**main.py**

from Department import Department  
from StudentGroup import StudentGroup  
from StudentGroupDepartment import StudentGroupDepartment  
from Requests import findGroupsEndWithKey, DepartmentsSortedByAvgNumberOfStudents, findDepartmentsStartWithKey  
  
# Список кафедр  
departments = [  
 Department(id=1, name="Прикладная математика"),  
 Department(id=2, name="Информатика"),  
 Department(id=3, name="Физика"),  
 Department(id=4, name="Астрономия"),  
 Department(id=5, name="История"),  
 Department(id=6, name="Археология"),  
 Department(id=7, name="Астрофизика"),  
]  
  
# Список студенческих групп  
student\_groups = [  
 StudentGroup(id=1, group\_name="Группа 101", number\_of\_students=30, department\_id=1),  
 StudentGroup(id=2, group\_name="Группа 102", number\_of\_students=25, department\_id=2),  
 StudentGroup(id=3, group\_name="Группа 103", number\_of\_students=28, department\_id=1),  
 StudentGroup(id=4, group\_name="Группа 104", number\_of\_students=22, department\_id=3),  
 StudentGroup(id=5, group\_name="Группа 105", number\_of\_students=27, department\_id=4),  
 StudentGroup(id=6, group\_name="Группа 1003", number\_of\_students=20, department\_id=4),  
 StudentGroup(id=7, group\_name="Группа 104А", number\_of\_students=21, department\_id=5),  
 StudentGroup(id=8, group\_name="Группа 105А", number\_of\_students=24, department\_id=6),  
 StudentGroup(id=9, group\_name="Группа 1003А", number\_of\_students=23, department\_id=7),  
 StudentGroup(id=10, group\_name="Группа 104Б", number\_of\_students=21, department\_id=5),  
 StudentGroup(id=11, group\_name="Группа 145Б", number\_of\_students=20, department\_id=4),  
 StudentGroup(id=12, group\_name="Группа 1Б03", number\_of\_students=19, department\_id=7),  
  
]  
  
group\_deps = [  
 StudentGroupDepartment(student\_group\_id=1, department\_id=1),  
 StudentGroupDepartment(student\_group\_id=2, department\_id=2),  
 StudentGroupDepartment(student\_group\_id=3, department\_id=1),  
 StudentGroupDepartment(student\_group\_id=4, department\_id=3),  
 StudentGroupDepartment(student\_group\_id=5, department\_id=4),  
 StudentGroupDepartment(student\_group\_id=6, department\_id=4),  
 StudentGroupDepartment(student\_group\_id=7, department\_id=5),  
 StudentGroupDepartment(student\_group\_id=8, department\_id=6),  
 StudentGroupDepartment(student\_group\_id=9, department\_id=7),  
 StudentGroupDepartment(student\_group\_id=10, department\_id=5),  
 StudentGroupDepartment(student\_group\_id=11, department\_id=4),  
 StudentGroupDepartment(student\_group\_id=12, department\_id=7),  
]  
  
# Таблица для отношения один-ко-многим  
one\_to\_many = [(g.group\_name, g.number\_of\_students, d.name)  
 for d in departments  
 for g in student\_groups  
 if g.department\_id == d.id]  
  
# Соединение таблицы многие-ко-многим  
temp = [(group.group\_name, group\_dep.department\_id)  
 for group in student\_groups  
 for group\_dep in group\_deps  
 if group.id == group\_dep.student\_group\_id  
]  
  
many\_to\_many = [(group\_name, dept.name)  
 for dept in departments  
 for group\_name, dep\_id in temp  
 if dep\_id == dept.id  
]  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # Запрос 1: Студенческие группы, заканчивающиеся на key и их кафедры (один-ко-многим)  
 findGroupsEndWithKey(one\_to\_many, "03")  
  
 # Запрос 2: Кафедры, отсортированные по среднему кол-ву студентов в группах (один-ко-многим)  
 DepartmentsSortedByAvgNumberOfStudents(one\_to\_many)  
  
 # Запрос 3: Кафедры, начинающиеся на букву == key и их студенческие группы (многие-ко-многим)  
 findDepartmentsStartWithKey(many\_to\_many, "А")

**StudentGroupDepartment.py**

class StudentGroupDepartment:  
 def \_\_init\_\_(self, student\_group\_id, department\_id):  
 self.student\_group\_id = student\_group\_id  
 self.department\_id = department\_id  
  
 def \_\_repr\_\_(self):  
 return f"StudentGroupDepartment(student\_group\_id={self.student\_group\_id}, department\_id={self.department\_id})"

**StudentGroup.py**

class StudentGroup:  
 def \_\_init\_\_(self, id, group\_name, number\_of\_students, department\_id):  
 self.id = id  
 self.group\_name = group\_name  
 self.number\_of\_students = number\_of\_students  
 self.department\_id = department\_id  
  
 def \_\_repr\_\_(self):  
 return (f"StudentGroup(id={self.id}, group\_name='{self.group\_name}', "  
 f"number\_of\_students={self.number\_of\_students}, department\_id={self.department\_id})")

**Department.py**

class Department:  
 def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
  
 def \_\_repr\_\_(self):  
 return f"Department(id={self.id}, name='{self.name}')"

**Requests.py**

from collections import defaultdict  
  
"""Запрос 1: Студенческие группы, заканчивающиеся на key и их кафедры"""  
def findGroupsEndWithKey(one\_to\_many, key, debug=True):  
 # Фильтруем группы, названия которых заканчиваются на ключ  
 filtered\_groups = [  
 {"group\_name": group\_name, "department\_name": department\_name}  
 for group\_name, \_, department\_name in one\_to\_many  
 if group\_name.endswith(key)  
 ]  
  
 if debug:  
 print(f"Запрос 1: Студенческие группы, заканчивающиеся на '{key}' и их кафедры:")  
 if not filtered\_groups:  
 print("Ничего не найдено")  
 else:  
 for item in filtered\_groups:  
 print(f"Группа: {item['group\_name']}, Кафедра: {item['department\_name']}")  
 print("\n")  
  
 return filtered\_groups  
  
"""Запрос 2: Кафедры, отсортированные по среднему кол-ву студентов в группах"""  
def DepartmentsSortedByAvgNumberOfStudents(one\_to\_many, debug=True):  
  
 # Сгруппируем количество студентов по названиям кафедр  
 department\_student\_counts = defaultdict(list)  
 for group\_name, number\_of\_students, department\_name in one\_to\_many:  
 department\_student\_counts[department\_name].append(number\_of\_students)  
  
 # Вычислим среднее количество студентов для каждой кафедры  
 average\_students = [  
 {  
 "department\_name": dept,  
 "average\_number\_of\_students": sum(counts) / len(counts)  
 }  
 for dept, counts in department\_student\_counts.items()  
 ]  
  
 # Отсортируем кафедры по среднему количеству студентов в порядке убывания  
 sorted\_average\_students = sorted(  
 average\_students,  
 key=lambda x: x["average\_number\_of\_students"],  
 reverse=True  
 )  
  
 if debug:  
 print("Запрос 2: Кафедры со средним количеством студентов, отсортированные по среднему количеству:")  
 for item in sorted\_average\_students:  
 print(  
 f"Кафедра: {item['department\_name']}, "  
 f"Среднее количество студентов: {item['average\_number\_of\_students']:.2f}"  
 )  
 print("\n")  
  
 return sorted\_average\_students  
  
  
"""Запрос 3: Кафедры, начинающиеся на букву == key и их студенческие группы"""  
def findDepartmentsStartWithKey(many\_to\_many, key, debug=True):  
 # Сгруппируем группы по названиям кафедр  
 department\_to\_groups = defaultdict(list)  
 for group\_name, department\_name in many\_to\_many:  
 department\_to\_groups[department\_name].append(group\_name)  
  
 # Фильтруем кафедры, названия которых начинаются с ключа  
 filtered\_departments = {  
 dept: groups  
 for dept, groups in department\_to\_groups.items()  
 if dept.startswith(key)  
 }  
  
 result = [  
 {"department\_name": dept, "student\_groups": groups}  
 for dept, groups in filtered\_departments.items()  
 ]  
  
 if debug:  
 print(f"Запрос 3: Кафедры, начинающиеся на '{key}', и их студенческие группы:")  
 if not result:  
 print("Ничего не найдено")  
 else:  
 for item in result:  
 groups = "\n\t".join(item["student\_groups"]) if item["student\_groups"] else "Нет групп"  
 print(f"Кафедра {item['department\_name']}:\n\t{groups}")  
 print("\n")  
  
 return result

**Результаты выполнения:**

|  |
| --- |
|  |
|  |
|  |