РК-2 Курс ПиКЯП Турсков Евгений ИУ5-36Б

Вариант Д26

**Текст программы:**

**from** operator **import** itemgetter  
  
**class** StudentGroup:  
 # Студенческая группа  
 **def** \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
  
**class** Course:  
 # Учебный курс  
 **def** \_\_init\_\_(self, id, title, avg\_grade, group\_id):  
 self.id = id  
 self.title = title  
 self.avg\_grade = avg\_grade  
 self.group\_id = group\_id  
  
**class** GroupCourse:  
 # 'Группы на курсе' для реализации  
 # связи многие-ко-многим  
 **def** \_\_init\_\_(self, group\_id, course\_id):  
 self.group\_id = group\_id  
 self.course\_id = course\_id  
  
# Студенческие группы  
groups = [  
 StudentGroup(1, 'Группа ИТ-101'),  
 StudentGroup(2, 'Армейская Группа СВО-243'),  
 StudentGroup(3, 'Группа Физ-301'),  
 StudentGroup(4, 'Группа Матем-201'),  
 StudentGroup(5, 'Астрономическая Группа 504'),  
 StudentGroup(6, 'Археологическая Группа А-102'),  
]  
  
# Учебные курсы  
courses = [  
 Course(1, 'Программирование', 85, 1),  
 Course(2, 'Математический анализ', 90, 2),  
 Course(3, 'Физика', 75, 3),  
 Course(4, 'История', 80, 4),  
 Course(5, 'Изучение языков', 88, 2),  
 Course(6, 'Философия', 93, 3),  
 Course(7, 'Экономический анализ', 87, 5),  
 Course(8, 'Введение в биологию', 76, 6),  
 Course(9, 'Эволюция', 84, 1),  
 Course(10, 'Инвестирование', 92, 2),  
 Course(11, 'Теория алгоритмов', 89, 3),  
 Course(12, 'Статистика данных', 78, 4),  
]  
  
# Связи многие-ко-многим  
groups\_courses = [  
 GroupCourse(1, 1),  
 GroupCourse(2, 2),  
 GroupCourse(3, 3),  
 GroupCourse(4, 4),  
 GroupCourse(2, 5),  
 GroupCourse(3, 6),  
 GroupCourse(4, 6),  
 GroupCourse(3, 1),  
 GroupCourse(5, 7),  
 GroupCourse(6, 8),  
 GroupCourse(1, 9),  
 GroupCourse(2, 10),  
 GroupCourse(3, 11),  
 GroupCourse(4, 12),  
 GroupCourse(5, 6),  
 GroupCourse(6, 11),  
]  
  
**def** build\_one\_to\_many(groups, courses):  
 **return** [(c.title, c.avg\_grade, g.name)  
 **for** g **in** groups  
 **for** c **in** courses  
 **if** c.group\_id == g.id]  
  
**def** build\_many\_to\_many(groups, courses, groups\_courses):  
 many\_to\_many\_temp = [(g.name, gc.group\_id, gc.course\_id)  
 **for** g **in** groups  
 **for** gc **in** groups\_courses  
 **if** g.id == gc.group\_id]  
  
 many\_to\_many = [(c.title, c.avg\_grade, group\_name)  
 **for** group\_name, group\_id, course\_id **in** many\_to\_many\_temp  
 **for** c **in** courses **if** c.id == course\_id]  
 **return** many\_to\_many  
  
**def** query\_d1(one\_to\_many):  
 """Запрос Д1: все курсы, оканчивающиеся на 'ов'."""  
 **return** [(title, group\_name)  
 **for** title, avg\_grade, group\_name **in** one\_to\_many  
 **if** title.endswith("ов")]  
  
**def** query\_d2(one\_to\_many):  
 """Запрос Д2: Средняя оценка курсов по каждой группе, отсортированные по убыванию."""  
 res\_d2\_unsorted = []  
 # Группы  
 groups\_names = set([g **for** \_, \_, g **in** one\_to\_many])  
 **for** g **in** groups\_names:  
 g\_courses = list(filter(**lambda** i: i[2] == g, one\_to\_many))  
 **if** len(g\_courses) > 0:  
 g\_grades = [avg\_grade **for** \_, avg\_grade, \_ **in** g\_courses]  
 g\_avg\_grade = sum(g\_grades) / len(g\_grades)  
 res\_d2\_unsorted.append((g, g\_avg\_grade))  
 res\_d2 = sorted(res\_d2\_unsorted, key=itemgetter(1), reverse=**True**)  
 **return** res\_d2  
  
**def** query\_d3(groups, many\_to\_many):  
 """Запрос Д3: Для всех групп, начинающихся с 'А', выдать список курсов."""  
 res\_d3 = {}  
 **for** g **in** groups:  
 **if** g.name.startswith("А"):  
 g\_courses = list(filter(**lambda** i: i[2] == g.name, many\_to\_many))  
 g\_courses\_titles = [x **for** x, \_, \_ **in** g\_courses]  
 res\_d3[g.name] = g\_courses\_titles  
 **return** res\_d3  
  
**def** main():  
 one\_to\_many = build\_one\_to\_many(groups, courses)  
 many\_to\_many = build\_many\_to\_many(groups, courses, groups\_courses)  
  
 print('Запрос Д1')  
 d1 = query\_d1(one\_to\_many)  
 **for** item **in** d1:  
 print(f"{item[0]} - {item[1]}")  
  
 print('\nЗапрос Д2')  
 d2 = query\_d2(one\_to\_many)  
 **for** item **in** d2:  
 print(f"{item[0]} - {item[1]}")  
  
 print('\nЗапрос Д3')  
 d3 = query\_d3(groups, many\_to\_many)  
 **for** g **in** d3:  
 print(f"{g}: {', '.join(d3[g])}")  
  
**if** \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Модульные тесты:**

**import** pytest  
**from** program **import** groups, courses, groups\_courses, build\_one\_to\_many, build\_many\_to\_many, query\_d1, query\_d2, query\_d3

#pytest RK2/test\_program.py

@pytest.fixture  
**def** data():  
 one\_to\_many = build\_one\_to\_many(groups, courses)  
 many\_to\_many = build\_many\_to\_many(groups, courses, groups\_courses)  
 **return** one\_to\_many, many\_to\_many  
  
**def** test\_query\_d1(data):  
 one\_to\_many, \_ = data  
 result = query\_d1(one\_to\_many)  
 # проверим, что все названия заканчиваются на "ов"  
 **for** title, \_ **in** result:  
 **assert** title.endswith("ов"), f"Название курса {title} не заканчивается на 'ов'"  
 # можем также проверить кол-во результатов или конкретные значения  
 **assert** len(result) > 0  
  
**def** test\_query\_d2(data):  
 one\_to\_many, \_ = data  
 result = query\_d2(one\_to\_many)  
 # проверим, что результат отсортирован по убыванию средней оценки  
 avg\_grades = [r[1] **for** r **in** result]  
 **assert** avg\_grades == sorted(avg\_grades, reverse=**True**), "Результат не отсортирован по убыванию"  
 # проверим, что существует хотя бы одна группа  
 **assert** len(result) > 0  
  
**def** test\_query\_d3(data):  
 one\_to\_many, many\_to\_many = data  
 result = query\_d3(groups, many\_to\_many)  
 # проверим, что все ключи результата начинаются на "А"  
 **for** g\_name **in** result.keys():  
 **assert** g\_name.startswith("А"), f"Группа {g\_name} не начинается на 'А'"  
 # проверим, что результаты непустые  
 **for** courses\_list **in** result.values():  
 **assert** len(courses\_list) >= 0 # может быть 0, если группа не содержит курсов

**Результаты тестирования:**

**============================================================================= test session starts ==============================================================================**

**platform darwin -- Python 3.10.2, pytest-7.1.1, pluggy-1.0.0**

**rootdir: /Users/evgenijturskov/Desktop/вуз/3 семестр/ПКиЯП/BMSTU\_3SEM\_TURSKOV**

**plugins: anyio-3.7.1**

**collected 3 items**

**RK2/test\_program.py ... [100%]**