Вариант 26Б  
**Текст программы:**

rk2.py:

# Класс сущности "Студенческая группа"  
class Group:  
  
 def \_\_init\_\_(self, id, name, number\_of\_students):  
 self.id = id # id группы  
 self.name = name # Название группы  
 self.number\_of\_students = number\_of\_students # Кол-во студентов  
  
  
# Класс сущности "Учебный курс"  
class Course:  
  
 def \_\_init\_\_(self, id, name):  
 self.id = id # id курса  
 self.name = name # Название курса  
  
  
# Класс сущности "Студенческая группа - Учебный курс" (Для связи многие ко многим)  
class GroupCourse:  
  
 def \_\_init\_\_(self, course\_id, group\_id):  
 self.course\_id = course\_id # id курса  
 self.group\_id = group\_id # id группы  
  
  
# Функции для выполнения запросов  
def query\_b1(groups, courses): #Запрос Б1  
 one\_to\_many = [(g.name, g.number\_of\_students, c.name)  
 for c in courses  
 for g in groups  
 if g.id == c.id]  
 one\_to\_many.append((groups[4].name, groups[4].number\_of\_students, courses[3].name))  
 return sorted(one\_to\_many, key=lambda x: x[0])  
  
def query\_b2(groups, courses): #Запрос Б2  
 b2 = []  
 for course in courses:  
 course\_groups = list(filter(lambda i: i[2] == course.name, query\_b1(groups, courses)))  
 if len(course\_groups) > 0:  
 course\_students = [students for \_, students, \_ in course\_groups]  
 course\_students\_sum = sum(course\_students)  
 b2.append((course.name, course\_students\_sum))  
 return sorted(b2, key=lambda x: x[1], reverse=True)  
  
def query\_b3(groups, courses, groups\_courses): #Запрос Б3  
 many\_to\_many\_group = [(c.name, gc.course\_id, gc.group\_id)  
 for c in courses  
 for gc in groups\_courses  
 if c.id == gc.course\_id]  
  
 many\_to\_many = [(g.name, g.number\_of\_students, name)  
 for name, \_, group\_id in many\_to\_many\_group  
 for g in groups if g.id == group\_id]  
  
 b3 = {}  
 """  
 Пояснение к запросу Б3:  
 Вместо окончания "ов" в фамилии сотрудников,   
 я взял окончание "B" в имени группы  
 """  
 for g in groups:  
 if 'B' in g.name:  
 course\_groups = list(filter(lambda i: i[0] == g.name, many\_to\_many))  
 course\_groups\_names = [x for \_, \_, x in course\_groups]  
 b3[g.name] = course\_groups\_names  
 return b3

TDD\_test.py

import unittest  
from rk2 import Group, Course, GroupCourse, query\_b1, query\_b2, query\_b3  
  
class TestQueries(unittest.TestCase):  
 def setUp(self):  
 # Cтуденческие группы  
 self.groups = [  
 Group(1, 'IU1-12A', 26),  
 Group(2, 'IU1-11B', 28),  
 Group(3, 'IU3-21C', 20),  
 Group(4, 'IU4-42D', 24),  
 Group(5, 'IU5-44B', 19),  
 ]  
 # Учебные курсы  
 self.courses = [  
 Course(1, 'First'),  
 Course(2, 'Second'),  
 Course(3, 'Third'),  
 Course(4, 'Fourth'),  
 ]  
 # Студенческая группа - Учебный курс  
 self.groups\_courses = [  
 GroupCourse(1, 1),  
 GroupCourse(1, 2),  
 GroupCourse(2, 3),  
 GroupCourse(4, 4),  
 GroupCourse(4, 5),  
 GroupCourse(3, 5),  
 ]  
  
 def test\_query\_b1(self):  
 result = query\_b1(self.groups, self.courses)  
 self.assertEqual(len(result), 5)  
  
 def test\_query\_b2(self):  
 result = query\_b2(self.groups, self.courses)  
 self.assertEqual(len(result), 4)  
  
 def test\_query\_b3(self):  
 result = query\_b3(self.groups, self.courses, self.groups\_courses)  
 self.assertEqual(len(result), 2)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

**Результат выполнения:**============================= test session starts =============================

collecting ... collected 3 items

TDD\_test.py::TestQueries::test\_query\_b1 PASSED [ 33%]

TDD\_test.py::TestQueries::test\_query\_b2 PASSED [ 66%]

TDD\_test.py::TestQueries::test\_query\_b3 PASSED [100%]

============================== 3 passed in 0.03s ==============================