Занятие 5. Встроенные коллекции (часть 1)

* list
* tuple
* list comprehension

**Списки и кортежи- это нумерованные наборы объектов.** Каждый элемент набора содер­жит лишь ссылку на объект. По этой nричине сnиски и кортежи могут содержать объекты nроизвольного типа данных и иметь неограниченную стеnень вложенности. Позиция эле­мента в наборе задается индексом. Обратите внимание на то, что нумерация элементов на­чинается с 0, а не с 1.

Списки и кортежи являются **упорядоченными** nоследовательностями элементов.   
 Как и все nоследовательности, они nоддерживают обращение к элементу по индексу, nолучение среза, конкатенацию (оnератор+), nовторение (оnератор\*), nроверку на вхождение (оnератор in).

Сnиски относятся к **изменяемым** тиnам данных.

## Списки

## 

Литералы   
 [ ]  
  
my\_supper\_list = [1,2,3,4,5]

print(my\_supper\_list)

Списки можно создавать не только вручную, но и с помощью функции. Чтобы создать пустой список, можно использовать list() или просто квадратные скобки []. Результат будет одинаковым.  
empty\_lst = list()

empty\_lst2 = []

Практика :

Cоздать список с элементами разного типа и назовем его **my\_list**. И с помощью изученного цикла **for** выясним тип данных для каждого элемента списка.

my\_list = [1, 1.4 , “name” , [1,2,3]]

for element in my\_list:

print(type(element), element)

Чтобы узнать длину списка, то есть сколько элементов в нем содержится, нужна команда **len(my\_list)**.

К элементам в списке мы можем обращаться по-разному. Не только с помощью циклов.   
в списке

**my\_list = [1, 1.4, 'line', [1, 2, 3], 1.4]**

строка “line” под индексом  **2**

print(my\_list[2])

my\_list[**0**] - обращение к элементу по индексу 0

индекс можно получить по методу   
**my\_list.index(1.4)**

если несколько значений то вернется индекс первого елемента в списке

my\_list.append(“another item”) - добавить в конец списка елемент  
  
  
  
**Списки могут хранить любой набор любого типа данных**

Также есть и специальная команда для удаления элементов из списка — **.remove()**, где в скобочках нужно указать, какой именно элемент должен быть удален из списка. Но если таких элементов в списке несколько, удалится только самый первый. Если вы попросите удалить элемент, которого нет в списке, то вы получите ошибку. Давайте удалим элемент **100** из списка.

**Сортировка списков**

Также списки можно сортировать с помощью функции **sorted()**. Давайте создадим новый список в числами **number\_list = [3, 108, 1, 245].** И применим к нему **sorted().** Мы увидим, в output отсортированный от меньшего к большему список.

Функция **sorted() -** вернет нам новый список отсортированный, а сам **number\_list -** останется неизмененным.

Здесь важно понимать, что после применения sorted() исходный **number\_list** на самом деле никак не изменило. Для того, чтобы сохранить список в отсортированном виде, нужно сохранить **sorted(number\_list)** в переменную.

Сортировать можно, кстати, не только числа в списках, но и строки.

У списка есть метод sort() - сортирет список в обьекте

Так записывать нельзя :

arr[] = 10 # приведет к ошибке

**Практика**

Давайте теперь попрактикуемся и попробуем написать небольшую программу. Она будет 5 раз просить пользователя ввести число и каждое из этих чисел добавлять в список. А потом распечатает нам список из этих 5 чисел.

**Груповое присваивание списков**

При создании сnиска в nеременной сохраняется ссылка на объект, а не сам объект. Это обя­ зательно следует учитывать nри груnnовом nрисваивании. Груnnовое nрисваиванне можно исnользовать для чисел и строк, но для сnисков этого делать нельзя.

x=y= [1,2,3] # создается один объект а не 2

x[2] = 100

изменится значение при обращении к каждой переменной

Как видно из nримера, изменение значения в nеременной у nривело также к изменению значения в nеременной х. Таким образом, обе nеременные ссылаются на один и тот же объ­ ект, а не на два разных объекта. Чтобы nолучить два объекта, необходимо nроизводить раз­ дельное nрисваивание:

x, y = [1,2] , [1,2]

Точно такая же ситуация возникает nри исnользовании оnератора nовторения \*.

arr = [ [] ] \* 2

Что будет в результате?

arr[0] .append(5)   
  
  
Список в скписке - вложеные списки  
lst = [ [1, 2, 3],

[4, 5, 6],

[7, 8, 9]  
 ]  
  
Вложенный список можо создать несколькими способами

arr = []

for i in range(2):   
 arr.append([])

Генератор списков   
  
lst = [ x for x in range(2) ]

Проверить, ссылаются ли две nеременнье на один и тот же объект, nозволяет оnератор **is** . Если nеременные ссылаются на один и тот же объект, то оnератор **is** возвращает значение **True**

**x=y=[1,2,3]**

**#?**

x is y

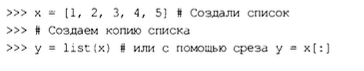
**Поверхностная копия списка**

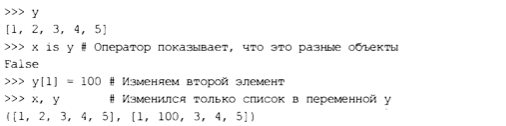
z = [1,2,3]

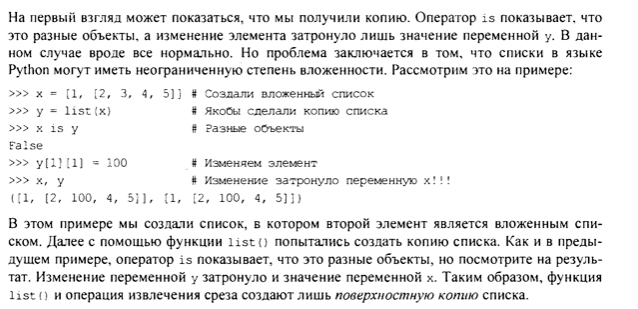
y = list(z)

x = y[:]

Сравните опратором **is** значения x , y, z  
  
  
Что будет?  
   
y[1] = 100







Чтобы получить полную копию списка можно использловать **deepcopy**() функцию из модуля **copy**

**import copy  
  
x = [1, 2, [3,4]]**

**y = copy.deepcopy(x)**

**y[1][1] =100**

Функция **deepcopy()** создает копию каждого объекта, при этом сохраняя внутреннюю структуру списка.   
 Иными словами, если в списке существуют два элемента. ссылающиеся на один объект, то будет создана копия объекта, и элементы будут ссылаться на этот новый объект, а не на разные объекты.

**x, \*y, z = [1,2,3,4,5]**

**\*x, y, z = [1,2,3,4,5]**

**x, y, \*z = [1,2,3,4,5]**

\*x,y,z = [1,2]

Обединить списки   
[1,2,3,] + [3,4,5]

* , +=

Повторить список указанное количество раз  
[1,2,3,4] \* 3

[Списки, методы](https://pythonworld.ru/tipy-dannyx-v-python/spiski-list-funkcii-i-metody-spiskov.html)

Многомерные массивы  
arr = [

[1,2,3],

[4,5,6],

[7,8,9]

]

arr = [ [1,2,3],[4,5,6],[7,8,9] ]

**Цыклы**

arr = [1,2,3,4]  
  
for i in arr( len(arr) -1 ):

arr[i] \*= 2

print(arr)

**Генераторы списков**

arr = [1,2,3,4]  
arr = [i\*2 for i in arr]  
Вместо изменения списка возвращают новый   
  
Генераторы работают быстрее циклов

arr= [1, 2, 3, 4]

arr = [ i \* 10 for i in arr if i %2 == О ]

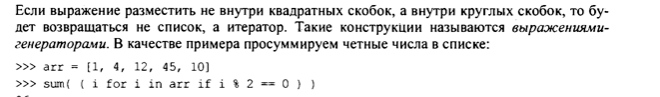
**усложняем**

arr = [[1, 2], [3, 4], [5, 6]]

arr = [ j \* 10 for i in arr for j in i if j % 2 == О]

p r i n t ( a r r )

**Генераторы выражений**

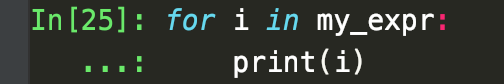
****

lst = [1,4,5,6,7]

my\_expr = ( i for i in lst if i % 2 == 0)

result = sum (my\_expr)

Протестировать что будет если еще раз вызвать последнюю строку

что будтет если   
 

**Кортежи** относятся к неизменяемым тиnам данных.   
Иными словами, можно получить эле­ мент по индексу, но изменить его нельзя

tuple()

(,)

(1,2,3,4)

запятые формируют кортеж

t = (5,)

n = () # пустой кортеж

Обращение по индексу

t[0]

[Котежи и методы](https://pythonworld.ru/tipy-dannyx-v-python/kortezhi-tuple.html)

[Больше об кортежах](https://www.bestprog.net/uk/2020/04/15/python-operations-on-tuples-bypass-of-tuple-methods-of-working-with-tuples-ua/)

Конкатенация +  
создается новый кортеж обьеденненный   
  
Срезы :  
  
t.index()

t.count()  
  
a = (2, 4, 6, 8, 10, 12)

b = [2, 4, 6, 8, 10, 12]

a.\_\_sizeof\_\_()

36

b.\_\_sizeof\_\_()

44