11.1 Введение в списки

Шаг 1

Тема урока: введение в списки

1. Создание списков
2. Пустые списки
3. Встроенная функция `list()`
4. Вывод списков

Аннотация. Списки как сохранение последовательностей и аналог массивов.

Списки

В предыдущих уроках мы работали с последовательностями чисел, символов, строк, но не сохраняли всю последовательность в памяти компьютера, а обрабатывали ее поэлементно, считывая раз за разом новый элемент. Однако во многих задачах требуется **сохранять всю последовательность**. Например, классическая задача сортировки (https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B8) (упорядочения) некоторой последовательности требует сохранения всех данных в памяти компьютера. Увы, не сохранив, их невозможно отсортировать. И тут на помощь приходит структура данных, которая в большинстве языков программирования называется массивом (https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)). В Python она называется **списком**.

Структура данных (data structure) — программная единица, позволяющая **хранить и обрабатывать** множество однотипных и/или логически связанных данных.

Список представляет собой последовательность элементов, пронумерованных от 0, как символы в строке.

Создание списка

Чтобы создать список, нужно перечислить его элементы через запятую в квадратных скобках:

```
numbers = [2, 4, 6, 8, 10]
languages = ['Python', 'C#', 'C++', 'Java']
```

Список `numbers` состоит из 5 элементов, и каждый из них — целое число.

- `numbers[0] == 2;`
- `numbers[1] == 4;`
- `numbers[2] == 6;`
- `numbers[3] == 8;`
- `numbers[4] == 10.`

Список `languages` состоит из 4 элементов, каждый из которых — **строка.**

- `languages[0] == 'Python';`
- `languages[1] == 'C#';`
- `languages[2] == 'C++';`
- `languages[3] == 'Java'.`

Значения, заключенные в квадратные скобки и отделенные запятыми, называются **элементами списка**.

Список может содержать значения **разных типов данных**:

```
info = ['Timur', 1992, 61.5]
```

Список `info` содержит строковое значение, целое число и число с плавающей точкой.

- `info[0] == 'Timur';`
- `info[1] == 1992;`
- `info[2] == 61.5.`

Обычно элементы списка содержат данные одного типа, и на практике редко приходится создавать списки, содержащие элементы разных типов данных.

Пустой список

Создать пустой список можно двумя способами:

1. Использовать пустые квадратные скобки `[]`;
2. Использовать встроенную функцию, которая называется `list`.

Следующие две строки кода создают пустой список:

```
mylist = []  # пустой список
mylist = list()  # тоже пустой список
```

Вывод списка

Для вывода всего списка можно применить функцию `print()`:

```
numbers = [2, 4, 6, 8, 10]
languages = ['Python', 'C#', 'C++', 'Java']
print(numbers)
print(languages)
```

Функция `print()` выводит на экран элементы списка, в квадратных скобках, разделенные запятыми:

```
[2, 4, 6, 8, 10]
['Python', 'C#', 'C++', 'Java']
```

Обратите внимание, что вывод списка содержит квадратные скобки. Позже мы научимся выводить элементы списка в более удобном виде с помощью циклов или с помощью распаковки.

Встроенная функция list

Python имеет встроенную функцию `list()`, которая помимо создания пустого списка может преобразовывать некоторые типы объектов в списки.

Например, мы знаем, что функция `range()` создает последовательность целых чисел в заданном диапазоне. Для преобразования этой последовательности в список, мы пишем следующий код:

```
numbers = list(range(5))
```

Во время исполнения этого кода происходит следующее:

1. Вызывается функция `range()`, в которую в качестве аргумента передается число 5;
2. Эта функция возвращает последовательность чисел `0, 1, 2, 3, 4`;
3. Последовательность чисел `0, 1, 2, 3, 4` передается в качестве аргумента в функцию `list()`;
4. Функция `list()` возвращает список `[0, 1, 2, 3, 4]`;
5. Список `[0, 1, 2, 3, 4]` присваивается переменной `numbers`.

Вот еще один пример:

```
even_numbers = list(range(0, 10, 2))  # список содержит четные числа 0, 2, 4, 6, 8
odd_numbers = list(range(1, 10, 2))  # список содержит нечетные числа 1, 3, 5, 7, 9
```

Точно так же с помощью функции `list()` мы можем создать список из символов строки. Для преобразования строки в список мы пишем следующий код:

```
s = 'abcde'
chars = list(s)  # список содержит символы 'a', 'b', 'c', 'd', 'e'
```

Во время исполнения этого кода происходит следующее:

1. Вызывается функция `list()`, в которую в качестве аргумента передается строка `'abcde'`;
2. Функция `list()` возвращает список `['a', 'b', 'c', 'd', 'e']`;
3. Список `['a', 'b', 'c', 'd', 'e']` присваивается переменной `chars`.

Примечания

Примечание 1. Как уже было сказано, списки в Python аналогичны массивам в других языках программирования. Однако разница между списками и массивами все же существует. Элементы массива всегда имеют одинаковый тип данных и располагаются в памяти компьютера непрерывным блоком, а элементы списка могут быть разбросаны по памяти как угодно и могут иметь разный тип данных.

Примечание 2. Обратите внимание, при выводе содержимого списка с помощью функции `print()`, все строковые элементы списка обрамляются **одинарными кавычками**. Если требуется осуществить вывод в двойных кавычках, нужно самостоятельно писать код вывода.

❤️ Happy Pythoning!

Шаг 2

Значения в списках, заключённые в квадратные скобки и отделённые запятыми, называются

Чтобы решить это задание откройте https://stepik.org/lesson/324750/step/2

Шаг 3

Из скольких элементов состоит список `numbers`?

```
numbers = [3, 5, 7, 9]
```

Чтобы решить это задание откройте https://stepik.org/lesson/324750/step/3

Шаг 4

Какой индекс у числа 17 в списке `numbers`?

```
numbers = [1, 100, 7, 20, 17, 37, 22]
```

Чтобы решить это задание откройте https://stepik.org/lesson/324750/step/4

Шаг 5

Может ли список в Python содержать значения разных типов данных?

Чтобы решить это задание откройте https://stepik.org/lesson/324750/step/5

Шаг 6

Что покажет приведённый ниже код?

```
numbers = [0, 1, 3, 14, 2, 7, 9, 8, 10]
print(numbers)
```

Чтобы решить это задание откройте https://stepik.org/lesson/324750/step/6

Шаг 7

Что покажет приведённый ниже код?

```
names = ['Michael', 'John', 'Freddie']
print(names)
```

Чтобы решить это задание откройте https://stepik.org/lesson/324750/step/7

Шаг 8

Список чисел

На вход программе подаётся одно число n. Напишите программу, которая выводит список `[1, 2, 3, ..., n]`.

Формат входных данных

На вход программе подаётся одно натуральное число.

Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

▶ Тестовые данные

Sample Input 1:

```
1
```

Sample Output 1:

```
[1]
```

Sample Input 2:

```
5
```

Sample Output 2:

```
[1, 2, 3, 4, 5]
```

Sample Input 3:

```
10
```

Sample Output 3:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Чтобы решить это задание откройте https://stepik.org/lesson/324750/step/8

Шаг 9

Список букв

На вход программе подаётся натуральное число n. Напишите программу, которая выводит список, состоящий из n букв английского алфавита `['a', 'b', 'c', ...]` в нижнем регистре.

Формат входных данных

На вход программе подаётся натуральное число n $(1 \leq n \leq 26)$.

Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

▶ Тестовые данные

Sample Input 1:

```
1
```

Sample Output 1:

```
['a']
```

Sample Input 2:

```
5
```

Sample Output 2:

```
['a', 'b', 'c', 'd', 'e']
```

Sample Input 3:

```
10
```

Sample Output 3:

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
```

Чтобы решить это задание откройте https://stepik.org/lesson/324750/step/9

11.2 Основы работы со списками

Шаг 1

Тема урока: основы работы со списками

1. Встроенные функции `len()`, `sum()`, `min()`, `max()`
2. Оператор принадлежности `in`
3. Индексация и срезы
4. Конкатенация и умножение на число
5. Отличие списков от строк

Аннотация. Начинаем работать со списками.

Основы работы со списками

Работа со списками очень сильно напоминает работу со строками, поскольку и списки, и строки содержат отдельные элементы. Однако элементы списка могут иметь произвольный тип, а элементами строк всегда являются символы. Многое из того, что мы делали со строками, доступно и при работе со списками.

Функция len()

Длиной списка называется количество его элементов. Для того чтобы посчитать длину списка, мы используем встроенную функцию `len()` (от слова length – длина).

Приведённый ниже код:

```
numbers = [2, 4, 6, 8, 10]
languages = ['Python', 'C#', 'C++', 'Java']

print(len(numbers))      # выводим длину списка numbers
print(len(languages))    # выводим длину списка languages

print(len(['apple', 'banana', 'cherry']))   # выводим длину списка, состоящего из 3 элементов
```

выводит:

```
5
4
3
```

Оператор принадлежности in

Оператор `in` позволяет проверить, содержит ли список некоторый элемент.

Рассмотрим следующий код:

```
numbers = [2, 4, 6, 8, 10]

if 2 in numbers:
    print('Список numbers содержит число 2')
else:
    print('Список numbers не содержит число 2')
```

Такой код проверяет, содержит ли список `numbers` число 2, и выводит соответствующий текст:

```
Список numbers содержит число 2
```

Мы можем использовать оператор `in` вместе с логическим оператором `not`. Например:

```
numbers = [2, 4, 6, 8, 10]

if 0 not in numbers:
    print('Список numbers не содержит нулей')
```

Индексация

При работе со строками мы использовали **индексацию**, то есть обращение к конкретному символу строки по его индексу. Аналогично, можно индексировать и списки.

Для индексации списков в Python используются квадратные скобки `[]`, в которых указывается индекс (номер) нужного элемента в списке:

Пусть `numbers = [2, 4, 6, 8, 10]`.

Таблица ниже показывает, как работает индексация:

Выражение	Результат	Пояснение
`numbers[0]`	2	первый элемент списка
`numbers[1]`	4	второй элемент списка
`numbers[2]`	6	третий элемент списка
`numbers[3]`	8	четвертый элемент списка
`numbers[4]`	10	пятый элемент списка

Обратите внимание: первый элемент списка `numbers[0]`, а не `numbers[1]`. Программисты же всё считают с нуля.

Так же, как и в строках, для нумерации с конца разрешены отрицательные индексы.

Выражение	Результат	Пояснение
`numbers[-1]`	10	пятый элемент списка
`numbers[-2]`	8	четвертый элемент списка
`numbers[-3]`	6	третий элемент списка
`numbers[-4]`	4	второй элемент списка
`numbers[-5]`	2	первый элемент списка

Как и в строках, попытка обратиться к элементу списка по несуществующему индексу:

```
print(numbers[17])
```

приводит к возникновению ошибки:

```
IndexError: index out of range
```

Срезы

Рассмотрим список `numbers = [2, 4, 6, 8, 10]`.

С помощью среза мы можем получить несколько элементов списка, создав диапазон индексов, разделенных двоеточием `numbers[x:y]`.

Приведённый ниже код:

```
print(numbers[1:3])
print(numbers[2:5])
```

выводит:

```
[4, 6]
[6, 8, 10]
```

При построении среза `numbers[x:y]` первое число – это то место, где начинается срез (**включительно**), а второе – это место, где заканчивается срез (**невключительно**). Разрезая списки, мы создаем новые списки, по сути, подсписки исходного.

При использовании срезов со списками мы также можем опускать второй параметр в срезе `numbers[x:]` (но поставить двоеточие), тогда срез берется до конца списка. Аналогично если опустить первый параметр `numbers[:y]`, то можно взять срез от начала списка.

Срез `numbers[:]` возвращает копию исходного списка.

Как и в строках, мы можем использовать отрицательные индексы в срезах списков.

Использование срезов для изменения элементов в заданном диапазоне

Для изменения целого диапазона элементов списка можно использовать срезы. Например, если мы хотим перевести на русский язык названия фруктов `'banana', 'cherry', 'kiwi'`, то это можно сделать с помощью среза.

Приведённый ниже код:

```
fruits = ['apple', 'apricot', 'banana', 'cherry', 'kiwi', 'lemon', 'mango']
fruits[2:5] = ['банан', 'вишня', 'киви']

print(fruits)
```

выводит:

```
['apple', 'apricot', 'банан', 'вишня', 'киви', 'lemon', 'mango']
```

Операция конкатенации + и умножения на число *

Мы можем применять операторы `+` и `*` для списков подобно тому, как мы это делали со строками.

Приведённый ниже код:

```
print([1, 2, 3, 4] + [5, 6, 7, 8])
print([7, 8] * 3)
print([0] * 10)
```

выводит:

```
[1, 2, 3, 4, 5, 6, 7, 8]
[7, 8, 7, 8, 7, 8]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Для генерации списков, состоящих строго из повторяющихся элементов, умножение на число — самый короткий и правильный метод.

Мы также можем использовать расширенные операторы `+=` и `*=` при работе со списками.

Приведённый ниже код:

```
a = [1, 2, 3, 4]
b = [7, 8]
a += b   # добавляем к списку a список b
b *= 5   # повторяем список b 5 раз

print(a)
print(b)
```

выводит:

```
[1, 2, 3, 4, 7, 8]
[7, 8, 7, 8, 7, 8, 7, 8, 7, 8]
```

Встроенные функции sum(), min(), max()

Встроенная функция `sum()` принимает в качестве параметра список чисел и вычисляет сумму его элементов.

Приведённый ниже код:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print('Сумма всех элементов списка =', sum(numbers))
```

выводит:

```
Сумма всех элементов списка = 55
```

Встроенные функции `min()` и `max()` принимают в качестве параметра список и находят минимальный и максимальный элементы соответственно.

Приведённый ниже код:

```
numbers = [3, 4, 10, 3333, 12, -7, -5, 4]
print('Минимальный элемент =', min(numbers))
print('Максимальный элемент =', max(numbers))
```

выводит:

```
Минимальный элемент = -7
Максимальный элемент = 3333
```

Отличие списков от строк

Несмотря на всю схожесть списков и строк, есть одно очень важное отличие: строки — **неизменяемые** объекты, а списки – **изменяемые**.

Приведённый ниже код:

```
s = 'abcdefg'
s[1] = 'x'    # пытаемся изменить 2 символ (по индексу 1) строки
```

приводит к возникновению ошибки:

```
object does not support item assignment
```

Приведённый ниже код:

```
numbers = [1, 2, 3, 4, 5, 6, 7]
numbers[1] = 101     # изменяем 2 элемент (по индексу 1) списка
print(numbers)
```

выводит:

```
[1, 101, 3, 4, 5, 6, 7]
```

Запомни: изменять отдельные символы строк нельзя, однако можно изменять отдельные элементы списков. Для этого используем индексатор и оператор присваивания.

Happy Pythoning!

Шаг 2

Используя индексатор, дополните приведённый ниже код так, чтобы он вывел 17-ый (по порядку) элемент списка `primes`.

Чтобы решить это задание откройте https://stepik.org/lesson/296419/step/2

Шаг 3

Используя индексатор, дополните приведённый ниже код так, чтобы он вывел последний элемент списка `primes`.

Чтобы решить это задание откройте https://stepik.org/lesson/296419/step/3

Шаг 4

Используя срезы, дополните приведённый ниже код так, чтобы он вывел первые 6 элементов списка `primes`.

Примечание. Результатом вывода должна быть строка `[2, 3, 5, 7, 11, 13]`.

Чтобы решить это задание откройте https://stepik.org/lesson/296419/step/4

Шаг 5

Дополните приведённый ниже код так, чтобы он вывел **сумму** минимального и максимального элементов списка `numbers`.

Чтобы решить это задание откройте https://stepik.org/lesson/296419/step/5

Шаг 6

Дополните приведённый ниже код так, чтобы он вывел среднее арифметическое элементов списка `evens`.

Чтобы решить это задание откройте https://stepik.org/lesson/296419/step/6

Шаг 7

Дополните приведённый ниже код так, чтобы элемент списка имеющий значение `Green` заменился на значение `Зеленый`, а элемент `Violet` – на `Фиолетовый`. Далее необходимо вывести полученный список.

Чтобы решить это задание откройте https://stepik.org/lesson/296419/step/7

Шаг 8

Дополните приведённый ниже код так, чтобы он вывел "перевёрнутый" список `languages` (то есть элементы будут идти в обратном порядке).

Примечание. Для вывода списка воспользуйтесь функцией `print()`.

Чтобы решить это задание откройте https://stepik.org/lesson/296419/step/8

Шаг 9

Используя операторы конкатенации (`+`) и умножения списка на число (`*`), дополните приведённый ниже код так, чтобы он вывел следующий список:

```
[1, 2, 3, 1, 2, 3, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 8, 9, 10, 11, 12, 13]
```

Чтобы решить это задание откройте https://stepik.org/lesson/296419/step/9

11.3 Методы списков. Часть 1

Шаг 1

Тема урока: методы добавления и удаления элементов

1. Метод добавления элемента `append()`
2. Метод расширения списка `extend()`
3. Оператор `del`
4. Решение задач

Аннотация. Добавление элементов в список. Оператор `del`, удаляющий элементы по заданному индексу.

Добавление элементов

Мы научились создавать статические списки, то есть списки, элементы которых известны на этапе создания. Следующий шаг – научиться добавлять элементы в уже существующие списки.

Метод append()

Для добавления нового элемента **в конец списка** используется метод `append()`.

Приведённый ниже код:

```
numbers = [1, 1, 2, 3, 5, 8, 13]  # создаем список

numbers.append(21)  # добавляем число 21 в конец списка
numbers.append(34)  # добавляем число 34 в конец списка

print(numbers)
```

выводит:

```
[1, 1, 2, 3, 5, 8, 13, 21, 34]
```

Обратите внимание, для того чтобы использовать метод `append()`, нужно, чтобы список был создан (при этом он может быть пустым).

Приведённый ниже код:

```
numbers = []  # создаем пустой список

numbers.append(1)
numbers.append(2)
numbers.append(3)

print(numbers)
```

выводит:

```
[1, 2, 3]
```

Важно: мы не можем использовать индексаторы для установки значений элементов списка, если список пустой.

Приведённый ниже код:

```
numbers = []  # создаем пустой список

numbers[0] = 1
numbers[1] = 2
numbers[2] = 3

print(numbers)
```

приводит к возникновению ошибки:

```
IndexError: list assignment index out of range
```

Метод extend()

Можно также расширить список другим списком путём вызова метода `extend()`.

Приведённый ниже код:

```
numbers = [0, 2, 4, 6, 8, 10]
odds = [1, 3, 5, 7]

numbers.extend(odds)
print(numbers)
```

выводит:

```
[0, 2, 4, 6, 8, 10, 1, 3, 5, 7]
```

Метод `extend()` как бы расширяет один список, добавляя к нему элементы другого списка.

Отличие между методами `append()` и `extend()` проявляется при добавлении строки к списку.

Приведённый ниже код:

```
words1 = ['iq option', 'stepik', 'beegeek']
words2 = ['iq option', 'stepik', 'beegeek']

words1.append('python')
words2.extend('python')

print(words1)
print(words2)
```

выводит:

```
['iq option', 'stepik', 'beegeek', 'python']
['iq option', 'stepik', 'beegeek', 'p', 'y', 't', 'h', 'o', 'n']
```

Метод `append()` добавляет строку `'python'` целиком к списку, а метод `extend()` разбивает строку `'python'` на символы `'p'`, `'y'`, `'t'`, `'h'`, `'o'`, `'n'` и их добавляет в качестве элементов списка.

Удаление элементов

С помощью оператора `del` можно удалять элементы списка по определенному индексу.

Приведённый ниже код:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
del numbers[5]    # удаляем элемент, имеющий индекс 5

print(numbers)
```

выводит:

```
[1, 2, 3, 4, 5, 7, 8, 9]
```

Элемент под указанным индексом удаляется, а список перестраивается.

Обратите внимание на синтаксис удаления, так как он отличается от обычного вызова метода. При удалении элементов не надо передавать аргумент внутри круглых скобок.

Оператор `del` работает и со срезами: мы можем удалить целый диапазон элементов списка.

Приведённый ниже код:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
del numbers[2:7]    # удаляем элементы с 2 по 6 включительно

print(numbers)
```

выводит:

```
[1, 2, 8, 9]
```

Мы можем удалить все элементы на чётных позициях (`0` , `2` , `4` , `...`) исходного списка.

Приведённый ниже код:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
del numbers[::2]

print(numbers)
```

выводит:

```
[2, 4, 6, 8]
```

❤️ Happy Pythoning!

Шаг 2

Допустим, программа состоит из строки кода:

```
names = []
```

Какую из приведённых ниже инструкций следует применить для добавления в список по 0 индексу строкового значения `'Chromatica'` ?

Чтобы решить это задание откройте https://stepik.org/lesson/327207/step/2

Шаг 3

Что покажет приведённый ниже код?

```
numbers = [4, 2, 8, 6, 5]
numbers.append(7)
numbers.append(1)

print(numbers)
```

Чтобы решить это задание откройте https://stepik.org/lesson/327207/step/3

Шаг 4

Что покажет приведённый ниже код?

```
numbers = [4, 2]
numbers.extend([1, 2, 3])
numbers.extend([7, 17, 777])

print(numbers)
```

Чтобы решить это задание откройте https://stepik.org/lesson/327207/step/4

Шаг 5

Что покажет приведённый ниже код?

```
colors = ['red', 'orange', 'yellow', 'green', 'blue', 'purple', 'brown', 'magenta']
del colors[2]
del colors[6]

print(colors)
```

Чтобы решить это задание откройте https://stepik.org/lesson/327207/step/5

Шаг 6

Все сразу 1 🌶️

Дополните приведённый ниже код, чтобы он:

1. Вывел длину списка;
2. Вывел последний элемент списка;
3. Вывел список в обратном порядке (вспоминаем срезы);
4. Вывел «YES» (без кавычек), если список содержит числа 5 и 17, или «NO» (без кавычек) в противном случае;
5. Вывел список с удалёнными первым и последним элементами.

Примечание. Каждый вывод осуществлять с новой строки.

Чтобы решить это задание откройте https://stepik.org/lesson/327207/step/6

Шаг 7

Список строк

На вход программе подаются натуральное число n, а затем n строк. Напишите программу, которая создаёт из указанных строк список, а затем выводит его.

Формат входных данных
На вход программе подаются натуральное число n, а затем n строк, каждая на отдельной строке.

Формат выходных данных
Программа должна вывести список, состоящий из указанных строк.

▶ Тестовые данные

Sample Input:

```
5
C#
C++
C
Python
F#
```

Sample Output:

```
['C#', 'C++', 'C', 'Python', 'F#']
```

Чтобы решить это задание откройте https://stepik.org/lesson/327207/step/7

Шаг 8

Алфавит

Напишите программу, выводящую следующий список:

```
['a', 'bb', 'ccc', 'dddd', 'eeeee', 'ffffff', ...]
```

Формат выходных данных
Программа должна вывести указанный список.

Примечание. Последний элемент списка должен состоять из 26 символов `z`.

Чтобы решить это задание откройте https://stepik.org/lesson/327207/step/8

Шаг 9

Список кубов

На вход программе подаются натуральное число n, а затем n целых чисел. Напишите программу, которая создаёт из указанных чисел список их кубов, а затем выводит его.

Формат входных данных

На вход программе подаются натуральное число n, а затем n целых чисел, каждое на отдельной строке.

Формат выходных данных

Программа должна вывести список, состоящий из кубов указанных чисел.

▶ Тестовые данные

Sample Input 1:

```
5
1
2
3
4
5
```

Sample Output 1:

```
[1, 8, 27, 64, 125]
```

Sample Input 2:

```
2
-5
-2
```

Sample Output 2:

```
[-125, -8]
```

Sample Input 3:

```
1
100
```

Sample Output 3:

```
[1000000]
```

Чтобы решить это задание откройте https://stepik.org/lesson/327207/step/9

Шаг 10

Список делителей

На вход программе подаётся натуральное число n. Напишите программу, которая создаёт список, состоящий из делителей введённого числа, а затем выводит его.

Формат входных данных

На вход программе подаётся натуральное число n.

Формат выходных данных

Программа должна вывести список, состоящий из делителей введённого числа.

▶ Тестовые данные

Sample Input 1:

```
17
```

Sample Output 1:

```
[1, 17]
```

Sample Input 2:

```
25
```

Sample Output 2:

```
[1, 5, 25]
```

Sample Input 3:

```
36
```

Sample Output 3:

```
[1, 2, 3, 4, 6, 9, 12, 18, 36]
```

Чтобы решить это задание откройте https://stepik.org/lesson/327207/step/10

Шаг 11

Суммы двух

На вход программе подаётся натуральное число n $(n \geq 2)$. Затем поступают n целых чисел. Напишите программу, которая создаёт из указанных чисел список, состоящий из сумм соседних чисел (0 и 1, 1 и 2, 2 и 3 и так далее).

Формат входных данных

На вход программе подаются натуральное число n $(n \geq 2)$, а затем n целых чисел, каждое на отдельной строке.

Формат выходных данных

Программа должна вывести список, состоящий из сумм соседних чисел.

▶ Тестовые данные

Sample Input 1:

```
5
1
2
3
4
5
```

Sample Output 1:

```
[3, 5, 7, 9]
```

Sample Input 2:

```
2
10
9
```

Sample Output 2:

```
[19]
```

Чтобы решить это задание откройте https://stepik.org/lesson/327207/step/11

Удалите нечётные индексы

На вход программе подаются натуральное число n, а затем n целых чисел. Напишите программу, которая создаёт из указанных чисел список, затем удаляет все элементы, стоящие по нечётным индексам, а затем выводит полученный список.

Формат входных данных

На вход программе подаются натуральное число n, а затем n целых чисел, каждое на отдельной строке.

Формат выходных данных

Программа должна вывести список в соответствии с условием задачи.

Примечание. Используйте оператор `del`.

▶ Тестовые данные

Sample Input 1:

```
10
0
1
2
3
4
5
6
7
8
9
```

Sample Output 1:

```
[0, 2, 4, 6, 8]
```

Sample Input 2:

```
1
8
```

Sample Output 2:

```
[8]
```

Sample Input 3:

```
2
9
6
```

Sample Output 3:

```
[9]
```

Шаг 13

k-ая буква слова

На вход программе подаются натуральное число n и n строк, а затем число k. Напишите программу, которая выводит k -ую букву из введённых строк на одной строке без пробелов.

Формат входных данных
На вход программе подаются натуральное число n, далее n строк, каждая на отдельной строке. В конце вводится натуральное число k – номер буквы (нумерация начинается с единицы).

Формат выходных данных
Программа должна вывести текст в соответствии с условием задачи.

Примечание. Если некоторые строки слишком короткие и в них нет символа с заданным номером, то такие строки при выводе нужно игнорировать.

▶ Тестовые данные

Sample Input 1:

```
5
abcdef
bcdefg
cdefgh
defghi
efghij
2
```

Sample Output 1:

```
bcdef
```

Sample Input 2:

```
5
aaaaa
bbbb
ccc
dd
e
3
```

Sample Output 2:

```
abc
```

Чтобы решить это задание откройте https://stepik.org/lesson/327207/step/13

Шаг 14

Символы всех строк

На вход программе подаются натуральное число n, а затем n строк. Напишите программу, которая создает список из символов всех строк, а затем выводит его.

Формат входных данных

На вход программе подаются натуральное число n, а затем n строк, каждая на отдельной строке.

Формат выходных данных

Программа должна вывести список, состоящий из символов всех введённых строк.

Примечание. В результирующем списке могут содержаться одинаковые символы.

▶ Тестовые данные

Sample Input:

```
3
abc
def
ghi
```

Sample Output:

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
```

11.4 Вывод элементов списка

Шаг 1

Тема урока: вывод элементов списка

1. Вывод списка с помощью `for`
2. Вывод списка с помощью распаковки
3. Вывод строки с помощью распаковки

Аннотация. Вывод элементов списка и строки.

При выводе содержимого списка с помощью функции `print()` вывод элементов осуществляется в квадратных скобках, причём все элементы разделены запятой. Такой вывод не всегда удобен и предпочтителен, поэтому нужно уметь выводить элементы списка нужным нам способом.

Вывод с помощью цикла for

Для вывода элементов списка **каждого на отдельной строке** можно использовать следующий код:

Вариант 1. Если нужны индексы элементов:

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for i in range(len(numbers)):
    print(numbers[i])
```

Мы передаем в функцию `range()` длину списка `len(numbers)`. В нашем случае длина списка `numbers` равна `11`. Таким образом, вызов функции `range(len(numbers))` имеет вид `range(11)`, и переменная цикла `i` последовательно перебирает все значения от `0` до `10`. Это означает, что выражение `numbers[i]` последовательно вернёт все элементы списка `numbers`. Такой способ итерации списка удобен, когда нам нужен не только сам элемент `numbers[i]`, но и его индекс `i`.

Вариант 2. Если индексы не нужны:

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for num in numbers:
    print(num)
```

Этот цикл пройдёт по списку `numbers`, придавая переменной цикла `num` значение каждого элемента списка (!) в отличие от предыдущего цикла, в котором переменная цикла «бегала» по индексам списка.

Если требуется выводить элементы списка на одной строке через пробел, то мы можем использовать необязательный параметр `end` функции `print()`:

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for num in numbers:
    print(num, end=' ')
```

Вывод с помощью распаковки списка

В Python есть удобный способ вывода элементов списка без использования цикла `for`.

Вариант 1. Вывод элементов списка через один символ пробела:

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print(*numbers)
```

Приведённый выше код выводит:

```
0 1 2 3 4 5 6 7 8 9 10
```

Вариант 2. Вывод элементов списка, каждого на отдельной строке:

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print(*numbers, sep='\n')
```

Приведённый выше код выводит:

```
0
1
2
3
4
5
6
7
8
9
10
```

Поскольку строки содержат символы, подобно тому, как списки содержат элементы, то мы можем использовать распаковку строк точно так же, как и распаковку списков.

Приведённый ниже код:

```
s = 'Python'

print(*s)
print()
print(*s, sep='\n')
```

выводит

```
P y t h o n

P
y
t
h
o
n
```

❤️ Happy Pythoning! 🐍

Шаг 2

Дополните приведённый ниже код так, чтобы он вывел сумму квадратов элементов списка `numbers`.

Чтобы решить это задание откройте https://stepik.org/lesson/328948/step/2

Значение функции

На вход программе подаются натуральное число n, а затем n целых чисел. Напишите программу, которая для каждого введённого числа x выводит значение функции $f(x) = x^2 + 2x + 1$, каждое на отдельной строке.

Формат входных данных

На вход программе подаются натуральное число n, а затем n целых чисел, каждое на отдельной строке.

Формат выходных данных

Программа должна вывести сначала введённые числа, каждое на отдельной строке, затем пустую строку, а затем соответствующие значения функции, каждое на отдельной строке.

Примечание. Для первого теста имеем:

$$f(1) = 1^2 + 2 \cdot 1 + 1 = 4,\ f(2) = 2^2 + 2 \cdot 2 + 1 = 9,\ f(3) = 3^2 + 2 \cdot 3 + 1 = 16,\ \ldots$$

▶ Тестовые данные

Sample Input 1:

```
5
1
2
3
4
5
```

Sample Output 1:

```
1
2
3
4
5

4
9
16
25
36
```

Sample Input 2:

```
3
10
20
30
```

Sample Output 2:

```
10
20
30

121
441
961
```

Чтобы решить это задание откройте https://stepik.org/lesson/328948/step/3

Шаг 4

Remove outliers

При анализе данных, собранных в рамках научного эксперимента, бывает полезно удалить самое большое и самое маленькое значение.

На вход программе подаются натуральное число n, а затем n **различных** натуральных чисел. Напишите программу, которая удаляет наименьшее и наибольшее значение из указанных чисел, а затем выводит оставшиеся числа каждое на отдельной строке, не меняя их порядок.

Формат входных данных
На вход программе подаются натуральное число n, а затем n различных натуральных чисел, каждое на отдельной строке.

Формат выходных данных
Программа должна вывести текст в соответствии с условием задачи.

▶ Тестовые данные

Sample Input:

```
10
9
17
189
3
55
78
11
7
888
160
```

Sample Output:

```
9
17
189
55
78
11
7
160
```

Чтобы решить это задание откройте https://stepik.org/lesson/328948/step/4

Шаг 5

Без дубликатов

На вход программе подаются натуральное число n, а затем n строк. Напишите программу, которая выводит только уникальные строки, в том же порядке, в котором они были введены.

Формат входных данных
На вход программе подаются натуральное число n, а затем n строк, каждая на отдельной строке.

Формат выходных данных
Программа должна вывести текст в соответствии с условием задачи.

Примечание. Считайте, что все строки состоят из строчных символов.

▶ Тестовые данные

Sample Input:

```
5
first
second
first
third
second
```

Sample Output:

```
first
second
third
```

Чтобы решить это задание откройте https://stepik.org/lesson/328948/step/5

Шаг 6

Google search - 1

На вход программе подаются натуральное число n, затем n строк, затем ещё одна строка – поисковый запрос. Напишите программу, которая выводит все введённые строки, в которых встречается поисковый запрос.

Формат входных данных
На вход программе подаются натуральное число n – количество строк, затем сами строки в указанном количестве, затем один поисковый запрос.

Формат выходных данных
Программа должна вывести все введённые строки, в которых встречается поисковый запрос.

Примечание. Поиск не должен быть чувствителен к регистру символов.

▶ Тестовые данные

Sample Input:

```
5
Язык Python прекрасен
C# - отличный язык программирования
Stepik - отличная платформа
BEEGEEK FOREVER!
язык Python появился 20 февраля 1991
язык
```

Sample Output:

```
Язык Python прекрасен
C# - отличный язык программирования
язык Python появился 20 февраля 1991
```

Чтобы решить это задание откройте https://stepik.org/lesson/328948/step/6

Шаг 7

Google search - 2 🌶️🌶️

На вход программе подаются натуральное число n, затем n строк, затем число k – количество поисковых запросов, затем k строк – поисковые запросы. Напишите программу, которая выводит все введённые строки, в которых встречаются одновременно **все** поисковые запросы.

Формат входных данных
На вход программе подаются натуральное число n – количество строк, затем сами строки в указанном количестве, затем число k, затем сами поисковые запросы.

Формат выходных данных
Программа должна вывести все введённые строки, в которых встречаются **все** поисковые запросы.

Примечание. Поиск не должен быть чувствителен к регистру символов.

▶ Тестовые данные 🟢

Sample Input:

```
5
Язык Python прекрасен
C# - отличный язык программирования
Stepik - отличная платформа
BEEGEEK FOREVER!
язык Python появился 20 февраля 1991
2
язык
python
```

Sample Output:

```
Язык Python прекрасен
язык Python появился 20 февраля 1991
```

Чтобы решить это задание откройте https://stepik.org/lesson/328948/step/7

Negatives, Zeros and Positives

На вход программе подаются натуральное число n, а затем n целых чисел. Напишите программу, которая сначала выводит все отрицательные числа, затем нули, а затем все положительные числа, каждое на отдельной строке. Числа должны быть выведены в том же порядке, в котором они были введены.

Формат входных данных
На вход программе подаются натуральное число n, а затем n целых чисел, каждое на отдельной строке.

Формат выходных данных
Программа должна вывести текст в соответствии с условием задачи.

▶ Тестовые данные

Sample Input 1:

```
7
3
-4
1
0
-1
0
-2
```

Sample Output 1:

```
-4
-1
-2
0
0
3
1
```

Sample Input 2:

```
5
4
3
-2
-10
0
```

Sample Output 2:

```
-2
-10
0
4
3
```

11.5 Методы строк: split(), join()

Шаг 1

Тема урока: строковые методы

1. Метод `split()`
2. Метод `join()`

Аннотация. Строковые методы `split()` и `join()`.

В предыдущем модуле мы детально изучили основные строковые методы, однако обошли стороной два важных: `split()` и `join()`, имеющих отношение к спискам. Они как бы противоположны по смыслу: метод `split()` разбивает строку по произвольному разделителю на список слов, а метод `join()` собирает строку из списка слов через заданный разделитель.

Метод split()

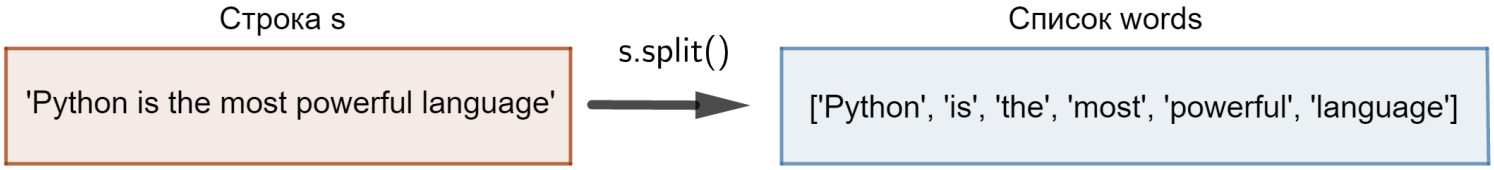
Метод `split()` разбивает строку на слова, используя в качестве разделителя последовательность пробельных символов, и возвращает список из этих слов.

Приведённый ниже код:

```
s = 'Python is the most powerful language'
words = s.split()
print(words)
```

выводит:

```
['Python', 'is', 'the', 'most', 'powerful', 'language']
```



Таким образом, вызов метода `split()` разбивает **строку** на слова и возвращает **список**, содержащий все слова.

Рассмотрим следующий программный код:

```
numbers = input().split()
```

Если при запуске этой программы ввести строку `1 2 3 4 5`, то список `numbers` будет следующим `['1', '2', '3', '4', '5']`. Обратите внимание, что список будет состоять из строк (тип `str`), а не из чисел (тип `int`). Если требуется получить именно список чисел, то затем нужно элементы списка по одному преобразовать в числа с помощью команды `int()`.

Приведённый ниже код:

```
numbers = input().split()
for i in range(len(numbers)):
    numbers[i] = int(numbers[i])
print(numbers)
```

выводит (если на вход программе была подана строка `1 2 3 4 5`):

```
[1, 2, 3, 4, 5]
```

Необязательный параметр

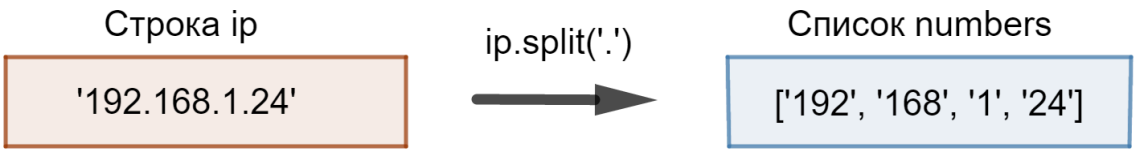
У метода `split()` есть необязательный параметр, который определяет, какой набор символов будет использоваться в качестве разделителя между элементами списка. Например, вызов метода `split('.')` вернёт список, полученный разделением исходной строки по символу `'.'`.

Приведённый ниже код:

```
ip = '192.168.1.24'
numbers = ip.split('.')    # указываем явно разделитель
print(numbers)
```

выводит:

```
['192', '168', '1', '24']
```



Метод join()

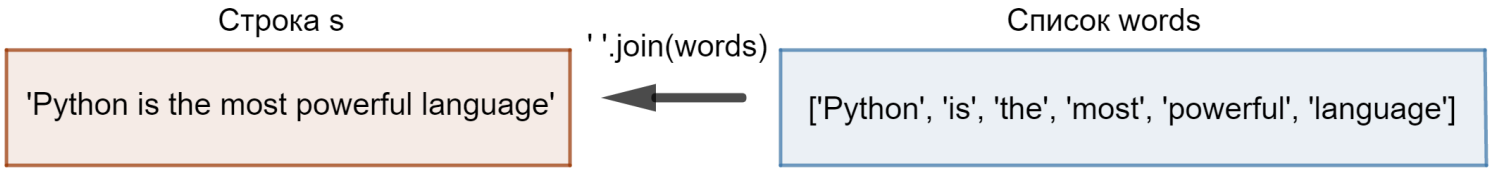
Метод `join()` собирает строку из элементов списка, используя в качестве разделителя строку, к которой применяется метод.

Приведённый ниже код:

```
words = ['Python', 'is', 'the', 'most', 'powerful', 'language']
s = ' '.join(words)
print(s)
```

выводит:

```
Python is the most powerful language
```



Обратите внимание: все слова разделены одним пробелом, поскольку метод `join()` вызывался на строке, состоящей из одного символа пробела ` `.

Рассмотрим ещё пару примеров:

```
words = ['Мы', 'учим', 'язык', 'Python']
print('*'.join(words))
print('-'.join(words))
print('?'.join(words))
print('!'.join(words))
print('*****'.join(words))
print('abc'.join(words))
print('123'.join(words))
```

Приведённый выше код выводит:

```
Мы*учим*язык*Python
Мы-учим-язык-Python
Мы?учим?язык?Python
Мы!учим!язык!Python
Мы*****учим*****язык*****Python
МыabcучимabcязыкabcPython
Мы123учим123язык123Python
```

Запомни: строковый метод `split()` служит для преобразования строки в список, а метод `join()` — для преобразования списка в строку.

Примечания

Примечание 1. Существует большая разница между результатами вызова методов `s.split()` и `s.split(' ')`. Разница в поведении проявляется, когда строка содержит несколько пробелов между словами.

Приведённый ниже код:

```
s = 'I love  Python'
words1 = s.split()
words2 = s.split(' ')
print(words1)
print(words2)
```

выводит:

```
['I', 'love', 'Python']
['I', 'love', '', 'Python']
```

Если в строке встречаются последовательные разделители, то они будут считаться как отдельные разделители, и между ними будет создана пустая строка (`''`).

Примечание 2. Методы `split()` и `join()` являются строковыми методами, они не могут применяться к спискам!

Приведённый ниже код:

```
print([1, 2].split())
```

приводит к возникновению ошибки:

```
AttributeError: 'list' object has no attribute 'split'
```

Примечание 3. Строковый метод `join()` не работает для списков, у которых элементы не строкового типа данных.

Приведённый ниже код:

```
numbers = [1, 2, 3, 4]  # список чисел
s = '*'.join(numbers)
print(s)
```

приводит к возникновению ошибки:

```
TypeError: sequence item 0: expected str instance, int found
```

Примечание 4. На самом деле, метод `join()` в качестве аргумента может принимать не только список, но и строку.

Приведённый ниже код:

```
s = '+'.join('pygen')
print(s)
```

выводит:

```
p+y+g+e+n
```

❤️ **Happy Pythoning!**

Шаг 2

Заполните пропуски.

Чтобы решить это задание откройте https://stepik.org/lesson/324755/step/2

Шаг 3

Что покажет приведённый ниже код?

```
s = 'a     b c'
print(s.split())
```

Чтобы решить это задание откройте https://stepik.org/lesson/324755/step/3

Шаг 4

Что покажет приведённый ниже код?

```
s = 'a     b c'
print(s.split(' '))
```

Чтобы решить это задание откройте https://stepik.org/lesson/324755/step/4

Шаг 5

Что покажет приведённый ниже код?

```
print('-'.join(['pen', 'pineapple', 'apple', 'pen']))
```

Чтобы решить это задание откройте https://stepik.org/lesson/324755/step/5

Шаг 6

Что покажет приведённый ниже код?

```
letters = ['B', ' ', 'T', ' ', 'S']
print(letters.split())
```

Чтобы решить это задание откройте https://stepik.org/lesson/324755/step/6

Шаг 7

Что покажет приведённый ниже код?

```
print('-'.join('DNA'))
```

Чтобы решить это задание откройте https://stepik.org/lesson/324755/step/7

Шаг 8

Что покажет приведённый ниже код?

```
symbols = ['I', 'D', 'O', 'L']
print(symbols.join('-'))
```

Чтобы решить это задание откройте https://stepik.org/lesson/324755/step/8

Шаг 9

С помощью функции `list()` можно из строки получить список ее символов, а с помощью функции `join()` можно склеить элементы списка, вставляя между ними разделитель.

Что покажет приведённый ниже код?

```
s = 'BEEGEEK'
chars = list(s)
s = '**'.join(chars)
print(s)
```

Чтобы решить это задание откройте https://stepik.org/lesson/324755/step/9

Шаг 10

Построчный вывод

На вход программе подаётся строка текста. Напишите программу, которая выводит слова введённой строки в столбик.

Формат входных данных
На вход программе подаётся строка текста.

Формат выходных данных
Программа должна вывести текст в соответствии с условием задачи.

▶ Тестовые данные

Sample Input:

```
У лукоморья дуб зеленый златая цепь на дубе том
```

Sample Output:

```
У
лукоморья
дуб
зеленый
златая
цепь
на
дубе
том
```

Чтобы решить это задание откройте https://stepik.org/lesson/324755/step/10

Шаг 11

Инициалы

На вход программе подаётся строка текста, содержащая имя, отчество и фамилию человека. Напишите программу, которая выводит инициалы человека.

Формат входных данных

На вход программе подаётся строка текста, содержащая имя, отчество и фамилию человека.

Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

▶ Тестовые данные

Sample Input 1:

```
Владимир Семенович Высоцкий
```

Sample Output 1:

```
В.С.В.
```

Sample Input 2:

```
Гуев Тимур Ахсарбекович
```

Sample Output 2:

```
Г.Т.А.
```

Чтобы решить это задание откройте https://stepik.org/lesson/324755/step/11

Шаг 12

Windows OS

В операционной системе Windows полное имя файла состоит из буквы диска, после которого ставится двоеточие и символ `\`, затем через такой же символ перечисляются подкаталоги (папки), в которых находится файл, в конце пишется имя файла (C:\Windows\System32\calc.exe).

На вход программе подаётся одна строка с корректным именем файла в операционной системе Windows. Напишите программу, которая разбирает строку на части, разделенные символом `\`. Каждую часть вывести в отдельной строке.

Формат входных данных
На вход программе подаётся одна строка.

Формат выходных данных
Программа должна вывести текст в соответствии с условием задачи.

Примечание. В Python символ `\` обычно используется для создания специальных символьных последовательностей, которые представляют собой управляющие символы или экранированные последовательности. Например, `\n` представляет символ новой строки, `\t` представляет символ табуляции и так далее. Однако если символ `\` используется как часть строки, его следует экранировать, то есть использовать два обратных слэша `\\`.

▶ Тестовые данные

Sample Input:

```
C:\Windows\System32\calc.exe
```

Sample Output:

```
C:
Windows
System32
calc.exe
```

Чтобы решить это задание откройте https://stepik.org/lesson/324755/step/12

Шаг 13

Диаграмма

На вход программе подаётся строка текста, содержащая целые числа. Напишите программу, которая по заданным числам строит столбчатую диаграмму.

Формат входных данных
На вход программе подаётся строка текста, содержащая целые числа, разделённые символом пробела.

Формат выходных данных
Программа должна вывести столбчатую диаграмму.

▶ Тестовые данные

Sample Input 1:

```
1 2 3 4 5
```

Sample Output 1:

```
+
++
+++
++++
+++++
```

Sample Input 2:

```
5 3 1 7 10 2
```

Sample Output 2:

```
+++++
+++
+
+++++++
++++++++++
++
```

Чтобы решить это задание откройте https://stepik.org/lesson/324755/step/13

Шаг 14

Корректный ip-адрес

На вход программе подаётся строка текста, содержащая 4 целых неотрицательных числа, разделённых точкой. Напишите программу, которая определяет, является ли введённая строка текста корректным ip-адресом.

Формат входных данных
На вход программе подаётся строка текста, содержащая 4 целых числа, разделённых точкой.

Формат выходных данных
Программа должна вывести «ДА» (без кавычек), если введённая строка является корректным ip-адресом, или «НЕТ» (без кавычек) в противном случае.

Примечание. ip-адрес является корректным, если все 4 числа находятся в диапазоне от 0 до 255 **включительно**.

▶ Тестовые данные

Sample Input 1:

```
192.168.0.3
```

Sample Output 1:

```
ДА
```

Sample Input 2:

```
192.168.0.300
```

Sample Output 2:

```
НЕТ
```

Чтобы решить это задание откройте https://stepik.org/lesson/324755/step/14

Шаг 15

Добавь разделитель

На вход программе подаётся строка текста и строка-разделитель. Напишите программу, которая вставляет указанный разделитель между каждым символом введённой строки текста.

Формат входных данных

На вход программе подаются строка текста и строка-разделитель, каждая на отдельной строке.

Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

▶ Тестовые данные

Sample Input 1:

```
1234567
*
```

Sample Output 1:

```
1*2*3*4*5*6*7
```

Sample Input 2:

```
qwerty and password
**
```

Sample Output 2:

```
q**w**e**r**t**y** **a**n**d** **p**a**s**s**w**o**r**d
```

Чтобы решить это задание откройте https://stepik.org/lesson/324755/step/15

Шаг 16

Количество совпадающих пар

На вход программе подаётся строка текста, содержащая целые числа. Из данной строки формируется список чисел. Напишите программу, которая подсчитывает, сколько в полученном списке пар элементов, равных друг другу. Считается, что любые два элемента, равные друг другу, образуют одну пару, которую необходимо посчитать.

Формат входных данных

На вход программе подаётся строка текста, содержащая целые числа, разделённые символом пробела.

Формат выходных данных

Программа должна вывести одно число – количество пар элементов, равных друг другу.

▶ Тестовые данные

Sample Input 1:

```
1 7 5 7 5
```

Sample Output 1:

```
2
```

Sample Input 2:

```
3 3 3 3 3
```

Sample Output 2:

```
10
```

Sample Input 3:

```
8 7 6
```

Sample Output 3:

```
0
```

Чтобы решить это задание откройте https://stepik.org/lesson/324755/step/16

11.6 Методы списков. Часть 2

Шаг 1

Тема урока: методы списков

1. Метод `insert()`
2. Метод `index()`
3. Метод `remove()`
4. Метод `pop()`
5. Метод `reverse()`
6. Метод `count()`
7. Метод `clear()`
8. Метод `copy()`
9. Метод `sort()`

Аннотация. Другие методы списков.

Мы уже познакомились с двумя списочными методами `append()` и `extend()`. Первый добавляет в конец списка один новый элемент, а второй расширяет список другим списком. К спискам в Python применимы и другие удобные методы, с которыми мы познакомимся в этом уроке.

Метод insert()

Метод `insert()` позволяет вставлять значение в список в заданной позиции. В него передается два аргумента:

1. `index` : индекс, задающий место вставки значения;
2. `value` : значение, которое требуется вставить.

Когда значение вставляется в список, список расширяется в размере, чтобы разместить новое значение. Значение, которое ранее находилось в заданной индексной позиции, и все элементы после него сдвигаются на одну позицию к концу списка.

Приведённый ниже код:

```
names = ['Gvido', 'Roman', 'Timur']
print(names)
names.insert(0, 'Anders')
print(names)
names.insert(3, 'Josef')
print(names)
```

выводит:

```
['Gvido', 'Roman', 'Timur']
['Anders', 'Gvido', 'Roman', 'Timur']
['Anders', 'Gvido', 'Roman', 'Josef', 'Timur']
```

Если указан недопустимый индекс, то во время выполнения программы не происходит ошибки. Если задан индекс за пределами конца списка, то значение будет добавлено в конец списка. Если применен отрицательный индекс, который указывает на недопустимую позицию, то значение будет вставлено в начало списка.

Метод index()

Метод `index()` возвращает индекс первого элемента, значение которого равняется переданному в метод значению. Таким образом, в метод передается один параметр:

1. `value` : значение, индекс которого требуется найти.

Если элемент в списке не найден, то во время выполнения происходит ошибка.

Приведённый ниже код:

```
names = ['Gvido', 'Roman', 'Timur']
position = names.index('Timur')
print(position)
```

выводит:

```
2
```

Приведённый ниже код:

```
names = ['Gvido', 'Roman', 'Timur']
position = names.index('Anders')
print(position)
```

приводит к возникновению ошибки:

```
ValueError: 'Anders' is not in list
```

Чтобы избежать таких ошибок, можно использовать метод `index()` вместе с оператором принадлежности `in` :

```
names = ['Gvido', 'Roman', 'Timur']
if 'Anders' in names:
    position = names.index('Anders')
    print(position)
else:
    print('Такого значения нет в списке')
```

Метод remove()

Метод `remove()` удаляет первый элемент, значение которого равняется переданному в метод значению. В метод передается один параметр:

1. `value` : значение, которое требуется удалить.

Метод уменьшает размер списка на один элемент. Все элементы после удаленного элемента смещаются на одну позицию к началу списка. Если элемент в списке не найден, то во время выполнения происходит ошибка.

Приведённый ниже код:

```
food = ['Рис', 'Курица', 'Рыба', 'Брокколи', 'Рис']
print(food)
food.remove('Рис')
print(food)
```

выводит:

```
['Рис', 'Курица', 'Рыба', 'Брокколи', 'Рис']
['Курица', 'Рыба', 'Брокколи', 'Рис']
```

Важно: метод `remove()` удаляет только первый элемент с указанным значением. Все последующие его вхождения остаются в списке. Чтобы удалить все вхождения, нужно использовать цикл `while` в связке с оператором принадлежности `in` и методом `remove` .

Метод pop()

Метод `pop()` удаляет элемент по указанному индексу и возвращает его. В метод `pop()` передается один **необязательный** аргумент:

1. `index` : индекс элемента, который требуется удалить.

Если индекс не указан, то метод удаляет и возвращает последний элемент списка. Если список пуст или указан индекс за пределами диапазона, то во время выполнения происходит ошибка.

Приведённый ниже код:

```
names = ['Gvido', 'Roman', 'Timur']
item = names.pop(1)
print(item)
print(names)
```

выводит:

```
Roman
['Gvido', 'Timur']
```

Метод count()

Метод `count()` возвращает количество элементов в списке, значения которых равны переданному в метод значению.

Таким образом, в метод передается один параметр:

1. `value` : значение, количество элементов, равных которому, нужно посчитать.

Если значение в списке не найдено, то метод возвращает 0.

Приведённый ниже код:

```
names = ['Timur', 'Gvido', 'Roman', 'Timur', 'Anders', 'Timur']
cnt1 = names.count('Timur')
cnt2 = names.count('Gvido')
cnt3 = names.count('Josef')

print(cnt1)
print(cnt2)
print(cnt3)
```

выводит:

```
3
1
0
```

Метод reverse()

Метод `reverse()` инвертирует порядок следования значений в списке, то есть меняет его на противоположный.

Приведённый ниже код:

```
names = ['Gvido', 'Roman', 'Timur']
names.reverse()
print(names)
```

выводит:

```
['Timur', 'Roman', 'Gvido']
```

Существует большая разница между вызовом метода `names.reverse()` и использованием среза `names[::-1]`. Метод `reverse()` меняет порядок элементов на обратный **в текущем списке**, а срез создаёт копию списка, в котором элементы следуют в обратном порядке.

Метод clear()

Метод `clear()` удаляет все элементы из списка.

Приведённый ниже код:

```
names = ['Gvido', 'Roman', 'Timur']
names.clear()
print(names)
```

выводит:

```
[]
```

Метод copy()

Метод `copy()` создаёт поверхностную копию списка.

Приведённый ниже код:

```
names = ['Gvido', 'Roman', 'Timur']
names_copy = names.copy()              # создаем поверхностную копию списка names

print(names)
print(names_copy)
```

выводит:

```
['Gvido', 'Roman', 'Timur']
['Gvido', 'Roman', 'Timur']
```

Аналогичного результата можно достичь с помощью срезов или функции `list()`:

```
names = ['Gvido', 'Roman', 'Timur']
names_copy1 = list(names)              # создаем поверхностную копию с помощью функции list()
names_copy2 = names[:]                 # создаем поверхностную копию с помощью среза от начала до конца
```

Примечания

Примечание. Существует большая разница в работе строковых и списочных методов. Строковые методы не изменяют содержимого объекта, к которому они применяются, а возвращают новое значение. Списочные методы, напротив, меняют содержимое объекта, к которому применяются.

❤️ Happy Pythoning!

Шаг 2

Установите соответствие между списочным методом и тем, что он выполняет.

Чтобы решить это задание откройте https://stepik.org/lesson/324754/step/2

Шаг 3

Что покажет приведённый ниже код?

```
colors = ['Orange']
colors.append('Red')
colors.append('Blue')
colors.append('Green')
colors.insert(0, 'Violet')
colors.insert(2, 'Purple')

print(colors)
```

Чтобы решить это задание откройте https://stepik.org/lesson/324754/step/3

Шаг 4

Что покажет приведённый ниже код?

```
colors = ['Red', 'Blue', 'Green', 'Black', 'White']
del colors[-1]
colors.remove('Green')

print(colors)
```

Чтобы решить это задание откройте https://stepik.org/lesson/324754/step/4

Шаг 5

Всё сразу 2 🌶

Дополните приведённый ниже код, чтобы он:

1. Заменил второй (по порядку) элемент списка на `17`;
2. Добавил числа `4`, `5` и `6` в конец списка;
3. Удалил первый (по порядку) элемент списка;
4. Удвоил список;
5. Вставил число `25` по индексу `3`;
6. Вывел список с помощью функции `print()`

Примечание. Под удвоением списка мы понимаем добавление к исходному списку всех его элементов, сохраняя порядок. Например, при удвоении списка `['py', 'gen']` мы получаем список `['py', 'gen', 'py', 'gen']`.

Чтобы решить это задание откройте https://stepik.org/lesson/324754/step/5

Шаг 6

Переставить min и max

На вход программе подаётся строка текста, содержащая **различные** натуральные числа. Вам необходимо переставить максимальный и минимальный элементы местами и вывести изменённую строку.

Формат входных данных
На вход программе подаётся строка текста, содержащая различные натуральные числа, разделённые символом пробела.

Формат выходных данных
Программа должна вывести текст в соответствии с условием задачи.

Примечание. Используйте подходящие встроенные функции и списочные методы.

▶ Тестовые данные

Sample Input 1:

```
3 4 5 2 1
```

Sample Output 1:

```
3 4 1 2 5
```

Sample Input 2:

```
10 9 8 7 6 5 4 3 2 1
```

Sample Output 2:

```
1 9 8 7 6 5 4 3 2 10
```

Sample Input 3:

```
1 2
```

Sample Output 3:

```
2 1
```

Sample Input 4:

```
1
```

Sample Output 4:

```
1
```

Чтобы решить это задание откройте https://stepik.org/lesson/324754/step/6

Шаг 7

Количество артиклей

На вход программе подаётся строка, содержащая английский текст. Напишите программу, которая подсчитывает общее количество артиклей: `a` , `an` , `the` .

Формат входных данных

На вход программе подаётся строка, содержащая английский текст. Слова текста разделены символом пробела.

Формат выходных данных

Программа должна вывести общее количество артиклей `a`, `an`, `the` вместе с поясняющим текстом.

Примечание. Артикли могут начинаться с заглавной буквы `A`, `An`, `The`.

▶ Тестовые данные

Sample Input:

```
William Shakespeare was born in the town of Stratford, England, in the year 1564. When he was a young man, Shakespeare moved to the city of London, where he began writing plays. His plays were soon very successful, and were enjoyed both by the common people of London and also by the rich and famous. In addition to his plays, Shakespeare wrote many short poems and a few longer poems. Like his plays, these poems are still famous today.
```

Sample Output:

```
Общее количество артиклей: 7
```

Чтобы решить это задание откройте https://stepik.org/lesson/324754/step/7

Шаг 8

Взлом Братства Стали

Немалоизвестный в пустошах Мохаве Курьер забрел в Хидден-Вэли – секретный бункер Братства Стали и любезно соглашается помочь им в решении их проблем. Одной из такой проблем являлся странный компьютерный вирус, который проявлялся в виде появления комментариев к программам на терминалах Братства Стали. Известно, что программисты Братства никогда не оставляют комментарии к коду и пишут программы на Python, поэтому удаление всех этих комментариев никак не навредит им. Помогите писцу Ибсену удалить все комментарии из программы.

Формат входных данных

На первой строке подаётся символ решётки и сразу же натуральное число n – количество строк в программе, не считая первой. Далее следуют n строк кода.

Формат выходных данных

Нужно вывести те же строки, но удалить комментарии и символы пустого пространства в конце строк. Пустую строку вместо первой строки ввода выводить не надо.

Примечание 1. Обращайте внимание на лишние пробелы в конце строк. Проверяющая система не пропустит ваше решение с ними.

Примечание 2. Гарантируется, что в самом коде программы отсутствуют символы `#`.

▶ Тестовые данные

Sample Input:

```
#12
print("Введите своё имя")
name = input()
print("Введите пароль, если имеется")    # ахахахах вам не поймать меня
password = input()
if password == "hoover":
    print("Здравствуйте, рыцарь", name)         #долой Макнамару
elif password == "noncr":
    print("Здравствуйте, паладин", name)
elif password == "gelios":
    print("Здравствуйте, старейшина", name)          #Элайджа вперёд
else:
    print("Здравствуйте, послушник", name)
```

Sample Output:

```
print("Введите своё имя")
name = input()
print("Введите пароль, если имеется")
password = input()
if password == "hoover":
    print("Здравствуйте, рыцарь", name)
elif password == "noncr":
    print("Здравствуйте, паладин", name)
elif password == "gelios":
    print("Здравствуйте, старейшина", name)
else:
    print("Здравствуйте, послушник", name)
```

Чтобы решить это задание откройте https://stepik.org/lesson/324754/step/8

Шаг 9

Метод sort()

В Python списки имеют встроенный метод `sort()`, который сортирует элементы списка по возрастанию или убыванию.

Приведённый ниже код:

```
a = [1, 7, -3, 9, 0, -67, 34, 12, 45, 1000, 6,  8, -2, 99]
a.sort()
print('Отсортированный список:', a)
```

выводит:

```
Отсортированный список: [-67, -3, -2, 0, 1, 6, 7, 8, 9, 12, 34, 45, 99, 1000]
```

По умолчанию метод `sort()` сортирует список по возрастанию. Если требуется отсортировать список по убыванию, необходимо явно указать параметр `reverse = True`.

Приведённый ниже код:

```
a = [1, 7, -3, 9, 0, -67, 34, 12, 45, 1000, 6,  8, -2, 99]
a.sort(reverse=True)  # сортируем по убыванию
print('Отсортированный список:', a)
```

выводит:

```
Отсортированный список: [1000, 99, 45, 34, 12, 9, 8, 7, 6, 1, 0, -2, -3, -67]
```

Примечания

Примечание 1. С помощью метода `sort()` можно сортировать списки содержащие не только числа, но и строки. В таком случае элементы списка сортируются в соответствии с лексикографическим порядком (https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BA%D1%81%D0%B8%D0%BA%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA).

Приведённый ниже код:

```
a = ['бета', 'альфа', 'дельта', 'гамма']
a.sort()
print('Отсортированный список:', a)
```

выводит:

```
Отсортированный список: ['альфа', 'бета', 'гамма', 'дельта']
```

Примечание 2. Метод `sort()` использует алгоритм Timsort (https://ru.wikipedia.org/wiki/Timsort).

❤️ Happy Pythoning!

Шаг 10

Что покажет приведённый ниже код?

```
numbers = [4, 2, 8, 6, 5, 3, 10, 4, 100, 1, -7]
numbers.sort()
del numbers[0]
del numbers[-1]
numbers.sort(reverse=True)

print(numbers)
```

Чтобы решить это задание откройте https://stepik.org/lesson/324754/step/10

Шаг 11

Сортировка чисел

На вход программе подаётся 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.

Формат входных данных

На вход программе подаётся строка текста, содержащая целые числа, разделённые символом пробела.

Формат выходных данных

Программа должна вывести две строки текста в соответствии с условием задачи.

▶ Тестовые данные

Sample Input:

```
4 5 1 2 3 8
```

Sample Output:

```
1 2 3 4 5 8
8 5 4 3 2 1
```

Чтобы решить это задание откройте https://stepik.org/lesson/324754/step/11

11.7 Списочные выражения

Шаг 1

Тема урока: списочные выражения

1. Списочные выражения
2. Решение задач

Аннотация. Списочные выражения. Создание списков без явного использования циклов и вызова списочного метода `append()`.

Создание списков

Для того чтобы создать список, состоящий из 10 нулей, мы можем использовать приведённый ниже код:

```
zeros = []
for i in range(10):
    zeros.append(0)
```

В Python, однако, есть более простой и компактный способ для создания такого списка. Мы можем использовать оператор умножения списка на число:

```
zeros = [0] * 10
```

Для создания списков, заполненных по более сложным правилам, нам приходится явно использовать цикл `for`.

Например, для создания списка целых чисел от 0 до 9 мы вынуждены писать такой код:

```
numbers = []
for i in range(10):
    numbers.append(i)
```

Приведённый выше код хоть и не является сложным, однако достаточно громоздок.

Списочные выражения

В Python есть механизм для создания списков из неповторяющихся элементов. Такой механизм называется **списочное выражение** (list comprehension).

Предыдущий код можно записать следующим образом:

```
numbers = [i for i in range(10)]
```

Общий вид списочного выражения следующий:

`[выражение for переменная in последовательность]`

где `переменная` – имя некоторой переменной, `последовательность` – последовательность значений, которые она принимает (список, строка или объект, полученный при помощи функции `range`), `выражение` – некоторое выражение, как правило, зависящее от использованной в списочном выражении переменной, которым будут заполнены элементы списка.

Примеры использования списочных выражений

1. Создать список, заполненный 10 нулями можно и при помощи списочного выражения:

```
zeros = [0 for i in range(10)]
```

2. Создать список, заполненный квадратами целых чисел от 0 до 9 можно так:

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

3. Создать список, заполненный кубами целых чисел от 10 до 20 можно так:

```
cubes = [i ** 3 for i in range(10, 21)]
```

4. Создать список, заполненный символами строки:

```
chars = [c for c in 'abcdefg']
print(chars)
```

Считывание входных данных

При решении многих задач из предыдущих уроков мы считывали начальные данные (строки, числа) и заполняли ими список. С помощью списочных выражений процесс заполнения списка можно заметно сократить.

Например, если сначала вводится число `n` – количество строк, а затем на вход поступают сами строки, то создать список можно так:

```
n = int(input())
lines = [input() for _ in range(n)]
```

Можно опустить описание переменной `n`:

```
lines = [input() for _ in range(int(input()))]
```

Если требуется считать список чисел, то необходимо добавить преобразование типов:

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

Обратите внимание, мы используем символ `_` в качестве имени переменной цикла, поскольку она не используется.

Списочные выражения часто используются для инициализации списков. В Python не принято создавать пустые списки, а затем заполнять их значениями, если можно этого избежать.

Условия в списочном выражении

В списочных выражениях можно использовать условный оператор. Например, если требуется создать список чётных чисел от 0 до 20, то мы можем написать такой код:

```
evens = [i for i in range(21) if i % 2 == 0]
```

Важно: для того, чтобы получить список, состоящий из четных чисел, лучше использовать функцию `range(0, 21, 2)`. Предыдущий пример приведен для демонстрации возможности использования условий в списочных выражениях.

Вложенные циклы

В списочном выражении можно использовать вложенные циклы.

Приведённый ниже код:

```
numbers = [i * j for i in range(1, 5) for j in range(2)]
print(numbers)
```

выводит:

```
[0, 1, 0, 2, 0, 3, 0, 4]
```

Приведённый выше код код равнозначен следующему:

```
numbers = []

for i in range(1, 5):
    for j in range(2):
        numbers.append(i * j)
print(numbers)
```

Подводя итог

Пусть `word = 'Hello', numbers = [1, 14, 5, 9, 12], words = ['one', 'two', 'three', 'four', 'five', 'six']`.

Списочное выражение	Результирующий список
`[0 for i in range(10)]`	`[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`
`[i ** 2 for i in range(1, 8)]`	`[1, 4, 9, 16, 25, 36, 49]`
`[i * 10 for i in numbers]`	`[10, 140, 50, 90, 120]`
`[c * 2 for c in word]`	`['HH', 'ee', 'll', 'll', 'oo']`
`[m[0] for m in words]`	`['o', 't', 't', 'f', 'f', 's']`
`[i for i in numbers if i < 10]`	`[1, 5, 9]`
`[m[0] for m in words if len(m) == 3]`	`['o', 't', 's']`

❤️ Happy Pythoning!

Шаг 2

Как записывается списочное выражение?

Чтобы решить это задание откройте https://stepik.org/lesson/326725/step/2

Шаг 3

Используя списочное выражение, дополните приведённый ниже код так, чтобы получить новый список, содержащий строки исходного списка, где у каждой строки удалён первый символ.

Чтобы решить это задание откройте https://stepik.org/lesson/326725/step/3

Шаг 4

Используя списочное выражение, дополните приведённый ниже код так, чтобы получить новый список, содержащий длины строк исходного списка.

Чтобы решить это задание откройте https://stepik.org/lesson/326725/step/4

Шаг 5

Используя списочное выражение, дополните приведённый ниже код так, чтобы получить новый список, содержащий только слова длиной не менее пяти символов (включительно).

Чтобы решить это задание откройте https://stepik.org/lesson/326725/step/5

Шаг 6

Используя списочное выражение, дополните приведённый ниже код так, чтобы получить список всех чисел-палиндромов от 100 до 1000 (включительно).

Примечание. Результирующий список должен состоять из целых чисел.

Чтобы решить это задание откройте https://stepik.org/lesson/326725/step/6

Шаг 7

Списочное выражение 1

На вход программе подаётся натуральное число n. Напишите программу, использующую **списочное выражение**, которая создаёт список, содержащий квадраты чисел от 1 до n (включительно), а затем выводит его элементы построчно, то есть каждый на отдельной строке.

Формат входных данных
На вход программе подаётся натуральное число.

Формат выходных данных
Программа должна вывести текст в соответствии с условием задачи.

Примечание. Для вывода элементов списка используйте цикл `for`.

▶ Тестовые данные

Sample Input:

```
5
```

Sample Output:

```
1
4
9
16
25
```

Чтобы решить это задание откройте https://stepik.org/lesson/326725/step/7

Шаг 8

Списочное выражение 2

На вход программе подаётся строка текста, содержащая целые числа. Напишите программу, **использующую списочное выражение**, которая выведет кубы указанных чисел на одной строке.

Формат входных данных
На вход программе подаётся строка текста, содержащая целые числа, разделённые символом пробела.

Формат выходных данных
Программа должна вывести текст в соответствии с условием задачи.

Примечание 1. Для вывода элементов списка используйте цикл `for`.

Примечание 2. Используйте метод `split()`.

▶ Тестовые данные

Sample Input 1:

```
2 4 3
```

Sample Output 1:

```
8 64 27
```

Sample Input 2:

```
-2 -5 0
```

Sample Output 2:

```
-8 -125 0
```

Чтобы решить это задание откройте https://stepik.org/lesson/326725/step/8

Шаг 9

В одну строку 1

На вход программе подаётся строка текста, содержащая слова. Напишите программу, которая выводит слова введённой строки в столбик.

Формат входных данных
На вход программе подаётся строка текста, содержащая слова, разделённые символом пробела.

Формат выходных данных
Программа должна вывести текст в соответствии с условием задачи.

Примечание. Программу можно написать в одну строку кода.

▶ Тестовые данные

Sample Input:

```
Умей ценить того кто без тебя не может
```

Sample Output:

```
Умей
ценить
того
кто
без
тебя
не
может
```

Чтобы решить это задание откройте https://stepik.org/lesson/326725/step/9

Шаг 10

В одну строку 2

На вход программе подаётся строка текста. Напишите программу, **использующую списочное выражение**, которая выводит все цифровые символы данной строки.

Формат входных данных
На вход программе подаётся строка текста.

Формат выходных данных
Программа должна вывести текст в соответствии с условием задачи.

Примечание. Программу можно написать в одну строку кода.

▶ Тестовые данные

Sample Input 1:

```
Число Pi примерно равно 3.1415
```

Sample Output 1:

```
31415
```

Sample Input 2:

```
123Python awesome!56
```

Sample Output 2:

```
12356
```

Чтобы решить это задание откройте https://stepik.org/lesson/326725/step/10

В одну строку 3

На вход программе подаётся строка текста, содержащая целые числа. Напишите программу, использующую списочное выражение, которая выведет квадраты чётных чисел, кроме тех квадратов, которые оканчиваются на цифру 4.

Формат входных данных

На вход программе подаётся строка текста, содержащая целые числа, разделённые символом пробела.

Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

Примечание. Программу можно написать в одну строку кода.

▶ Тестовые данные

Sample Input 1:

```
1 2 3 4 5 6 7 8 9
```

Sample Output 1:

```
16 36
```

Sample Input 2:

```
4 4 10 6 4
```

Sample Output 2:

```
16 16 100 36 16
```

11.8 Сортировка списков

Шаг 1

Тема урока: сортировка списков

1. Задача сортировки
2. Сортировка пузырьком
3. Сортировка выбором
4. Сортировка простыми вставками

Аннотация. Задачи и способы (алгоритмы) сортировки списков.

Задача сортировки

Задача сортировки списка заключается в перестановке его элементов так, чтобы они были упорядочены по возрастанию или убыванию. Это одна из основных задач программирования. Мы сталкиваемся с ней очень часто: при записи фамилий учеников в классном журнале, при подведении итогов соревнований и т.д.

Алгоритмы сортировки

Алгоритм сортировки — это алгоритм упорядочивания элементов в списке. Алгоритмы сортировки оцениваются по скорости выполнения и эффективности использования памяти:

- время — основной параметр, характеризующий быстродействие алгоритма;
- память — ряд алгоритмов требует выделения дополнительной памяти под временное хранение данных.

Алгоритмы сортировки, не потребляющие дополнительной памяти, относят к **сортировкам на месте**.

Основные алгоритмы сортировки

Медленные:

1. Пузырьковая сортировка (Bubble sort);
2. Сортировка выбором (Selection sort);
3. Сортировка простыми вставками (Insertion sort).

Быстрые:

1. Сортировка Шелла (Shell sort);
2. Быстрая сортировка (Quick sort);
3. Сортировка слиянием (Merge sort);
4. Пирамидальная сортировка (Heap sort);
5. Сортировка TimSort (используется в Java и Python).

Большинство алгоритмов сортировки, в частности, указанные выше, основаны на сравнении двух элементов списка. Существуют однако алгоритмы не основанные на сравнениях. Такие алгоритмы, как правило, используют наперед заданные условия относительно элементов списка. Например, элементами списка являются натуральные или целые числа в некотором диапазоне, элементами являются строки и т.д.

К алгоритмам, не основанным на сравнениях, можно отнести следующие:

1. Сортировка подсчетом (Counting sort)

2. Блочная сортировка (Bucket sort)
3. Поразрядная сортировка (Radix sort)

В рамках курса мы рассмотрим несложные алгоритмы пузырьковой сортировки, сортировки выбором и сортировки простыми вставками.

Примечания

Примечание 1. Подробнее об алгоритмах сортировки можно почитать по ссылке (https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B8).

Примечание 2. Мы называем некоторые алгоритмы сортировки **медленными**, поскольку они тратят много времени на сортировку больших списков. Например, если список содержит порядка миллиона элементов, то такие алгоритмы тратят часы, а то и дни на выполнение сортировки, в то время как быстрые алгоритмы справляются с задачей за секунды.

Примечание 3. Наглядную работу алгоритмов сортировки на разных входных данных можно посмотреть по ссылке (https://www.toptal.com/developers/sorting-algorithms).

Happy Pythoning!

Шаг 2

Сортировка пузырьком

Алгоритм сортировки пузырьком состоит из повторяющихся проходов по сортируемому списку. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по списку повторяются $n - 1$ раз, где n – длина списка. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент списка ставится на свое место в конце списка рядом с предыдущим «наибольшим элементом».

Наибольший элемент каждый раз «всплывает» до нужной позиции, как пузырёк в воде — отсюда и название алгоритма.

Алгоритм пузырьковой сортировки считается учебным и практически не применяется вне учебной литературы, а на практике применяются более эффективные.

Рассмотрим работу алгоритма на примере сортировки списка `a = [5, 1, 4, 2, 8]` по возрастанию.

Первый проход:

1. `[5, 1, 4, 2, 8]` → `[1, 5, 4, 2, 8]`: меняем местами первый и второй элементы, так как $5 > 1$;
2. `[1, 5, 4, 2, 8]` → `[1, 4, 5, 2, 8]`: меняем местами второй и третий элементы, так как $5 > 4$;
3. `[1, 4, 5, 2, 8]` → `[1, 4, 2, 5, 8]`: меняем местами третий и четвертый элементы, так как $5 > 2$;
4. `[1, 4, 2, 5, 8]` → `[1, 4, 2, 5, 8]`: не меняем четвертый и пятый элементы местами, так как $5 < 8$;
5. Самый большой элемент встал («всплыл») на свое место.

Второй проход:

1. `[1, 4, 2, 5, 8]` → `[1, 4, 2, 5, 8]`: не меняем первый и второй элементы местами, так как $1 < 4$;
2. `[1, 4, 2, 5, 8]` → `[1, 2, 4, 5, 8]`: меняем местами второй и третий элементы, так как $4 > 2$;
3. `[1, 2, 4, 5, 8]` → `[1, 2, 4, 5, 8]`: не меняем местами третий и четвертый элементы, так как $4 < 5$;
4. Второй по величине элемент встал («всплыл») на свое место.

Теперь список полностью отсортирован, но алгоритму это неизвестно и он работает дальше.

Третий проход:

1. `[1, 2, 4, 5, 8]` → `[1, 2, 4, 5, 8]`: не меняем первый и второй элементы местами, так как $1 < 2$;

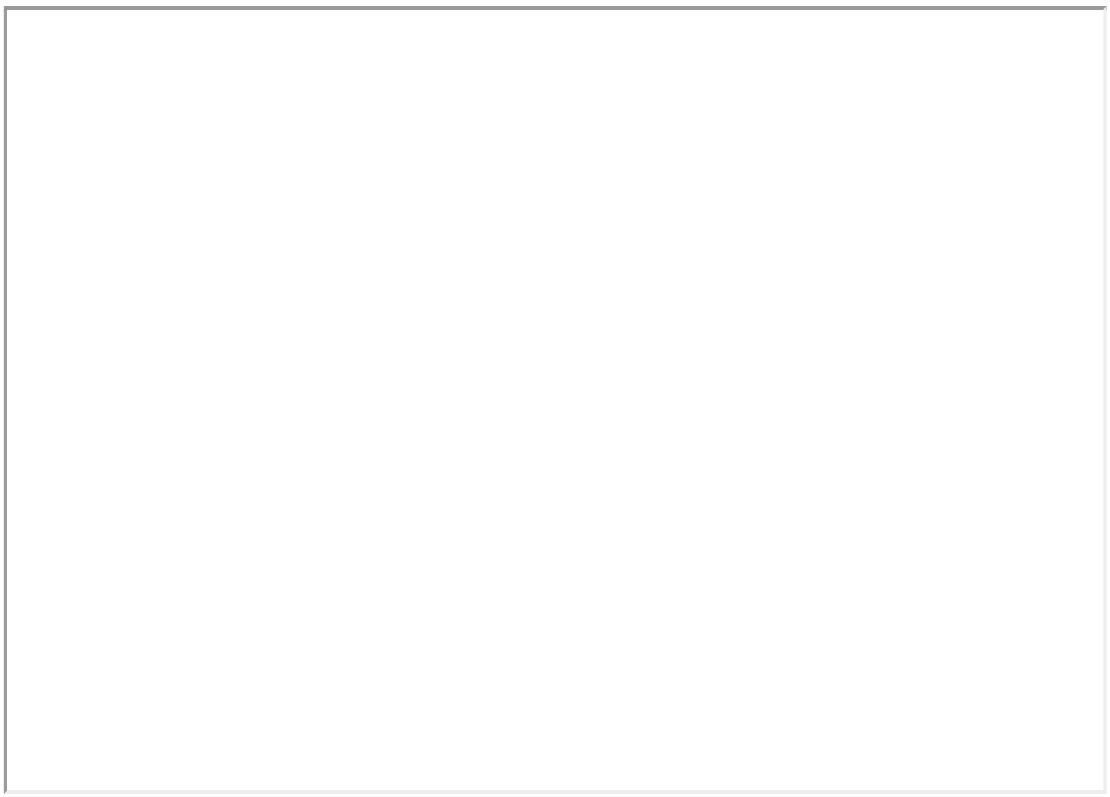
2. `[1, 2, 4, 5, 8]` → `[1, 2, 4, 5, 8]`: не меняем второй и третий элементы местами, так как $2 < 4$;
3. Третий по величине элемент встал («всплыл») на свое место. (на котором и был)

Четвертый проход:

1. `[1, 2, 4, 5, 8]` → `[1, 2, 4, 5, 8]`:
2. Четвертый по величине элемент встал («всплыл») на свое место.

Теперь список отсортирован и алгоритм может быть завершен.

Визуализация алгоритма



Реализация алгоритма

Пусть требуется отсортировать по возрастанию список чисел: `a = [1, 7, -3, 9, 0, -67, 34, 12, 45, 1000, 6,  8, -2, 99]`.

Следующий программный код реализует алгоритм пузырьковой сортировки:

```
a = [1, 7, -3, 9, 0, -67, 34, 12, 45, 1000, 6,  8, -2, 99]
n = len(a)

for i in range(n - 1):
    for j in range(n - 1 - i):
        if a[j] > a[j + 1]:                  # если порядок элементов пары неправильный
            a[j], a[j + 1] = a[j + 1], a[j]  # меняем элементы пары местами

print('Отсортированный список:', a)
```

Результатом выполнения такого кода будет:

```
Отсортированный список: [-67, -3, -2, 0, 1, 6, 7, 8, 9, 12, 34, 45, 99, 1000]
```

Оптимизация алгоритма

Алгоритм пузырьковой сортировки можно немного ускорить. Если на одном из очередных проходов окажется, что обмены больше не нужны, то это означает, что все элементы списка находятся на своих местах, то есть список отсортирован. Для реализации такого ускорения нужно воспользоваться сигнальной меткой, то есть флажком и оператором прерывания `break`.

❤️ Happy Pythoning!

Шаг 3

Чтобы просмотреть это видео откройте https://stepik.org/lesson/310445/step/3

Шаг 4

Оптимизируйте приведённый ниже код, реализующий алгоритм пузырьковой сортировки.

Чтобы решить это задание откройте https://stepik.org/lesson/310445/step/4

Сортировка выбором

Сортировка выбором улучшает пузырьковую сортировку, совершая всего **один обмен за каждый проход по списку**. Для этого алгоритм ищет максимальный элемент и помещает его на соответствующую позицию. Как и для пузырьковой сортировки, после первого прохода самый большой элемент находится на правильном месте. После второго прохода на своё место становится следующий максимальный элемент. Проходы по списку повторяются $n - 1$ раз, где n – длина списка, поскольку последний из них автоматически оказывается на своем месте.

Алгоритм сортировки выбором также считается учебным и практически не применяется вне учебной литературы. На практике используют более эффективные алгоритмы.

Рассмотрим работу алгоритма на примере сортировки списка `a = [5, 1, 8, 2, 4]` по возрастанию.

Первый проход:

Находим максимальный элемент `8` в неотсортированной части списка и меняем его с последним элементом списка:

`[5, 1, 4, 2, 8]`.

Второй проход:

Находим максимальный элемент `5` в неотсортированной части списка и меняем его с предпоследним элементом списка:

`[2, 1, 4, 5, 8]`.

Третий проход:

Находим максимальный элемент `4` в неотсортированной части списка и меняем его с пред-предпоследним элементом списка:

`[2, 1, 4, 5, 8]`.

Четвертый проход:

Находим максимальный элемент `2` в неотсортированной части списка и меняем его с вторым элементом списка:

`[1, 2, 4, 5, 8]`.

Теперь список отсортирован и алгоритм может быть завершен.

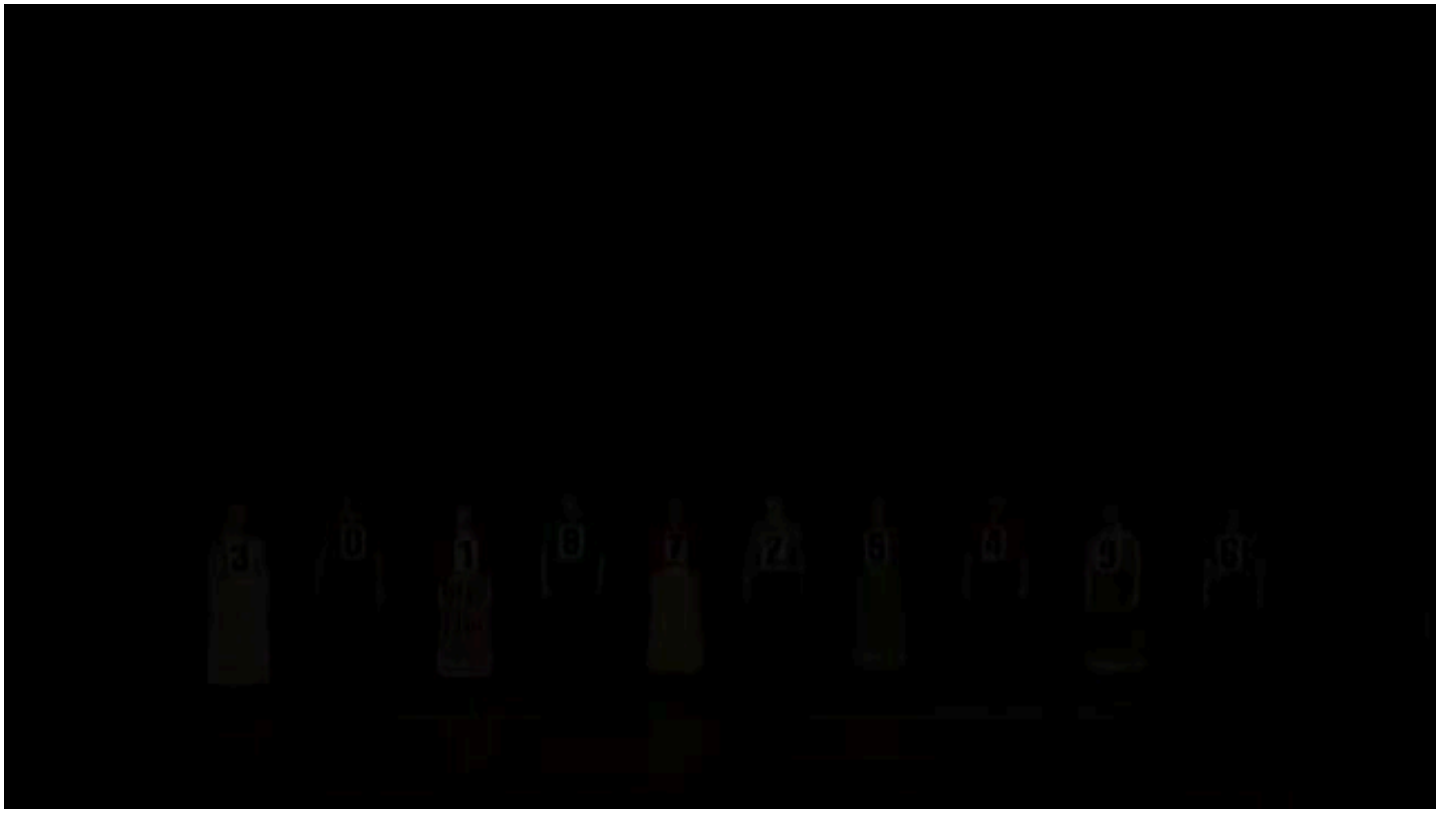
Вместо максимального элемента можно искать минимальный.

Визуализация алгоритма

Выберите в выпадающем списке алгоритм: **Selection sort**.

❤️ **Happy Pythoning!**

Шаг 6



Чтобы просмотреть это видео откройте

https://stepik.org/lesson/310445/step/6

Шаг 7

Отсортируйте список по возрастанию, реализовав алгоритм сортировки выбором.

Чтобы решить это задание откройте https://stepik.org/lesson/310445/step/7

Шаг 8

Сортировка простыми вставками

Алгоритм сортировки простыми вставками делит список на 2 части — отсортированную и неотсортированную. Из неотсортированной части извлекается очередной элемент и вставляется на нужную позицию в отсортированной части, в результате чего отсортированная часть списка увеличивается, а неотсортированная уменьшается. Так происходит, пока не исчерпан набор входных данных и не отсортированы все элементы.

Сортировка простыми вставками наиболее эффективна, когда список уже частично отсортирован и элементов массива немного. Если элементов в списке меньше 10, то этот алгоритм — один из самых быстрых.

Рассмотрим его работу на примере сортировки списка `a = [5, 1, 8, 2, 4]` по возрастанию.

Первый проход:

Делим список на две части: отсортированную `[5]` и неотсортированную `[1, 8, 2, 4]`.

Извлекаем первый элемент `1` из неотсортированной части списка и находим ему место в отсортированной части:

`[1, 5, 8, 2, 4]`.

Второй проход:

Делим список на две части: отсортированную `[1, 5]` и неотсортированную `[8, 2, 4]`.

Извлекаем первый элемент `8` из неотсортированной части списка и находим ему место в отсортированной части:

`[1, 5, 8, 2, 4]`.

Третий проход:

Делим список на две части: отсортированную `[1, 5, 8]` и неотсортированную `[2, 4]`.

Извлекаем первый элемент `2` из неотсортированной части списка и находим ему место в отсортированной части:

`[1, 2, 5, 8, 4]`.

Четвертый проход:

Делим список на две части: отсортированную `[1, 2, 5, 8]` и неотсортированную `[4]`.

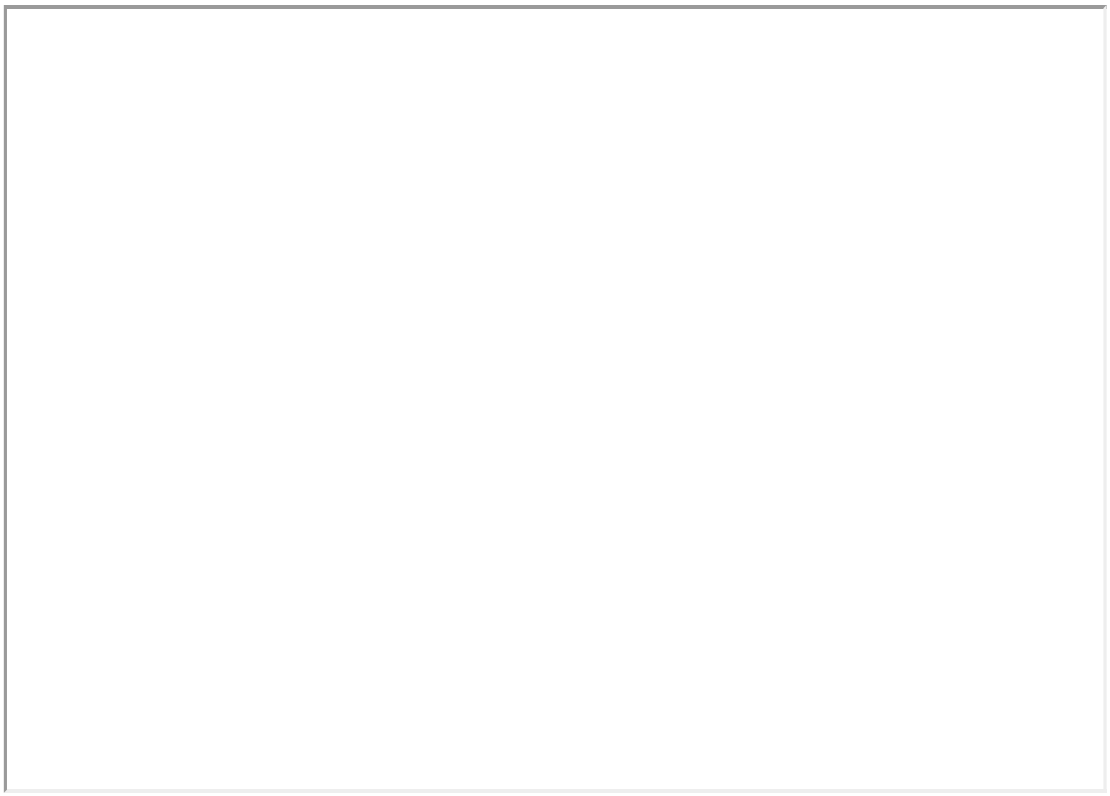
Извлекаем первый элемент `4` из неотсортированной части списка и находим ему место в отсортированной части:

`[1, 2, 4, 5, 8]`.

Теперь список отсортирован, и алгоритм может быть завершен.

Визуализация алгоритма

Выберите в выпадающем списке алгоритм: **Insert sort**.



Реализация алгоритма

Пусть требуется отсортировать по возрастанию список чисел: `a = [1, 7, -3, 9, 0, -67, 34, 12, 45, 1000, 6,  8, -2, 99]`.

Следующий программный код реализует алгоритм сортировки простыми вставками:

```
a = [1, 7, -3, 9, 0, -67, 34, 12, 45, 1000, 6,  8, -2, 99]
n = len(a)

for i in range(1, n):
    elem = a[i]  # берем первый элемент из неотсортированной части списка
    j = i

    # пока элемент слева существует и больше нашего текущего элемента
    while j >= 1 and a[j - 1] > elem:
        # смещаем j-й элемент отсортированной части вправо
        a[j] = a[j - 1]
        # сами идём влево, дальше ищем место для нашего текущего элемента
        j -= 1

    # нашли место для нащего текущего элемента из неотсортированной части
    # и вставляем его на индекс j в отсортированной части
    a[j] = elem


print('Отсортированный список:', a)
```

Результатом выполнения такого кода будет:

```
Отсортированный список: [-67, -3, -2, 0, 1, 6, 7, 8, 9, 12, 34, 45, 99, 1000]
```

Оптимизация алгоритма

Алгоритм сортировки простыми вставками можно значительно ускорить, если осуществлять поиск нужной позиции для вставки очередного элемента из неотсортированной части списка с помощью бинарного поиска (https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA).

Шаг 9

Чтобы просмотреть это видео откройте

https://stepik.org/lesson/310445/step/9