

```

import unittest
from operator import itemgetter

class Department:
    def __init__(self, id, name):
        self.id = id
        self.name = name

    def get_students(self, student_groups, students_in_departments):
        department_students = [link.student_id for link in students_in_departments if
link.department_id == self.id]
        return [group for group in student_groups if group.id in department_students]

class StudentGroup:
    def __init__(self, id, number, department_id, student_count):
        self.id = id
        self.number = number
        self.department_id = department_id
        self.student_count = student_count

class StudentsInDepartment:
    def __init__(self, department_id, student_id):
        self.department_id = department_id
        self.student_id = student_id

def get_related_students_and_departments(student_groups, departments,
students_in_departments):
    one_to_many = [(group.number, department.name) for group in student_groups for
department in departments if
                    group.department_id == department.id]
    return sorted(one_to_many, key=itemgetter(1))

def get_department_student_counts(student_groups, departments,
students_in_departments):
    res_2_unsorted = []
    for department in departments:
        group_student_counts = [group.student_count for group in student_groups if
group.department_id == department.id]
        total_student_count = sum(group_student_counts)
        res_2_unsorted.append((department.name, total_student_count))
    return sorted(res_2_unsorted, key=itemgetter(1), reverse=True)

def get_group_student_counts(student_groups):
    return {group.number: group.student_count for group in student_groups}

class TestStudentDepartmentFunctions(unittest.TestCase):
    def setUp(self):
        self.departments = [
            Department(1, 'Кафедра информационных технологий'),
            Department(2, 'Кафедра математики'),
            Department(3, 'Кафедра физики'),
        ]
        self.student_groups = [
            StudentGroup(101, 'ИТ-101', 1, 203),
            StudentGroup(102, 'Мат-201', 2, 104),
            StudentGroup(103, 'Физ-301', 3, 162),
        ]
        self.students_in_departments = [
            StudentsInDepartment(1, 1),
            StudentsInDepartment(2, 2),
            StudentsInDepartment(3, 3),

```

```

        StudentsInDepartment(3, 4),
        StudentsInDepartment(3, 5),
    ]

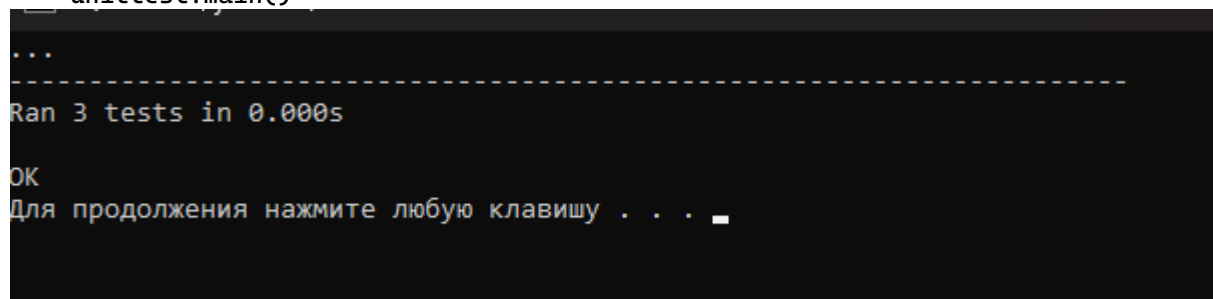
    def test_get_related_students_and_departments(self):
        result = get_related_students_and_departments(self.student_groups,
self.departments, self.students_in_departments)
        expected_result = [('ИТ-101', 'Кафедра информационных технологий'),
                           ('Мат-201', 'Кафедра математики'),
                           ('Физ-301', 'Кафедра физики')]
        self.assertEqual(result, expected_result)

    def test_get_department_student_counts(self):
        result = get_department_student_counts(self.student_groups, self.departments,
self.students_in_departments)
        expected_result = [('Кафедра физики', 162),
                           ('Кафедра информационных технологий', 203),
                           ('Кафедра математики', 104)]
        self.assertCountEqual(result, expected_result)

    def test_get_group_student_counts(self):
        result = get_group_student_counts(self.student_groups)
        expected_result = {'ИТ-101': 203, 'Мат-201': 104, 'Физ-301': 162}
        self.assertEqual(result, expected_result)

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

```



```

...
-----
Ran 3 tests in 0.000s

OK
Для продолжения нажмите любую клавишу . . . _

```