**Шаг 1. Основные операции со списками**

**СПИСКИ**

Для хранения нескольких значений используются списки (они же листы).

Для того чтобы создать список, используется синтаксис вида:

myList =['один', 'два', 'три']

 В качестве элементов листа могут выступать любые типы данных:

list\_of\_some\_values =[ **123**, 'питон', True ]

С листами очень удобно работать при необходимости фильтрации, преобразования и объединения данных.

Посмотрим основные операции для работы с листами на примере списка значений:

just\_numbers =[**2** ,**1**, **5**, **4**, **3**]

Основные операции с листами

Основные простые операции с листами:

Количество элементов в листе

len( just\_numbers )

Что получится в результате?



Сумма элементов листа

sum( just\_numbers )

Что получится в результате?



**Шаг 2. Сортировка списков**

Сортировка по возрастанию

Cортировка элементов листа по возрастанию

sorted( just\_numbers )

Что получится в результате?

СОРТИРОВКА ПО УБЫВАНИЮ

Если элементы листа представляют собой слова, то сортировка будет происходить по алфавиту.

Для сортировки по убыванию добавим параметр reverse = True:

sorted( just\_numbers, reverse = True )

[5, 4, 3, 2, 1]

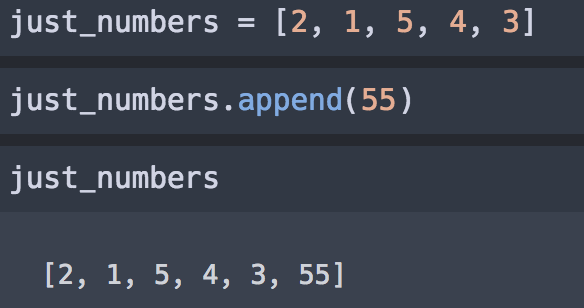
**Шаг 3. Добавление элемента к списку**

**ДОБАВЛЕНИЕ ЭЛЕМЕНТА К СПИСКУ**

Для добавления элемента к списку используется метод append:

just\_numbers = [**2**, **1**, **5**, **4**, **3**]

just\_numbers.append(**55**)



Упражнение

Создайте лист из трех слов: 'один', 'два', 'три'.

Выведите на экран этот лист, отсортированный по алфавиту (используйте следующий формат: ['...', '...', '...'], где вместо многоточия должны стоять элементы листа).



**Шаг 4. Списки и строки**

**КАК ПОЛУЧИТЬ ЛИСТ ИЗ СТРОКИ**

В большинстве случаев данные будут поступать вам в виде листов, а не строк. Например, при чтении таблицы с данными из файла. Если таблица имеет разделитель (например, как запятая в нашей задаче), то для перевода строки в лист можно использовать функцию split. В скобках указываем разделитель элементов (в нашем примере это запятая):

queries\_string ="смотреть сериалы онлайн,новости спорта,афиша кино,курс доллара,сериалы этим летом,курс по питону,сериалы про спорт"

Применяем split

Наберите в новой ячейке

**print**( queries\_string.split(',') )

Что получится в результате?



**Шаг 5. Проверочное задание**

Вам дана следующая строка с названиями файлов, которые идут не по порядку:

"003\_logs\_2017-11-03;001\_logs\_2017-11-01;005\_logs\_2017-11-05;002\_logs\_2017-11-02;004\_logs\_2017-11-04"

Запишите эту строку в переменную file\_string.

Вам необходимо:

1. Преобразовать эту строку в лист (назовите лист file\_list). В качестве разделителя используйте точку с запятой.

С помощью какой функции это можно сделать?

append

strip

split

2. Отсортировать лист по возрастанию дат (т. е. чтобы первым был файл 001\_logs\_2017-11-01, а последним — 005\_logs\_2017-11-05). Результат запишите в переменную file\_list\_sorted.

Какой код позволит сделать такое преобразование?

file\_list\_sorted = sorted(file\_list, reverse = True)

file\_list\_sorted = sorted(file\_list)

';'.join(file\_list)

3. Добавить в отсортированный список следующий элемент: 006\_logs\_2017-11-06.

Какой результат вы должны получить в итоге?io

['001\_logs\_2017-11-01', '002\_logs\_2017-11-02', '003\_logs\_2017-11-03', '004\_logs\_2017-11-04', '005\_logs\_2017-11-05', '006\_logs\_2017-11-06']

['006\_logs\_2017-11-06', '005\_logs\_2017-11-05', '004\_logs\_2017-11-04', '003\_logs\_2017-11-03', '002\_logs\_2017-11-02', '001\_logs\_2017-11-01']

['003\_logs\_2017-11-03', '001\_logs\_2017-11-01', '005\_logs\_2017-11-05', '002\_logs\_2017-11-02', '004\_logs\_2017-11-04', '006\_logs\_2017-11-06']

**Экспорт результатов вычислений**

**Шаг 1. Переводим результаты вычислений в строку**

**ПЕРЕВОД СПИСКА В СТРОКУ**

Для перевода списка в строку используется функция join. Желаемый разделитель указываем в кавычках. Например, дан лист с поисковымsoи запросами:

queriesList = [ 'смотреть сериалы онлайн', 'новости спорта', 'афиша кино', 'курс доллара', 'сериалы этим летом', 'курс по питону', 'сериалы про спорт' ]

Вы можете перевести лист обратно в строку с запятой в качестве разделителя.

**print**( ','.join( queriesList ) )

Перевод в строку

Какой текст выводится на экран в результате выполнения этой команды?



**Шаг 2. Используем другой разделитель**

**РАЗДЕЛИТЕЛЬ ТАБУЛЯЦИЯ**

Этот способ очень удобен при записи результатов в файл, особенно, если количество записываемых данных заранее неизвестно. Например, после фильтрации количество оставшихся запросов может быть любым. Если вам необходимо в качестве разделителя использовать знак табуляции (это удобно при дальнейшем экспорте в Excel), то используем служебный знак \t:

**print**( '**\t**'.join( queriesList ) )

Упражнение

В прошлом упражнении вы получили лист

results = ['001\_logs\_2017-11-01', '002\_logs\_2017-11-02', '003\_logs\_2017-11-03', '004\_logs\_2017-11-04', '005\_logs\_2017-11-05', '006\_logs\_2017-11-06']

Преобразуйте этот лист в строку, используя в качестве разделителя две вертикальные черты ||. Внесите ниже получившуюся строку.



**Шаг 3. Выбор элементов листов**

**ВЫБОР ЭЛЕМЕНТОВ СПИСКА**

В работе с данными очень часто возникает необходимость выбирать элементы списка на основе их позиции. Например, второй справа. Или выбрать только четные, начиная с третьего элемента. Для таких операций со списками в питоне есть простые и удобные способы фильтрации.

Возьмем другой пример листа:

sequence = [ 'Google Adwords', 'Yandex Direct', 'Facebook', 'VK', 'Partner' ]

Для получения нужного элемента укажите его номер:

sequence[**1**]

Проверим

Что выдаст система?

**Шаг 4. Индекс**

Обратите внимание, что вместо первого по счету элемента 'Google Adwords' мы почему-то получили элемент 'Yandex Direct'. В этом примере видна очень важная особенность нумерации элементов во всех основных языках программирования: номера всех элементов начинаются с нулевого. Т. е. элемент 'Google Adwords' имеет порядковый номер 0:

sequence[**0**]

Результат:

'Google Adwords'

Упражнение

Если мы хотим получить элементы с первого по третий, то пишем так:

sequence[**0**:**3**]

Что получится?

**Шаг 5. Границы указывать не обязательно**

Из-за того, что вся нумерация смещается на 1 влево, Python возьмет все элементы от нулевого до третьего минус 1, что мы и видим в примере. Такой алгоритм может показаться странным, однако в дальнейшем мы увидим, что он очень удобен при работе с различными интервалами.

В прошлом примере можно не указывать 0, т. е. просто написать

sequence[:**3**]

Результат:

['Google Adwords', 'Yandex Direct', 'Facebook']

Упражнение

Как можно получить следующий результат (возможны несколько правильных вариантов)?

['Google Adwords', 'Yandex Direct', 'Facebook']

sequence[0:2]

sequence[0:3]

sequence[:2]

sequence[:3]

**Шаг 6. Сложная фильтрация элементов**

Фильтрация элементов

Выведем все элементы от третьего по счету до последнего в листе

sequence = [ 'Google Adwords', 'Yandex Direct', 'Facebook', 'VK', 'Partner' ]

sequence[**2**:]

Какой результат вы получите?

**НЕГАТИВНЫЙ ИНДЕКС**

Как вы уже видели в прошлом шаге в листах можно использовать отрицательные индексы. Т. е. номер элемента будет считаться с конца. Например, чтобы вывести последний элемент в листе, введите sequence[-1]:

sequence[-**1**]

'Partner'

Упражнение

С помощью какой команды можно вывести все элементы с третьего до предпоследнего? Т. е. получить ['Facebook', 'VK']. Возможно несколько верных ответов.

sequence[2:-1]

sequence[-3:-1]

sequence[-3:4]

sequence[2:4]

**Шаг 7. Фильтрация с шагом**

Также можно задавать интервал, с которым мы проходимся по листу. Например, для получения только четных чисел из последовательности от 1 до 10 используйте следующий синтаксис:

justNumbers = [**1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **10**]

justNumbers[**1**::**2**]

Упражнение

Что выдаст система после такой команды?