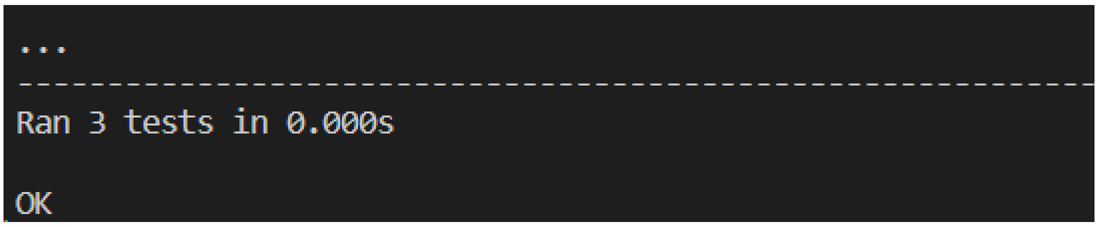
```
import unittest
from operator import itemgetter


class Faculty:
    def __init__(self, id, name, disciplines, university_id):
        self.id = id
        self.name = name
        self.disciplines = disciplines
        self.university_id = university_id


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


class UniversityFaculty:
    def __init__(self, faculty_id, university_id):
        self.faculty_id = faculty_id
        self.university_id = university_id


def get_facultys_by_university(universities, university_facultys, facultys):
    result = []
    for university in universities:
        university_faculty_ids = [fd.faculty_id for fd in university_facultys if
fd.university_id == university.id]
        related_facultys = [faculty for faculty in facultys if faculty.id in
university_faculty_ids]
        result.append((university.name, [faculty.name for faculty in
related_facultys]))
    return result


def get_university_discipline_counts(universities, university_facultys,
facultys):
    result = []
    for university in universities:
        university_faculty_ids = [fd.faculty_id for fd in university_facultys if
fd.university_id == university.id]

        related_facultys = [faculty for faculty in facultys if faculty.id in
university_faculty_ids]

        disciplines_sum = sum([faculty.disciplines for faculty in
related_facultys])

        result.append((university.name, disciplines_sum, [(faculty.name,
faculty.disciplines) for faculty in related_facultys]))

    return result


def get_matching_universities(universities, university_contains_word,
university_facultys, facultys):

    result = []

    for university in universities:

        if university_contains_word.lower() in university.name.lower():

            university_faculty_ids = [fd.faculty_id for fd in university_facultys
if fd.university_id == university.id]

            related_facultys = [faculty for faculty in facultys if faculty.id in
university_faculty_ids]

            result.append((university.name, [faculty.name for faculty in
related_facultys]))

    return result


class TestfacultyuniversityQueries(unittest.TestCase):

    def setUp(self):

        # Test data setup

        self.facultys = [

            Faculty(1, "Факультет Информатики", 5, 1),

            Faculty(2, "Факультет математики", 3, 1),

            Faculty(3, "Факультет дизайна", 4, 2),

            Faculty(4, "Факультет Физики", 2, 3),

            Faculty(5, "Факультет материалов", 6, 2)

        ]


        self.universities = [

            University(1, "МГТУ"),

            University(2, "Университет Президента"),
            University(3, "Технологический институт")
        ]

        self.university_facultys = [
            UniversityFaculty(1, 1),
            UniversityFaculty(2, 1),
            UniversityFaculty(3, 2),
            UniversityFaculty(4, 3),
            UniversityFaculty(5, 2)
        ]

    def test_get_facultys_by_university(self):
        result = get_facultys_by_university(self.universities,
self.university_facultys, self.facultys)
        expected_result = [
            ('МГТУ', ['Факультет Информатики', 'Факультет математики']),
            ('Университет Президента', ['Факультет дизайна', 'Факультет
материалов']),
            ('Технологический институт', ['Факультет Физики'])
        ]
        self.assertEqual(result, expected_result)

    def test_get_university_discipline_counts(self):
        result = get_university_discipline_counts(self.universities,
self.university_facultys, self.facultys)
        expected_result = [
            ('МГТУ', 8, [('Факультет Информатики', 5), ('Факультет математики',
3)]),
            ('Университет Президента', 10, [('Факультет дизайна', 4), ('Факультет
материалов', 6)]),
            ('Технологический институт', 2, [('Факультет Физики', 2)])
        ]
        self.assertEqual(result, expected_result)

    def test_get_matching_universities(self):
        result = get_matching_universities(self.universities, "университет",
self.university_facultys, self.facultys)

        expected_result = [

            ('МГТУ', ['Факультет Информатики', 'Факультет математики']),

            ('Университет Президента', ['Факультет дизайна', 'Факультет
материалов'])

        ]

        self.assertEqual(result, expected_result)


if __name__ == '__main__':

    unittest.main()
```



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

OK
```