```
# используется для сортировки
from operator import itemgetter
class Park:
    """Глава"""
    def __init__(self, id, park, page, driver_id):
        self.id = id
        self.park = park
        self.page = page
        self.driver_id = driver_id
class Driver:
    """Книга"""
    def __init__(self, id, name):
        self.id = id
        self.name = name
# Водители
drivers = [
    Driver(11, 'Гриша-камазист'),
    Driver(22, 'Рома'),
    Driver(33, 'Шамиль'),
    Driver(44, 'Никита'),
    Driver(55, 'Саша'),
]

# Автопарки
parks = [
    Park(1, 'Камаз', 6, 11),
    Park(2, 'Урал', 35, 22),
    Park(3, 'Газ', 94, 33),
    Park(4, 'Автоваз', 143, 44),
    Park(5, 'МАЗ', 254, 55),
]
def main():
    """Основная функция"""
    # Соединение данных один-ко-многим
    one_to_many = [(c.park, c.page, b.name)
        for b in drivers
        for c in parks
        if c.driver_id == b.id]
    print('Задание 1')
    res_11 = sorted(one_to_many, key=itemgetter(2))
    print(res_11)

    print('nЗадание 2')
    res_12_unsorted = []
    # Перебираем всех водителей
    for b in drivers:
        c_parks = list(filter(lambda i: i[2]==b.name, one_to_many))
        if len(c_parks) > 0:
            #  уровень водителя
            c_pages = [page for _,page,_ in c_parks]
            # Суммарный уровень
            c_pages_sum = sum(c_pages)
            res_12_unsorted.append((b.name, c_pages_sum))
```

```
            # Сортировка по суммарному уровню
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    print(res_12)
if __name__ == '__main__':
    main()
#***
#Задание 1
#[('СМ', 6, 'Гриша'), ('ЭНЕРГО', 35, 'Миша'), ('АК', 254, 'Рафаэль'),
('МТ', 94, 'Рафик'), ('ИБМ', 143, 'Эрнест')]
#Задание 2
#[('Рафаэль', 254), ('Эрнест', 143), ('Рафик', 94), ('Миша', 35),
('Гриша', 6)]
#***
```