A1.

**from** **dataclasses** **import** dataclass

**from** **itertools** **import** groupby

@dataclass

**class** **Student**:

id: int

surname: str

group\_id: int

@dataclass

**class** **Group**:

id: int

name: str

groups = [Group(0, "B15"), Group(1, "B16"), Group(2, "B17")]

students = [

Student(0, "Ivanov", 1),

Student(1, "Petrov", 0),

Student(2, "Medvedev", 2),

Student(3, "Kozlov", 0),

]

**def** query(groups: list[Group], students: list[Student]) -> list[tuple[str, list[str]]]:

*# Сортировка студентов по group\_id*

sorted\_students = sorted(students, key=**lambda** s: s.group\_id)

*# Группировка студентов по group\_id*

group\_students = groupby(sorted\_students, **lambda** s: s.group\_id)

*# Определение названии группы по group\_id и фамилии студента*

groups\_dict = {group.id: group.name **for** group **in** groups}

name\_students = [

(groups\_dict[g\_id], [gs.surname **for** gs **in** g\_students])

**for** g\_id, g\_students **in** group\_students

]

**return** name\_students

**if** \_\_name\_\_ == "\_\_main\_\_":

name\_students = query(groups, students)

*# Вывод результата*

**for** name, surnames **in** name\_students:

print(name)

**for** surname **in** surnames:

print(surname)

print()

B15

Petrov

Kozlov

B16

Ivanov

B17

Medvedev

A2.

**from** **dataclasses** **import** dataclass

**from** **itertools** **import** groupby

@dataclass

**class** **Student**:

id: int

surname: str

group\_id: int

score: float

@dataclass

**class** **Group**:

id: int

name: str

groups = [Group(0, "B15"), Group(1, "B16"), Group(2, "B17")]

students = [

Student(0, "Ivanov", 1, 4.2),

Student(1, "Petrov", 0, 3.1),

Student(2, "Medvedev", 2, 4.8),

Student(3, "Kozlov", 0, 5),

]

**def** query(groups: list[Group], students: list[Student]) -> list[tuple[str, float]]:

*# Группировка студентов по group\_id (предварительно отсортировав список по group\_id, так как этого требует функция groupby)*

group\_students = groupby(

sorted(students, key=**lambda** s: s.group\_id), **lambda** s: s.group\_id

)

*# Вычисление суммы score студентов в каждой группе*

group\_sum = [

(g\_id, sum([g\_student.score **for** g\_student **in** g\_students]))

**for** g\_id, g\_students **in** group\_students

]

*# Сортировка групп по возрастанию суммы score*

sorted\_group\_sum = sorted(group\_sum, key=**lambda** g: g[1])

*# Определение названии группы по group\_id*

groups\_dict = {group.id: group.name **for** group **in** groups}

name\_sum = [(groups\_dict[g\_id], g\_sum) **for** g\_id, g\_sum **in** sorted\_group\_sum]

**return** name\_sum

**if** \_\_name\_\_ == "\_\_main\_\_":

name\_sum = query(groups, students)

*# Вывод результата*

**for** name, group\_sum **in** name\_sum:

print(name, group\_sum)

B16 4.2

B17 4.8

B15 8.1

A3.

**from** **dataclasses** **import** dataclass

@dataclass

**class** **Student**:

id: int

surname: str

@dataclass

**class** **Group**:

id: int

name: str

@dataclass

**class** **GroupStudent**:

student\_id: int

group\_id: int

groups = [Group(0, "B15"), Group(1, "B16"), Group(2, "B17")]

students = [

Student(0, "Ivanov"),

Student(1, "Petrov"),

Student(2, "Medvedev"),

Student(3, "Kozlov"),

]

group\_students = [

GroupStudent(3, 1),

GroupStudent(0, 0),

GroupStudent(1, 1),

GroupStudent(2, 2),

GroupStudent(0, 2),

GroupStudent(1, 0),

GroupStudent(3, 0),

GroupStudent(3, 2),

]

**def** query(

groups: list[Group],

students: list[Student],

group\_students: list[GroupStudent],

name: str,

) -> list[tuple[str, list[str]]]:

*# Фильтрация групп содержащих заданное слово в названии*

filtered\_groups = [g **for** g **in** groups **if** name **in** g.name]

*# Получение набора GroupStudent для каждой группы*

groups\_gs = [

(g, [gs **for** gs **in** group\_students **if** gs.group\_id == g.id])

**for** g **in** filtered\_groups

]

*# Фильтрация пустных групп*

groups\_gs = [g **for** g **in** groups\_gs **if** len(g[1]) > 0]

*# Определение названии группы и фамилии студента по student\_id*

students\_dict = {student.id: student.surname **for** student **in** students}

groups\_surnames = [

(g[0].name, [students\_dict[gs.student\_id] **for** gs **in** g[1]]) **for** g **in** groups\_gs

]

**return** groups\_surnames

**if** \_\_name\_\_ == "\_\_main\_\_":

groups\_surnames = query(groups, students, group\_students, "B")

*# Вывод результата*

**for** name, surnames **in** groups\_surnames:

print(name)

**for** surname **in** surnames:

print(surname)

print()

B15

Ivanov

Petrov

Kozlov

B16

Kozlov

Petrov

B17

Medvedev

Ivanov

Kozlov