

# ПРОГРАММИРОВАНИЕ

PYTHON

Списки (list):

Часть 1

Методы списков

(Лекция 8)

# **СПИСКИ (list)**

- 1. Что такое списки? (начальные сведения)**
- 2. Создание списков - конструктор list():**
- 3. Индексы**
- 4. Примеры списков**
- 5. Перебор списка с помощью цикла for**
- 6. Перебор списка с помощью цикла while**
- 7. Печать списков в столбик**
- 8. Методы и функции по работе со списками**
  - Append(item) -добавляет элемент item в конец списка**
  - insert(index, item): добавляет элемент item в список по индексу index**

- **index(item):** возвращает индекс элемента `item`
- **pop([index]):** удаляет и возвращает элемент по индексу `index`.
- **remove(item):** удаляет элемент `item`.
- Проверка наличия элемента
- Подсчет вхождений
- Сортировка
- Сортировка в обратном порядке
- Минимальное и максимальное значения

9. Тренажер работы со списками

10. Функции работы со списками `type()`; `len()`; `id()`.

11. Ввод списка строк через пробел

# 1. Что такое списки? (начальные сведения)

Список (list) представляет тип данных, который хранит набор или последовательность элементов.

Для создания списка в квадратных скобках через запятую перечисляются все его элементы.

Во многих языках программирования есть аналогичная структура данных, которая называется массив. Например, определим список чисел:

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

## 2. Создание списков - конструктор list():

Также для создания списка можно использовать конструктор list():

```
numbers1 = []  
numbers2 = list()
```

Оба этих определения списка аналогичны - они создают пустой список.

### 3. Индексы

Для обращения к элементам списка надо использовать **индексы**, которые представляют номер элемента в списка.

Индексы начинаются с нуля. То есть второй элемент будет иметь индекс 1. Для обращения к элементам с конца можно использовать отрицательные индексы, начиная с -1. То есть у последнего элемента будет индекс -1, у предпоследнего - -2 и так далее.

```
numbers = [1, 2, 3, 4, 5]
print(numbers[0]) # 1
print(numbers[2]) # 3
print(numbers[-3]) # 3
```

```
numbers[0] = 125 # изменяем первый элемент списка
print(numbers[0]) # 125
```

## 4. Примеры списков

Список необязательно должен содержать только однотипные объекты. Мы можем поместить в один и тот же список одновременно строки, числа, объекты других типов данных:

```
a = [2, 2.25, "Python"]  
objects = [1, 2.6, "Hello", True]
```

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

```
b = [2.71828 , 3.14159 , -1]
```

```
c = ['Vera ', 'Nadezhda ', 'Lyubov ']
```

```
d = [True , False ]
```

```
f = [1, True , 2.333 , 'abcdefgh ']
```

# ПОВТОРЕНИЕ тема оператор цикла

for

```
for x in ["A",5,True,0.75,"Vasja"]:
```

```
    print(x, type(x))
```

```
# A <class 'str'>
```

```
# 5 <class 'int'>
```

```
# True <class 'bool'>
```

```
# 0.75 <class 'float'>
```

```
# Vasja <class 'str'>
```



```
lst = ["Москва", "Санкт-Петербург", "Тверь",  
"Казань"]
```

```
lst = ["Москва", "Санкт-Петербург", "Тверь", "Казань"]
```



#1

```
lst = ["Москва", "Санкт-Петербург", "Тверь",  
"Казань"]
```

```
print('lst=',lst,type(lst),'len(lst)=' ,len(lst))
```

```
print(lst[0])
```

```
print(lst[1])
```

```
print(lst[len(lst)-1])
```

```
print(lst[-1])
```

## 5. Перебор списка с помощью цикла

**for**

**#6**

```
companies = ["Microsoft", "Google", "Oracle",  
"Apple"]
```

```
for item in companies:
```

```
    print(item)
```

```
"""
```

Microsoft

Google

Oracle

Apple

```
"""
```

#7

#ПЕРЕБОР – ПЕЧАТЬ ЭЛЕМЕНТОВ СПИСКА

# для перебора элементов списка в Python

# очень удобно использовать цикл for:

```
lst = ["Москва", "Санкт-Петербург", "Тверь", "Казань"]
```

```
for city in lst:
```

```
    print(city)
```

```
    print(city,type(city),id(city))
```

## 6. Перебор с помощью цикла while:

#8

|||||

Для перебора с помощью функции len() получаем длину списка. С помощью счетчика i выводит по элементу, пока значение счетчика не станет равно длине списка.

|||||

```
companies = ["Microsoft", "Google", "Oracle",  
"Apple"]
```

```
i = 0
```

```
while i < len(companies):
```

```
    print(companies[i])
```

```
    i += 1
```

## 7. Печать списков в столбик

#9 Функция. Печать списка группы - передаем список

#Вариант 1 for

```
spis_gr_1=["Иванов","Петров","Сидоров","Аверин","  
Куценко"]
```

```
spis_gr_2=["Пугачева","Киркоров","Маменко","  
Булитко"]
```

```
def print_spisok(lst):
```

```
    n=0
```

```
    for st in lst:
```

```
        print(n,st)
```

```
        n+=1
```

```
print_spisok(spis_gr_1)
```

```
#10 Функция. Печать списка группы - передаем список
#Вариант 2 for
spis_gr_1=["Иванов","Петров","Сидоров","Аверин","
Куценко"]
spis_gr_2=["Пугачева","Киркоров","Маменко","Булитко"]

def print_spisok(lst):
    for i in range(len(lst)):
        print(i-1,lst[i])

print_spisok(spis_gr_1)
```

#11 Функция. Печать списка группы - передаем список

#Вариант 3 while

```
spis_gr_1=["Иванов","Петров","Сидоров","Аверин","  
Куценко"]
```

```
spis_gr_2=["Пугачева","Киркоров","Маменко","  
Булитко"]
```

```
def print_spisok(lst):
```

```
    i=0
```

```
    while i < len(lst):
```

```
        print(i, lst[i])
```

```
        i += 1
```

```
print_spisok(spis_gr_1)
```

## 8. Методы и функции по работе со списками

# 12. **append(item)**: добавляет элемент `item` в конец списка

```
users = ["Tom", "Bob"]
```

# добавляем в конец списка

```
users.append("Alice")
```

```
print(users)
```

```
# ["Tom", "Bob", "Alice"]
```



```
#13 Метод append() вставляет в конец
# исходного списка значение аргумента.
ist_000001=["Иванов","Петров"]
print(ist_000001,len(ist_000001),type(ist_000001),id
(ist_000001))
#['Иванов', 'Петров'] 2 <class 'list'> 49077960
ist_000001.append("Сидоров")
print(ist_000001,len(ist_000001),type(ist_000001),id
(ist_000001))
#['Иванов', 'Петров', 'Сидоров'] 3 <class 'list'>
49077960
```

```
def eggs(someParameter):  
    someParameter.append('Hello')
```

```
spam = [1, 2, 3]  
eggs(spam)  
print(spam)  
#[1, 2, 3, 'Hello']
```

|||||

**14. insert(index, item): добавляет  
ЭЛЕМЕНТ**

**item в список по индексу index**

|||||

# добавляем на вторую позицию

```
users=["Tom", "Bob", "Alice"]
```

```
users.insert(1, "Bill")
```

```
print(users)
```

```
# ["Tom", "Bill", "Bob", "Alice"]
```

|||||

**15. `index(item)`: возвращает индекс элемента `item`.**

Если элемент не найден, генерирует исключение `ValueError`

|||||

```
# получаем индекс элемента
users=["Tom", "Bill", "Bob", "Alice"]
i = users.index("Tom")
print(i) # 0
```

|||||

## 12. `pop([index])`: удаляет и возвращает элемент

по индексу `index`. Если индекс не передан, то просто удаляет последний элемент.

|||||

```
# # удаляем по этому индексу
users=["Tom", "Bill", "Bob", "Alice"]
i=2
removed_item = users.pop(i)
print(removed_item)#Bob
print(users)
#['Tom', 'Bill', 'Alice']
```

|||||

**13. remove(item): удаляет элемент item.**

Удаляется только первое вхождение элемента.

Если элемент не найден,  
генерирует исключение ValueError

|||||

```
users=["Tom", "Bill", "Bob", "Alice"]
# удаляем последний элемент
last_user = users[-1]
print(last_user) #Alice
users.remove(last_user)
print(users)
['Tom', 'Bill', 'Bob']
```

## #14. Проверка наличия элемента

"""

Проверка наличия элемента Если определенный элемент не найден, то методы `remove` и `index` генерируют исключение. Чтобы избежать подобной ситуации, перед операцией с элементом можно проверять его наличие с помощью ключевого слова `in`:

"""

```
companies = ["Microsoft", "Google", "Oracle", "Apple"]
```

```
item = "Oracle" # элемент для удаления
```

```
if item in companies:
```

```
    companies.remove(item)
```

```
print(companies)
```

```
#[ 'Microsoft', 'Google', 'Apple' ]
```

#Выражение `item in companies` возвращает `True`, если элемент `item` имеется в списке `companies`. Поэтому конструкция `if item in companies` может выполнить последующий блок инструкций в зависимости от наличия элемента в списке.

## #15. Подсчет вхождений

''''''

`count(item)`: возвращает количество вхождений элемента `item` в список

Если необходимо узнать, сколько раз в списке присутствует тот или иной элемент, то можно применить метод `count()`:

''''''

```
users = ["Tom", "Bob", "Alice", "Tom", "Bill", "Tom"]
users_count = users.count("Tom")
print(users_count)    # 3
```



## #16 Сортировка

"""

Для сортировки по возрастанию применяется

метод `sort()`:

"""

```
users = ["Tom", "Bob", "Alice", "Sam", "Bill"]
```

```
users.sort()
```

```
print(users)    # ["Alice", "Bill", "Bob", "Sam", "Tom"]
```

## #17. Сортировка в обратном порядке

''''''

Если необходимо отсортировать данные в обратном порядке, то мы можем после сортировки применить метод `reverse()`:

''''''

```
users = ["Tom", "Bob", "Alice", "Sam", "Bill"]
users.reverse()
print(users)
# ["Tom", "Sam", "Bob", "Bill", "Alice"]
```

## #18. Минимальное и максимальное значения

''''''

Встроенный функции Python `min()` и `max()`

позволяют найти минимальное и максимальное значения

соответственно:

''''''

```
numbers = [9, 21, 12, 1, 3, 15, 18]
```

```
print(min(numbers)) # 1
```

```
print(max(numbers)) # 21
```

## 9. Тренажер работы со списками

#Тренажер работы со списками

```
spis_gr=["Иванов","Петров","Сидоров","Аверин",  
        "Куценко","Пугачева",  
        "Киркоров","Маменко","Булитко"]
```

```
lst_0001=["Иванов","Петров"]
```

```
def print_spisok(lst):
```

```
    i=0
```

```
    while i < len(lst):
```

```
        print(i+1, lst[i])
```

```
        i += 1
```

```
menu=""
```

МЕНЮ ПРОГРАММЫ "Список Группы":

1 - вывести на печать текущий список в строку;

2 - вывод списка в столбик

3- добавить в список;

4 - отсортировать список по возрастанию;

5 - отсортировать список по убыванию;

6 - число элементов в списке;

Введите число или пробел - для выхода из программы:

```
'''
```

```
def work_with_list(lst):
    print(menu)
    while True:
        z = input('Введите режим: ')
        if z=='1':
            print(lst)
        elif z=='2':
            print_spisok(lst)
        elif z=='3':
            name= input("Введите фамилию: ")
            lst.append(name)
        elif z=='4':
            lst.sort()
        elif z=='5':
            lst.reverse()
        elif z=='6':
            print(len(lst))
        elif z=='':
            break
        else:
            print("Нет такой цифры")
    print("Работа программы закончена!")
```

```
work_with_list(spis_gr)
```

```
import copy
spam = ['A', 'B', 'C', 'D']
print(spam,id(spam))
cheese = copy.copy(spam)
print(cheese,id(cheese))
cheese[1] = 42
print(spam,id(spam))
print(cheese,id(cheese))
```

```
# spam ['A', 'B', 'C', 'D']
# cheese ['A', 42, 'C', 'D']
```

```
#Исходный список ist_0001_old не изменяется
# ДО вызова функции work_with_list(lst)
# ist_0001_old: ['Иванов', 'Петров'] 14707176
# ist_0001_old_copy: ['Иванов', 'Петров'] 14707112
# ПОСЛЕ вызова функции work_with_list(lst)
# ist_0001_old: ['Иванов', 'Петров'] 14707176
# ist_0001_old_copy: ['Иванов', 'Петров', 'Kozlovsky'] 14707112
# ist_0001_new: ['Иванов', 'Петров', 'Kozlovsky'] 14707112
import copy
ist_0001_old=["Иванов","Петров"]
ist_0001_old_copy = copy.copy(ist_0001_old)
def print_spisok(lst):
    i=0
    while i < len(lst):
        print(i+1, lst[i])
        i += 1

menu=""
МЕНЮ ПРОГРАММЫ "Список Группы":
1 - вывести на печать текущий список в строку;
2 - вывод списка в столбик
3- добавить в список;
4 - отсортировать список по возрастанию;
5 - отсортировать список по убыванию;
6 - число элементов в списке;
Введите число или пробел - для выхода из программы:
'''
```

```
def work_with_list(lst):
    print(menu)
    while True:
        z = input('Введите режим: ')
        if z=='1':
            print(lst)
        elif z=='2':
            print_spisok(lst)
        elif z=='3':
            name= input("Введите фамилию: ")
            lst.append(name)
        elif z=='4':
            lst.sort()
        elif z=='5':
            lst.reverse()
        elif z=='6':
            print(len(lst))
        elif z=="":
            break
        else:
            print("Нет такой цифры")
    print("Работа программы закончена!")
    return lst
```

```
print("ist_0001_old:",ist_0001_old,id(ist_0001_old))
print("ist_0001_old_copy:",ist_0001_old_copy,id(ist_0001_old_copy))
# ist_0001_old: ['Иванов', 'Петров'] 14707176
# ist_0001_old_copy: ['Иванов', 'Петров'] 14707112
ist_0001_new= work_with_list(ist_0001_old_copy)
print("ist_0001_old:",ist_0001_old,id(ist_0001_old))
print("ist_0001_old_copy:",ist_0001_old_copy,id(ist_0001_old_copy))
print("ist_0001_new:",ist_0001_new,id(ist_0001_new))
# ist_0001_old: ['Иванов', 'Петров'] 14707176
# ist_0001_old_copy: ['Иванов', 'Петров', 'Kozlovsky'] 14707112
# ist_0001_new: ['Иванов', 'Петров', 'Kozlovsky'] 14707112
```



## 10 Ввод списка строк через пробел

#Использование split()

''''''

Метод Python split string разбивает строку с помощью указанного спецсимвола и возвращает список (массив) подстрок.

''''''

```
z="ff gg hh jj"
```

```
A=z.split()
```

```
print(A)
```

```
#[ 'ff', 'gg', 'hh', 'jj']
```