Толкачев. РТ5-31Б РК1

Вариант Е. Предметная область 28.

«Кафедра» и «Студенческая группа» связаны соотношением один-ко-многим. Выведите список всех кафедр, у которых в названии присутствует слово «кафедра», и список учащихся в них студенческих групп.

«Кафедра» и «Студенческая группа» связаны соотношением один-ко-многим. Выведите список кафедр со средним количеством студенческих групп в каждом кафедре, отсортированный по среднему количеству студенческих групп. Среднее количество студенческих групп должно быть округлена до 2 знака после запятой.

«Кафедра» и «Студенческая группа» связаны соотношением многие-ко-многим. Выведите список всех студенческих групп, у которых название начинается с буквы «Р», и названия их кафедр. Текст программы:

# используется для сортировки  
from operator import itemgetter  
  
  
class Emp:  
 *"""Студенческая группа"""* def \_\_init\_\_(self, id, name, quan, dep\_id):  
 self.id = id  
 self.name = name  
 self.quan = quan  
 self.dep\_id = dep\_id  
  
  
class Dep:  
 *"""Кафедра"""* def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
  
  
class EmpDep:  
 *"""  
 'Сотрудники отдела' для реализации  
 связи многие-ко-многим  
 """* def \_\_init\_\_(self, dep\_id, emp\_id):  
 self.dep\_id = dep\_id  
 self.emp\_id = emp\_id  
  
  
# Отделы  
deps = [  
 Dep(1, 'Системы обработки информации и управления'),  
 Dep(2, 'Системы автоматического управления'),  
 Dep(3, 'Проектирование и технология производства электронной аппаратуры'),  
 Dep(4, 'Радиоэлектронные системы и устройства'),  
  
]  
  
# Сотрудники  
emps = [  
 Emp(1, 'РТ5-31Б', 27, 1),  
 Emp(2, 'РТ4-31', 22, 2),  
 Emp(3, 'РТ2-31', 21, 3),  
 Emp(4, 'РТ1-31', 18, 4),  
 Emp(5, 'ИУ5-32', 15, 1),  
 Emp(6, 'ИУ5-31', 14, 1),  
]  
  
emps\_deps = [  
 EmpDep(1, 1),  
 EmpDep(2, 2),  
 EmpDep(3, 3),  
 EmpDep(4, 4),  
 EmpDep(1, 5),  
 EmpDep(1, 6),  
  
]  
  
  
def main():  
 *"""Основная функция"""* # Соединение данных один-ко-многим  
 one\_to\_many = [(e.name, e.quan, d.name)  
 for d in deps  
 for e in emps  
 if e.dep\_id == d.id]  
  
 # Соединение данных многие-ко-многим  
 many\_to\_many\_temp = [(d.name, ed.dep\_id, ed.emp\_id)  
 for d in deps  
 for ed in emps\_deps  
 if d.id == ed.dep\_id]  
  
 many\_to\_many = [(e.name, e.quan, dep\_name)  
 for dep\_name, dep\_id, emp\_id in many\_to\_many\_temp  
 for e in emps if e.id == emp\_id]  
  
 print('Задание E1')  
 # res\_11 = sorted(one\_to\_many, key=itemgetter(2))  
 res\_11 = list(filter(lambda x: 'Системы' in x[2], one\_to\_many))  
 print(res\_11)  
  
 print('\nЗадание E2')  
 res = []  
 for emp in emps:  
 mid\_res = []  
 sum = 0  
 count = 0  
 for emp\_dep in emps\_deps:  
 if emp\_dep.ide\_id == emp.id:  
 for dep in deps:  
 if dep.id == emp\_dep.language\_id:  
 sum += dep.diificulity  
 count += 1  
 try:  
 mid\_res.append(emp.name)  
 mid\_res.append(round(sum / count, 2))  
 res.append(mid\_res)  
 except ZeroDivisionError:  
 pass  
 return sorted(res, key=itemgetter(1), reverse=False)  
  
 print('\nЗадание E3')  
 res\_13 = {}  
 # Перебираем все отделы  
 for d in deps:  
 if 'П' in d.name[1]:  
 # Список сотрудников отдела  
 d\_emps = list(filter(lambda i: i[2] == d.name, many\_to\_many))  
 # Только ФИО сотрудников  
 d\_emps\_names = [x for x, \_, \_ in d\_emps]  
 # Добавляем результат в словарь  
 # ключ - отдел, значение - список фамилий  
 res\_13[d.name] = d\_emps\_names  
  
 print(res\_13)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()