Повторение

списки, строки

Задачи

1. На вход программе подается строка из нескольких слов разделенных пробелом. Программа должна вернусь список из этих слов.
2. Дан список из трех слов, введенных пользователем. Программа должна напечатать эти слова в одну строку через запятую (пробел, знак доллара).
3. Дан список чисел. Верните сумму индексов минимального и максимального элемента списка.
4. Дан список. Программа должна напечатать его задом наперед (без квадратных скобок).
5. Напишите программу, которая объединяет два заданных списка в один и выводит полученный список.

Списочные выражения
Генераторы списков

Как добавить число в список?

Как добавить случайное число от 1 до 100 в список?

Как создать список из трех случайных чисел от 1 до 100?

Как создать список из 10 элементов введенных пользователем?

В Python, помимо стандартных конструкций — цикла for и встроенных функций вроде append() или range(), — есть более лаконичный способ создавать списки. Это списочные выражения

```
[выражение for элемент in итерируемый_объект]
```

```
squares = [x**2 for x in range(10)]

print(squares)
```

```
numbers = [int(input()) for i in range(5)]
print(numbers)

from random import *
numbers = [random.randint(1,100) for i in range(3)]
print(numbers)
```

Списочные выражения позволяют быстро и лаконично создавать списки. Это удобно, если вам сразу нужен весь результат — например, список чисел, строк или объектов.

Но в Python есть ещё один способ перебора значений — `генераторы`. Внешне они похожи на списочные выражения, но работают иначе: список создаёт все элементы сразу и хранит их в памяти, а генератор формирует значения по одному, «по запросу». Это экономит ресурсы и полезно при работе с большими объёмами данных.

```
numbers = [int(input()) for i in range(5)]

avg = sum(numbers) // len(numbers)

numbers = [element for element in numbers if element > avg]

print(numbers)
```

А что с производительностью?

```
numbers = [int(input()) for i in range(5)]

numbers = [element for element in numbers if element > sum(numbers) // len(numbers)]

print(numbers)
```

здесь кроется ошибка производительности: `sum(numbers)` будет пересчитываться заново `при каждой итерации`. Это значит, что для списка из 5 элементов функция `sum()` вызовется 5 раз, а при 1000 элементах — 1000 раз.

Вместо этого лучше один раз вычислить среднее и сохранить в переменной

Задачи

1. На вход программе подается натуральное число n. Напишите программу, использующую списочное выражение, которая создает список, содержащий кубы чисел от 1 до n, а затем выводит его элементы построчно, то есть каждый на отдельной строке.На вход программе подается натуральное число n. Напишите программу, использующую списочное выражение, которая создает список, содержащий квадраты чисел от 1 до n, а затем выводит его элементы построчно, то есть каждый на отдельной строке.

2. Используя генератора списка создайте список только четных чисел от 0 до 19.

3. На вход программе подается строка текста, содержащая различные натуральные числа. Из данной строки формируется список чисел. Напишите программу, которая меняет местами минимальный и максимальный элемент этого списка.

4. На вход программе подается строка текста, содержащая целые числа. Из данной строки формируется список чисел. Напишите программу, которая сортирует и выводит данный список сначала по возрастанию, а затем по убыванию.

Задачи

5. Создайте список, содержащий первые буквы каждого слова в строке.

6. Есть список строк, создайте новый список, содержащий длины этих строк

7. В вашем распоряжении имеется список names. На основании его при помощи генератора списков создайте новый список, в котором каждый элемент создается по шаблону **My name is <имя>** Полученный список выведите на экран.

8. При помощи **генератора списков** создайте **список слов**, которые **начинаются** с буквы «t» или «T». **Исходные слова возьмите из переменной phrase**. Не забудьте использовать метод **.split()** для разделения фразы на отдельные слова. Программа должна **вывести список слов**, которые начинаются с «t» или «T», сохраняя **тот же порядок**, в котором они встречались в изначальной фразе.