1. Свой map

Напишите свою **функцию** **map(function, values)**, которая из одного **списка** делает другой. Каждый элемент результирующего списка должен получаться применением функции преобразования **functions**  к соответствующему элементу списка **values**. Использовать стандартный map нельзя.

## Пример

| Ввод | Вывод |
| --- | --- |
| values = [1, 3, 1, 5, 7]  operation = lambda x: x + 5  print(\*simple\_map(operation, values)) | 6 8 6 10 12 |

1. filter, map, sum

Посчитайте сумму квадратов всех двузначных чисел, делящихся на 9.

При решении задачи используйте комбинацию функций filter, map, sum.

Примечание: на 9 должно делиться исходное двузначное число, а не его квадрат.

Отправьте на проверку получившееся число и код, его вычисляющий.

1. Гематрия

Словесной гематрией называется сумма номеров (кодов, числовых значений) входящих в слово букв.

На вход программы поступает список английских слов. На одной строке записано одно слово, количество слов неизвестно.

Для вычисления гематрии поступим следующим образом:

1. Переведём слово в верхний регистр.
2. Числовое значение буквы вычислим как **КодБуквы - КодБуквыA** **+ 1**

Выведите полученные слова в порядке возрастания их гематрии. Если для каких-то слов гематрия совпадает, то их выводите в алфавитном порядке.

## Формат ввода

Набор слов на английском языке, каждое слово на отдельной строке.

## Формат вывода

Набор слов в требуемом порядке.

1. Ход конем

Вам дана позиция коня на шахматной доске (например, ‘A2’). Необходимо написать функцию **list\_of\_turns(cell)**, которая возвращает список возможных координат, на которые может попасть конь за один по стандартной шахматной доске. Результат отсортировать по алфавиту.

Рекомендация:

Можно сделать две вспомогательные функции для перевода координат из шахматных в числовые (‘A2’ -> (1, 2)) и наоборот. Кроме этого, можно написать функцию, которая проверяет, является ли клетка внутри шахматного поля или нет.

### Пример 1

| Ввод | Вывод |
| --- | --- |
| print(list\_of\_turns("B1")) | ['A3', 'C3', 'D2'] |

1. Пин-код

Пин-код имеет вид **a-b-c**, где a, b, c – натуральные числа. Число **a** должно быть простым, число **b** должно быть палиндромом, число **c** должно быть степенью двойки.

Напишите функцию **check\_pin(pinCode)**, которая проверяет, является ли пин-код корректным и возвращает необходимый вердикт.

### Пример 1

| Ввод | Вывод |
| --- | --- |
| print(check\_pin('7-101-4')) | Корректен |

### Пример 2

| Ввод | Вывод |
| --- | --- |
| print(check\_pin('12-22-16')) | Некорректен |

1. Расчет стоимости

Напишите функцию calculate\_order\_cost, которая вычисляет итоговую стоимость заказа на основе переданных аргументов.

Функция принимает два обязательных позиционных аргумента: quantity (количество товаров, целое число), price (цена за единицу товара, число) и два именованных аргумента со значениями по умолчанию: discount (скидка в виде десятичной дроби, по умолчанию 0), tax (налог в виде десятичной дроби, по умолчанию 0).

Сначала считается базовая стоимость, затем применяется скидка, к результату налог, итог округляется до двух знаков после запятой.

### Пример 1

| Ввод | Вывод |
| --- | --- |
| # Без скидки и налога  calculate\_order\_cost(5, 10)  # Со скидкой 10%  calculate\_order\_cost(5, 10, discount=0.1)  # С налогом 20%  calculate\_order\_cost(5, 10, tax=0.2)  calculate\_order\_cost(quantity=5, price=10) | 50.0  45.0  60.0  50.0 |

1. Сортировка

Напишите программу для сортировки списка студентов, хранящегося в виде вложенного объекта. Каждый студент представлен словарем с информацией: name (имя, строка), grades (список курсов, где каждый курс — словарь с course\_name и score — оценка от 0 до 100).

Отсортируйте студентов по следующим критериям (в порядке приоритета).

1. По количеству курсов с оценкой **выше 80** (убывание).

2. По средней оценке по всем курсам (убывание).

3. По имени в алфавитном порядке (возрастание).

В задаче нужно использовать lambda-функции.

### Пример 1

| Ввод | Вывод |
| --- | --- |
| students = [  {  "name": "Карл",  "grades": []  },  {  "name": "Боб",  "grades": [  {"course\_name": "Математика", "score": 78},  {"course\_name": "История", "score": 92},  {"course\_name": "Химия", "score": 88}  ]  },  {  "name": "Алиса",  "grades": [  {"course\_name": "Математика", "score": 90},  {"course\_name": "Физика", "score": 85}  ]  }  ]  sorted\_students = sorted(students, key="your\_key")  pprint(sorted\_students) | [  {  "name": "Алиса",  "grades": [  {"course\_name": "Математика", "score": 90},  {"course\_name": "Физика", "score": 85}  ]  },  {  "name": "Боб",  "grades": [  {"course\_name": "Математика", "score": 78},  {"course\_name": "История", "score": 92},  {"course\_name": "Химия", "score": 88}  ]  },  {  "name": "Карл",  "grades": []  }  ] |