос 1: Сотрудники по отделам"
    )

    print_data(
        second_query(departments, employees),
        ["Отдел", "Суммарная зарплата"],
        "Запрос 2: Суммарная зарплата по отделам"
    )

    print_data(
        third_query(departments, employees, relations,
                    lambda name: "отдел" in name.lower()),
        ["Отдел", "Сотрудник", "Зарплата"],
        "Запрос 3: Сотрудники в отделах, содержащих 'отдел'"
    )
```



```
    print_data(
        second_query(departments, employees),
        ["Отдел", "Суммарная зарплата"],
        "Запрос 2: Суммарная зарплата по отделам"
    )

    print_data(
        third_query(departments, employees, relations,
                    lambda name: "отдел" in name.lower()),
        ["Отдел", "Сотрудник", "Зарплата"],
        "Запрос 3: Сотрудники в отделах, содержащих 'отдел'"
    )

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

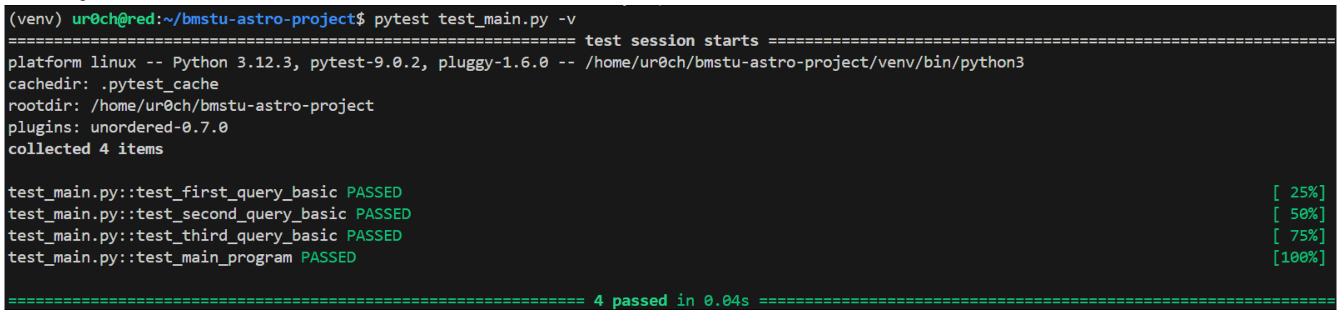


```python
import pytest
from main import Employee, Department, EmployeeDepartment, first_query, second_query, third_query


@pytest.fixture
def test_data():
    departments = [
        Department(1, "Производственный отдел"),
        Department(2, "Отдел кадров"),
        Department(3, "IT отдел"),
        Department(4, "Логистика"),
    ]

    employees = [
        Employee(1, "Иванов", 50000, 1),
        Employee(2, "Петров", 45000, 1),
        Employee(3, "Сидоров", 60000, 2),
        Employee(4, "Козлов", 55000, 3),
        Employee(5, "Смирнов", 70000, 4),
    ]

    relations = [
        EmployeeDepartment(1, 1),
        EmployeeDepartment(1, 2),
        EmployeeDepartment(2, 3),
        EmployeeDepartment(3, 4),
        EmployeeDepartment(4, 5),
        EmployeeDepartment(2, 1),
        EmployeeDepartment(3, 2),
    ]

    return departments, employees, relations


def test_first_query_basic(test_data):
    departments, employees, _ = test_data
    result = first_query(departments, employees[:3])


    assert len(result) == 3
    for item in result:
        assert len(item) == 3
        assert isinstance(item[0], str)
        assert isinstance(item[1], int)
        assert isinstance(item[2], str)


def test_second_query_basic(test_data):
```



```python
test_main.py
def test_first_query_basic(test_data):


    assert len(result) == 3
    for item in result:
        assert len(item) == 3
        assert isinstance(item[0], str)
        assert isinstance(item[1], int)
        assert isinstance(item[2], str)


def test_second_query_basic(test_data):
    departments, employees, _ = test_data
    result = second_query(departments, employees)

    assert len(result) == 4  # Для 4 отделов
    for item in result:
        assert len(item) == 2
        assert isinstance(item[0], str)
        assert isinstance(item[1], int)


def test_third_query_basic(test_data):
    departments, employees, relations = test_data
    result = third_query(
        departments,
        employees,
        relations,
        lambda name: "отдел" in name.lower()
    )
    assert len(result) > 0
    for item in result:
        assert len(item) == 3
        assert "отдел" in item[0].lower()


def test_main_program():

    try:
        from main import main
        main()
        assert True
    except Exception as e:
        pytest.fail(f"Программа вызвала исключение: {e}")

```

Результат выполнения



```
(venv) ur0ch@red:~/bmstu-astro-project$ pytest test_main.py -v
============================================================= test session starts ==============================================================
platform linux -- Python 3.12.3, pytest-9.0.2, pluggy-1.6.0 -- /home/ur0ch/bmstu-astro-project/venv/bin/python3
cachedir: .pytest_cache
rootdir: /home/ur0ch/bmstu-astro-project
plugins: unordered-0.7.0
collected 4 items

test_main.py::test_first_query_basic PASSED                                                                                              [ 25%]
test_main.py::test_second_query_basic PASSED                                                                                             [ 50%]
test_main.py::test_third_query_basic PASSED                                                                                              [ 75%]
test_main.py::test_main_program PASSED                                                                                                   [100%]

============================================================== 4 passed in 0.04s ===============================================================
```

==

Отрефакторенная РК1

```
Запрос 1: Сотрудники по отделам
Сотрудник          Зарплата          Отдел
------------------------------------------------------------
Смирнов            70000             IT отдел
Васильев           48000             IT отдел
Сидоров            60000             Отдел кадров
Иванов             50000             Производственный отдел
Петров             45000             Производственный отдел
Козлов             55000             Финансовый отдел

Запрос 2: Суммарная зарплата по отделам=
Отдел              Суммарная зарплата
----------------------------------------
Логистика          0
Отдел маркетинга   0
Финансовый отдел   55000
Отдел кадров       60000
Производственный отдел95000
IT отдел           118000

Запрос 3: Сотрудники в отделах, содержащих 'отдел'=====
Отдел              Сотрудник          Зарплата
------------------------------------------------------------
Производственный отделИванов             50000
Производственный отделПетров             45000
Отдел кадров       Сидоров            60000
Финансовый отдел   Козлов             55000
IT отдел           Смирнов            70000
IT отдел           Васильев           48000
Отдел маркетинга   Иванов             50000
Отдел маркетинга   Сидоров            60000
```