Код:

class Faculty:

    def \_\_init\_\_(self, id, name):

        self.id = id

        self.name = name

        self.departments = []

class Department:

    def \_\_init\_\_(self, id, name, faculty\_id):

        self.id = id

        self.name = name

        self.faculty\_id = faculty\_id

class Professor:

    def \_\_init\_\_(self, id, name, salary):

        self.id = id

        self.name = name

        self.salary = salary

        self.departments = []

class ProfessorDepartment:

    def \_\_init\_\_(self, professor\_id, department\_id):

        self.professor\_id = professor\_id

        self.department\_id = department\_id

def get\_professor\_department\_mapping(professor\_department\_relations, professors, departments):

    """Создает отображение кафедра-преподаватели."""

    department\_by\_id = {dept.id: dept for dept in departments}

    professor\_by\_id = {prof.id: prof for prof in professors}

    for relation in professor\_department\_relations:

        professor\_by\_id[relation.professor\_id].departments.append(department\_by\_id[relation.department\_id])

    departments\_with\_professors = {dept.name: [] for dept in departments}

    for relation in professor\_department\_relations:

        dept\_name = department\_by\_id[relation.department\_id].name

        prof\_name = professor\_by\_id[relation.professor\_id].name

        departments\_with\_professors[dept\_name].append(prof\_name)

    return departments\_with\_professors

def calculate\_total\_salary\_by\_department(professor\_department\_relations, professors, departments):

    """Рассчитывает суммарную зарплату по кафедрам."""

    department\_by\_id = {dept.id: dept for dept in departments}

    professor\_by\_id = {prof.id: prof for prof in professors}

    salary\_by\_department = {dept.name: 0 for dept in departments}

    for relation in professor\_department\_relations:

        prof = professor\_by\_id[relation.professor\_id]

        dept\_name = department\_by\_id[relation.department\_id].name

        salary\_by\_department[dept\_name] += prof.salary

    return salary\_by\_department

def get\_departments\_with\_keyword(keyword, professor\_department\_relations, professors, departments):

    """Возвращает кафедры с указанным ключевым словом и связанных с ними преподавателей."""

    professor\_by\_id = {prof.id: prof for prof in professors}

    departments\_with\_keyword = {

        dept.id: dept.name for dept in departments if keyword in dept.name

    }

    result = {}

    for dept\_id, dept\_name in departments\_with\_keyword.items():

        result[dept\_name] = [

            professor\_by\_id[relation.professor\_id].name

            for relation in professor\_department\_relations

            if relation.department\_id == dept\_id

        ]

    return result

# Данные для тестов

faculty1 = Faculty(1, "IU")

faculty2 = Faculty(2, "RK")

# Кафедры

department1 = Department(1, "Кафедра U5", faculty1.id)

department2 = Department(2, "RK9", faculty2.id)

department3 = Department(3, "IU4", faculty1.id)

faculty1.departments.extend([department1, department3])

faculty2.departments.append(department2)

# Преподаватели

professor1 = Professor(1, "Ivanov", 50000)

professor2 = Professor(2, "Petrov", 55000)

professor3 = Professor(3, "Sidorov", 60000)

professor4 = Professor(4, "Smirnov", 58000)

professor\_department\_relations = [

    ProfessorDepartment(professor1.id, department1.id),

    ProfessorDepartment(professor1.id, department3.id),

    ProfessorDepartment(professor2.id, department1.id),

    ProfessorDepartment(professor3.id, department2.id),

    ProfessorDepartment(professor4.id, department3.id),

]

# Тесты

if \_\_name\_\_ == "\_\_main\_\_":

    import unittest

    class TestFacultySystem(unittest.TestCase):

        def setUp(self):

            self.professors = [professor1, professor2, professor3, professor4]

            self.departments = [department1, department2, department3]

            self.relations = professor\_department\_relations

        def test\_professor\_department\_mapping(self):

            result = get\_professor\_department\_mapping(self.relations, self.professors, self.departments)

            self.assertIn("Кафедра U5", result)

            self.assertIn("Ivanov", result["Кафедра U5"])

        def test\_calculate\_total\_salary\_by\_department(self):

            result = calculate\_total\_salary\_by\_department(self.relations, self.professors, self.departments)

            self.assertEqual(result["Кафедра U5"], 105000)  # Ivanov + Petrov

            self.assertEqual(result["IU4"], 108000)         # Ivanov + Smirnov

        def test\_get\_departments\_with\_keyword(self):

            result = get\_departments\_with\_keyword("Кафедра", self.relations, self.professors, self.departments)

            self.assertIn("Кафедра U5", result)

            self.assertIn("Ivanov", result["Кафедра U5"])

    unittest.main()

Результат:

...

----------------------------------------------------------------------

Ran 3 tests in 0.000s

OK