Текст программы

- main.py

```
from operator import itemgetter


class Student:
    def __init__(self, student_id, fio, gpa, class_id):
        self.student_id = student_id
        self.fio = fio
        self.gpa = gpa
        self.class_id = class_id


class Class:
    def __init__(self, student_id, name):
        self.student_id = student_id
        self.name = name


class ClassStudents:
    def __init__(self, class_id, pupil_id):
        self.class_id = class_id
        self.pupil_id = pupil_id


classes = [
    Class(1, '9 мат'),
    Class(2, '10 мат'),
    Class(3, '11 мат'),

    Class(11, '9 гум'),
    Class(22, '10 гум'),
    Class(33, '11 гум'),
]

students = [
    Student(1, 'Алексеев', 4.0, 1),
    Student(2, 'Борисов', 4.8, 2),
    Student(3, 'Иванов', 5.0, 3),
    Student(4, 'Петров', 4.4, 3),
    Student(5, 'Сидоров', 3.9, 3),
    Student(6, 'Антонов', 4.0, 3),
]

class_students = [
    ClassStudents(1, 1),
    ClassStudents(2, 2),
    ClassStudents(3, 3),
    ClassStudents(3, 4),
    ClassStudents(3, 5),
```

```
    ClassStudents(3, 6),

    ClassStudents(11, 1),
    ClassStudents(22, 2),
    ClassStudents(33, 3),
    ClassStudents(33, 4),
    ClassStudents(33, 5),
    ClassStudents(33, 6),
]


def otm_function(one_list):
    in_list = [(s.fio, s.gpa, c.name)
               for c in classes
               for s in students
               if s.class_id == c.student_id]
    one_list[:] = in_list
    return one_list


def mtm_function(many_list):
    in_list_temp = [(c.name, cs.class_id, cs.pupil_id)
                    for c in classes
                    for cs in class_students
                    if c.student_id == cs.class_id]

    in_list = [(s.fio, s.gpa, class_name)
               for class_name, class_id, pupil_id in in_list_temp
               for s in students if s.student_id == pupil_id]
    many_list[:] = in_list
    return many_list


def set_a(many_to_many, res_3):
    for s in students:
        if s.fio.startswith('A'):
            # Список учеников
            s_students = list(filter(lambda i: i[0] == s.fio, many_to_many))
            # Класс ученика
            s_students_fios = [x for _, _, x in s_students]
            # Добавляем результат в словарь
            # ключ - ученик, значение - класс ученика
            res_3[s.fio] = s_students_fios
    return res_3


def main():

    print('Задание E1')
    one_to_many = []
    otm_function(one_to_many)
```

```
    res_1 = {}
    # Перебираем все классы, для поиска классов, содержащих 'мат'
    for c in classes:
        if 'мат' in c.name:
            # Список учеников класса
            c_classes = list(filter(lambda i: i[2] == c.name, one_to_many))
            # Только ФИО учеников
            c_classes_names = [x for x, _, _ in c_classes]
            # Добавляем результат в словарь
            # ключ - класс, значение - список фамилий
            res_1[c.name] = c_classes_names
    print(res_1)

    print('\nЗадание E2')
    res_2_unsorted = []
    # Перебираем все классы
    for c in classes:
        # Список учеников класса
        c_classes = list(filter(lambda i: i[2] == c.name, one_to_many))
        # Если класс не пустой
        if len(c_classes) > 0:
            # Баллы учеников класса
            c_gpa = [gpa for _, gpa, _ in c_classes]
            # Средний балл учеников класса, с округлением до 2 знаков
            c_gpa_sum = round(sum(c_gpa)/len(c_classes), 2)
            res_2_unsorted.append((c.name, c_gpa_sum))
    # Сортировка по среднему баллу
    res_2 = sorted(res_2_unsorted, key=itemgetter(1), reverse=True)
    print(res_2)

    print('\nЗадание E3')
    many_to_many = []
    mtm_function(many_to_many)
    res_3 = {}
    print(set_a(many_to_many, res_3))


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

- test_main.py

```
from main import otm_function, mtm_function, set_a

#TDD-фреймворк
def test_otm_function():
    expected = [('Алексеев', 4.0, '9 мат'), ('Борисов', 4.8, '10 мат'), ('Иванов', 5.0, '11 мат'),
                ('Петров', 4.4, '11 мат'), ('Сидоров', 3.9, '11 мат'), ('Антонов', 4.0, '11 мат')]
    actual = []
    assert otm_function(actual) == expected
```

```python
def test_mtm_function():
    expected = [('Алексеев', 4.0, '9 мат'), ('Борисов', 4.8, '10 мат'), ('Иванов', 5.0, '11 мат'),
            ('Петров', 4.4, '11 мат'), ('Сидоров', 3.9, '11 мат'), ('Антонов', 4.0, '11 мат'),
            ('Алексеев', 4.0, '9 гум'), ('Борисов', 4.8, '10 гум'), ('Иванов', 5.0, '11 гум'),
            ('Петров', 4.4, '11 гум'), ('Сидоров', 3.9, '11 гум'), ('Антонов', 4.0, '11 гум')]
    actual = []
    assert mtm_function(actual) == expected


def test_set_a():
    expected = {'Алексеев': ['9 мат', '9 гум'], 'Антонов': ['11 мат', '11 гум']}
    m_to_m = []
    res = {}
    actual = set_a(mtm_function(m_to_m), res)
    assert expected == actual
```

Результаты вывода

```
(venv) PS C:\Users\nanny\Desktop\git\RK2> pytest -v
============================= test session starts =============================
platform win32 -- Python 3.11.4, pytest-7.4.3, pluggy-1.3.0 -- C:\Users\nanny\Desktop\git\Lab2\venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\nanny\Desktop\git\RK2
collected 3 items

test_main.py::test_otm_function PASSED                                   [ 33%]
test_main.py::test_mtm_function PASSED                                   [ 66%]
test_main.py::test_set_a PASSED                                          [100%]

============================== 3 passed in 0.04s ==============================
```