# Лекция 3

Списки и массивы

Начну я свой рассказ по это теме со списков. Если коротко, то список это переменная в которой хранится не одно значение, а несколько. Сформировать его можно разными способами: объявление пустого списка и дальнейшее дополнение его различными значениями, сразу создать его с необходимыми значениями, преобразование строковых переменных в списки, путем заключения строковой переменной в функцию list(\_\_) и итоговое ее присвоение к переменной и еще нескольких

Каждый из элементов в списке имеет свой порядковый номер, по которому можно к нему обратиться. Обращение к списку происходит следующим образом: дан список **b=[1,2,3,56]** и мы хотим вывести на экран его второй элемент, для этого мы сделаем следующее **print(b[1])**  **2**. Вы, наверное, спросите почему один, но в самой первой лекции, когда я рассказывал про строки, я же говорил, что нумерация (индексация) в python начинается не с 1, как нумеровали бы математики, а с 0, как всегда, нумеруют программисты, поэтому чтобы достать **n-ый** элемент мы запросим элемент с индексом **n-1**.

Помимо обращения по одиночному индексу к списку можно обращаться, как и к строке, по так называемому *срезу*, то есть достать несколько значений, от и до определенного индекса. Покажем на примере: возьмём ранее объявленный список b и вытащи из него первые три значения **print(b[0:3])** или же можно просто написать **print(b[:3])** ответ будет одинаковый **[1,2,3]**. Индексация среза так же ведется с нуля, элемент с последним индексом не включается в срез.

Но всё это так или иначе уже было у нас в теме про строки, давайте перейдём к чему-то новому и еще не встречавшемуся ранее.

Для всех наших манипуляций мы возьмём такой список:

**Sp = [1,56,35,95,364,1]** и нам еще нужен будет другой список, для склейки, возьмём его из прошлого примера: **b = [1,2,3,56]**

Представляю вашему вниманию список *основных функций* необходимый для работы со списками.

1. **print(len(**Sp**))** – выводит целое число, длину списка
2. **Sp.append(**элемент**)** – вставка элемента в конец списка
3. **Sp.extend(**b**)** – конкатенация («склейка») двух списков, дополнительный список дописывается в конец исходного, дальнейшее обращение к уже «склеенному» списку происходит как и к исходному.
4. **Sp.insert(**индекс, элемент**)** – вставка в место с выбранным индексом, выбранного элемента
5. **Sp.remove(**элемент**)** – удаление конкретного элемента, не важно знаем мы его индекс или нет. Если таких элементов несколько, то удалится только первый из них (с наименьшим индексом). Если таких элементов нет, то выдастся ошибка ***ValueError: «элемент» is not in list***
6. **Sp.pop(**индекс**)** – удаляет элемент по индексу. Если индекс не указывать и оставить просто пустые скобки, то будет удалён последний элемент в списке.
7. **print(Sp.index(**элемент**))** – выводит индекс элемента
8. **print(Sp.count(**элемент**))** – выводит количество таких элементов
9. **Sp.sort()** – сортирует ***по возрастанию*** все элементы в списке. Если будет указать в скобках reverse = True, то сортировка будет производиться ***по убыванию***.
10. **Sp.reverse()** – делает список задом на перед
11. **Sp.clear()** – очищает список (мало ли вам это понадобится)

Теперь мы вернемся к нашим спискам и посмотри все на примерах:

|  |  |
| --- | --- |
| Sp.append(45)  print(Sp) | [1, 56, 35, 95, 364, 1, 45] |
| Sp.extend(b)  print(Sp) | [1, 56, 35, 95, 364, 1, 45, 1, 2, 3, 56] |
| Sp.insert(1, 56)  print(Sp) | [1, 56, 56, 35, 95, 364, 1, 45, 1, 2, 3, 56] |
| Sp.remove(56)  print(Sp) | [1, 56, 35, 95, 364, 1, 45, 1, 2, 3, 56] |
| Sp.pop(1)  print(Sp) | [1, 35, 95, 364, 1, 45, 1, 2, 3, 56] |
| print(Sp.index(56)) | 9 |
| print(Sp.count(35)) | 1 |
| Sp.sort()  print(Sp) | [1, 1, 1, 2, 3, 35, 45, 56, 95, 364] |
| Sp.sort(reverse = True) | [364, 95, 56, 45, 35, 3, 2, 1, 1, 1] |
| Sp.reverse()  print(Sp) | [364, 95, 56, 45, 35, 3, 2, 1, 1, 1] |
| Sp.clear()  print(Sp) | [] |

«Массивы»

Спросив у программиста на python что такое массив в этой языке он на вас посмотрит очень странно, потому что нет такого в python есть списки, разной вложенности. Я намерено не стал рассказывать об этом ранее, но можно создать список со списками. Проще если я все это продемонстрирую на примере и всё сразу станет ясно.

Возьмем следующие данные, полученные с термометров в разных регионах России:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Станция | День 1 | День 2 | День 3 | День 4 |
| **I** | -8 | -14 | -19 | -18 |
| **II** | 25 | 28 | 26 | 20 |
| **III** | 11 | 18 | 20 | 25 |

Давайте посмотрим, что мы получили в некоторые из дней, например 1 день на 3 станции, но у нас же еще нет данных, только таблица, ну мы ее обработаем сейчас.

Создадим список temp в который будем записывать списки, с температурами по станциям, один вложенный список = одна станция.

**temp=[[-8,-14,-19,-18],[25,28,26,20],[11,18,20,25]]**

Хорошо, список со списками у python разработчиков и массив у нормальных людей мы получили, теперь выведем нужные нам данные, но если написать **print(temp[2])**, то получить данные за 1 день будет проблемно, так как мы получим список значений, а не одно, для получения нужного мы воспользуемся двойной индексацией, а именно напишем вот так **print(temp[2][0])** и получим в итоговом выводе значение **11**. На основе этого мы можем понять, что индексы идут в порядке вложенности списков. Никто кроме здравого смысла нам не мешает сделать и список в списке в списке в списке, если это конечно имеет смысл.

Понимая, что массив это список со списками можем логично заключить, что на него работают все те же функции что уже были описаны в пункте выше.