МГТУ им Н. Э. Баумана

Кафедра «Системы обработки информации и управления»

Рубежный контроль №1

«Базовые компоненты интернет-технологий»

Преподаватель: Студент группы ИУ5-51Б: Гапанюк Ю. Е.

Афанасьев Д. М.

Москва, 2022

**Вариант А. Предметная область 27.**

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по отделам, сортировка по сотрудникам произвольная.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов с суммарной зарплатой сотрудников в каждом отделе, отсортированный по суммарной зарплате.
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех отделов, у которых в названии присутствует слово «отдел», и список работающих в них сотрудников.

|  |  |  |
| --- | --- | --- |
| 27 | Преподаватель | Учебный курс |

**Листинг программы**

from operator import itemgetter  
class Teacher:  
 def \_\_init\_\_(self, id, name, wage, id\_course):  
 self.id = id  
 self.name = name  
 self.wage = wage  
 self.id\_course = id\_course  
class Course:  
 def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
class Teacher\_and\_course:  
 def \_\_init\_\_(self, id\_teacher, id\_course):  
 self.id\_teacher = id\_teacher  
 self.id\_course = id\_course  
teachers = [  
 Teacher(1, "Александров", 1000, 1),  
 Teacher(2, "Иванов", 1500, 3),  
 Teacher(3, "Барановский", 2000, 2),  
 Teacher(4, "Егоров", 1600, 4),  
 Teacher(5, "Ченский", 1800, 2),  
 Teacher(6, "Соколов", 1300, 4),  
 Teacher(7, "Сороковиков", 1900, 1),  
 Teacher(8, "Смирнов", 1600, 3),  
]  
courses = [  
 Course(1, "физика"),  
 Course(2, "математика"),  
 Course(3, "химия"),  
 Course(4, "информатика")  
 ]  
teacherofcourse = [  
 Teacher\_and\_course(1, 1),  
 Teacher\_and\_course(2, 3),  
 Teacher\_and\_course(3, 2),  
 Teacher\_and\_course(4, 4),  
 Teacher\_and\_course(5, 2),  
 Teacher\_and\_course(6, 4),  
 Teacher\_and\_course(7, 3),  
 Teacher\_and\_course(8, 1),  
 ]  
  
def main():  
 one\_to\_many = [(t.name, t.wage, c.name)  
 for t in teachers  
 for c in courses  
 if t.id\_course == c.id]  
 print('Task А1')  
 res1 = sorted(one\_to\_many, key=itemgetter(2))  
 print(res1)  
  
 many\_to\_many\_tmp = [(c.name, tc.id\_course, tc.id\_teacher)  
 for c in courses  
 for tc in teacherofcourse  
 if c.id == tc.id\_course]  
 many\_to\_many = [(t.id, id\_course)  
 for name, id\_course, id\_teacher in many\_to\_many\_tmp  
 for t in teachers  
 if t.id == id\_teacher]  
 print('Task А2')  
 res2\_unsorted = []  
 for c in courses:  
 c\_courses = list(filter(lambda i:i[2]==c.name, one\_to\_many))  
 if len(c\_courses) > 0:  
 c\_sizes = [sal for \_, sal, \_ in c\_courses]  
 c\_sizes\_sum = sum(c\_sizes) / 2  
 res2\_unsorted.append((c.name, c\_sizes\_sum))  
 res2 = sorted(res2\_unsorted, key=itemgetter(1), reverse=True)  
 print(res2)  
  
 many\_to\_many\_temp = [(c.name, tc.id\_course, tc.id\_teacher)  
 for c in courses  
 for tc in teacherofcourse  
 if c.id == tc.id\_course]  
 many\_to\_many = [(t.name, t.wage, course\_name)  
 for course\_name, id\_course, id\_teacher in many\_to\_many\_temp  
 for t in teachers  
 if t.id == id\_teacher]  
 print('Task А3')  
 res3 = {}  
 for c in courses:  
 if 'физика' in c.name:  
 c\_courses = list(filter(lambda i: i[2] == c.name, many\_to\_many))  
 c\_courses\_names = [x for x, \_, \_ in c\_courses]  
 res3[c.name] = c\_courses\_names  
 print(res3)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Результат выполнения**

