Список list

`list` это контейнер хранящия в себе элементы разделенные запятой. Элементы в списке находятся внутри квадратных скобок `[]`. Элементы в списке могут быть разных типов.

```
lst1 = [1, 2, 3, 4, 5, 6, 7, 8, 9]
lst2 = ['Hello', 'Hi', 'Goodbye', 'Bye']
lst3 = [1, 3.1415, 'Hello', True, [1, 2, 3, 4]]
```

Список без каких-либо элементов называется пустой список.

```
emtpy_list1 = []
empty_list2 = list()

print(empty_list1)
print(empty_list2)

>> []
>> []
```

На списки можно использовать оператор сложения `+`, чтобы соеденить один или больще списков вместе.

```
colors1 = ['Blue', 'White']
colors2 = ['Red', 'Yellow']
colors3 = ['Green', 'Black']

all_colors = colors1 + colors2 + colors3
print(all_colors)

>> ['Blue', 'White', 'Red', 'Yellow', 'Green', 'Black']
```

Также можно использовать, оператор умножения `*`, чтобы копировать список несколько раз.

```
numbers = [1, 2, 3, 4]

copy_numbers = numbers * 3
print(copy_numbers)

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

Индексинг и слайсинг

Индексинг в питоне во всех случаях начинается с нуля `0`. Положительный индекс отсчитывает элемент с начала, отрицательный - с конца. Поэтому у каждого элементы два альтернативных индекса.

```
colors = ['Red', 'Blue', 'Green', 'Yellow']

print(colors[0]) # Первый элемент

>> Red

print(colors[-1]) # Последний элемент

>> Yellow
```

Если заданный индекс превышает последний возможный индекс в списке, то будет выдана ошибка `IndexError`.

```
colors = ['Red', 'Blue', 'Green', 'Yellow']
print(colors[5])

>> Traceback (most recent call last):
     File "<pyshell#1>", line 1, in <module>
      colors[5]
   IndexError: list index out of range
```

Для выбора нескольких элементов исрользуется слайсинг при помощи двоиточия `:`. Синтакс `[i:j]` выбирает все элементы начиная с индекса `i` до `j`, не включая конечный индекс `j`.

```
colors = ['Red', 'Blue', 'Green', 'Yellow', 'Black', 'White']
print(colors[1:4])

>> ['Blue', 'Green', 'Yellow']
```

Также можно задать шаг выбора. Синтакс `[i:j:n]` выбирает каждый `n` элемент начиная с индекса `i` до `j`, не включая конечный индекс `j`.

```
colors = ['Red', 'Blue', 'Green', 'Yellow', 'Black', 'White', 'Cyan', 'Magenta']
print(colors[1:5:2])

>> ['Blue', 'Yellow']
```

Используя индексинт и слайсинг, возможно заменять элементы в списке.

```
colors = ['Red', 'Blue', 'Green', 'Yellow']
colors[1] = 'Black'
print(colors)

>> ['Red', 'Black', 'Green', 'Yellow']
##
colors = ['Red', 'Blue', 'Green', 'Yellow', 'Black', 'White']
colors[1:3] = ['Cyan', 'Magenta']
print(colors)

>> ['Red', 'Cyan', 'Magenta', 'Yellow', 'Black', 'White']
##
colors = ['Red', 'Blue', 'Green', 'Yellow', 'Black', 'White']
colors[1:5:2] = ['Cyan', 'Magenta']
print(colors)

>> ['Red', 'Cyan', 'Green', 'Magenta', 'Black', 'White']
```

Также при помощи коллекций можно обозначит несколько переменных сразу на одной строке кода.

> **Например: У нас имеется кортеж, обозначающий вектор с координатами `x` и `y`.**

```
vector = (3, 6)
x, y = vector
```

```
print(x)
print(y)

>> 3
>> 6
```

> **Анологично:**

```
vector = (3, 6)
x = vector[0]
y = vector[1]
print(x)
print(y)

>> 3
>> 6
```

Если количество переменных и элементов в коллекции не совпадают, то будет выдана ошибка `ValueError`.

```
vector = (3, 6, 4)
x, y = vector

>> Traceback (most recent call last):
    File "<pyshell#1>", line 1, in <module>
      x, y = vector
ValueError: too many values to unpack (expected 2)
###
x, y, z, s = vector

>> Traceback (most recent call last):
    File "<pyshell#2>", line 1, in <module>
      x, y, z, s = vector
ValueError: not enough values to unpack (expected 4, got 3)
```

Полезные фукции

del lst[*i*]

Удаляет элеметов с индексом `i` из списка `lst`.

```
colors = ['Blue', 'Red', 'Green', 'Yellow']
del colors[2]
print(colors)

>> ['Blue', 'Red', 'Yellow']
```

x in lst

Возвращает `True` если элемент `x` существует внутри списка `lst`. В противном случае возвращается `False`.

```
colors = ['Blue', 'Red', 'Green', 'Yellow']
print('Red' in colors)

>> True
```

x not in lst

Возвращает `True` если элемент `x` не существует внутри списка `lst`. В противном случае возвращается `False`.

```
colors = ['Blue', 'Red', 'Green', 'Yellow']
print('White' not in colors)

>> True
```

len(*lst*)

Возвращает длину списка `lst`.

```
lst = [1, 2, 3, 4]
print(len(lst))
```

```
>> 4
```

lst.append(*x*)

Добавляет элемент `x` в конец списка `lst`.

- ***x*: *AnyType*** → Добавляемый элемент

```
colors = ['Blue', 'Red', 'Green', 'Yellow']
colors.append('Black')
print(colors)

>> ['Blue', 'Red', 'Green', 'Yellow', 'Black']
```

lst.insert(*i, x*)

Вставляет элемент `x` в список `lst` на место индекса `i`.

- ***i*: *int*** → Индекс нового элемента
- ***x*: *AnyType*** → Элемент который нужно вставить

```
colors = ['Blue', 'Red', 'Green', 'Yellow']
colors.insert(3, 'Black')
print(colors)

>> ['Blue', 'Red', 'Black', 'Green', 'Yellow']
```

lst.extend(*L*)

Дополняет список `lst` списком `L`. То же самое что и `lst += L`.

- ***L*: *list*** → **Дополняющий список**

```
colors = ['Blue', 'Red', 'Green', 'Yellow']
extra_colors = ['Black', 'White']

colors.extend(extra_colors)
print(colors)

>> ['Blue', 'Red', 'Green', 'Yellow', 'Black', 'White']
```

lst.copy()

Создает копию списка `lst`. То же самое что и `lst[:]`.

```
colors = ['Blue', 'Red', 'Green', 'Yellow']
colors2 = colors.copy()

colors[2] = 'Black'

print('colors: ', colors)
print('colors2: ', colors2)

>> colors:  ['Blue', 'Red', 'Black', 'Yellow']
>> colors2:  ['Blue', 'Red', 'Green', 'Yellow']
```

---

lst.clear()

Опустошает список `lst`.

```
colors = ['Blue', 'Red', 'Green', 'Yellow']
colors.clear()

print(colors)

>> []
```

---

lst.remove(*x*)

Удаляет из списка первый по счету элемент `x`. Выдает ошибку `ValueError` если элемента не существует в списку.

- ***x*: *AnyType*** → Элемент в списке

```
colors = ['Blue', 'Red', 'Green', 'Yellow', 'Red']
colors.remove('Red')

print(colors)

>> ['Blue', 'Green', 'Yellow', 'Red']
```

lst.pop(*[i]*)

Выдает элемент под индексом `i` из списка `lst` и удаляет его из списка. По умолчанию, `i == -1`, поэтому выдается последний элемент.

- ***i*: *int*** → Индекс элемента

```
colors = ['Blue', 'Red', 'Green', 'Yellow']
color = colors.pop()

print('Retrieved color: ', color)
print('Remaining list: ', colors)

>> Retrieved color:  Yellow
>> Remaining list:  ['Blue', 'Red', 'Green']
```

lst.reverse()

Разворачивает список `lst`.

```
colors = ['Blue', 'Red', 'Green', 'Yellow']
colors.reverse()
print(colors)

>> ['Yellow', 'Green', 'Red', 'Blue']
```

lst.sort(*key=None, reverse=False*)

Сортирует список `lst` на основе функции `key`. Если функция не дана (`None`), то список сортируется в увеличительной последовательности (н.п. по алфавиту).

- ***key**: function → Сортирующая функция*
- ***reverse**: bool → Если `True`, то сортируется в обратной последовательности.*

```
colors = ['Blue', 'Red', 'Green', 'Yellow']
colors.sort()
print(colors)

>> ['Blue', 'Green', 'Red', 'Yellow']
```