



# Обобщение пройденного

---

# Типы данных

Числа: 5 9892 10.2 0.342

Строки: 'строка' 'число номер 45'

Списки: [ 'строка', 5, 'Строка номер 2' ]

Кортеж: 5, 3, 10.1 (5, 3, 10.1) ('один элемент',)

Словарь: { 'Ключ': 'Значение', 'Ключ2': 'Значение2' }

Множества: { 'строка1', 'строка2', 'строка3' }

Булевы: True False

Отсутствие данных: None

# Переменные

Имя переменной: `perem1`, `color`, `colors`, `фрукт`, `фрукты`

Присваивание и выражения :

```
chislo = 5   stroka = 'Привет Мир'   spisok = [ 4, 5, 6]
```

```
a = b + c * d   a = (b+c) * d   stroka = 'STRO' + 'KA'
```

```
a = b = 3
```

# Простые типы данных

Числа	Булевы
<code>chislo = 5      float = 2.4</code>	<code>bool_type = True      bool_type = False</code>
<code>chislo = chislo * 5 #25      chislo = chislo**2    #25</code>	<code>a = 1 + True # 2</code>
<code>chislo = 5/2    # 2.5      chislo = 5//2    #2</code>	<code>a = 1 + False # 1</code>
<code># пример с условием</code>	<code>if a &gt; 1:     print(True) # пример с условием</code>
<code>a = a + 1      =&gt;      a += 1</code>	
<code>a = a * 1      =&gt;      a *= 1</code>	
<code>chislo = 1 + '23' # error    chislo = 1 + int('23')</code>	

# Последовательности

Строки	Кортежи	Списки
<code>S = 'Привет мир'</code>	<code>tuple1 = ('привет', 'мир')</code>	<code>spisok = ['привет', 'мир']</code>
<code>S += ' Здравствуй'</code>	<code>tuple1 += (' Здравствуй')</code>	<code>spisok += [' Здравствуй']</code>
<code>S[1]      S[7] = 'п' #error</code>	<code>tuple1[1]   tuple[1] = 'пир' #error</code>	<code>spisok[1]      spisok[1] = 'пир'</code>
<code>S[7:10]   # 'мир'</code>	<code>tuple1[0:2]   # ('привет', 'мир')</code>	<code>spisok[0:2]   # ['привет', 'мир']</code>
<code>len(S)      # 21</code>	<code>len(tuple1)   # 3</code>	<code>len(spisok)   # 3</code>
<code>S.count('п')   # 3</code>	<code>tuple1.count('мир')   # 1</code>	<code>spisok.count('мир')   # 1</code>
<code>S.index('и')   # 2   S.index('и',3)   #8</code>	<code>tuple1.index('мир')   # 1</code>	<code>spisok.index('мир')   # 1</code>

# Неупорядоченные массивы

Соварь	Множество
<code>slovar = {'ключ': 'значение', 'ключ2': 'значение2'}</code>	<code>mnoz = {'значение', 'значение2'}</code>
<code>slovar = { }</code> <code>slovar = dict()</code>	<code>mnoz = set()</code>
работает с for	работает с for
<code>slovar.pop('ключ2')</code> # 'значение2'	<code>mnoz.pop('значение2')</code> # 'значение2'
<code>len(slovar)</code> # 2	<code>len(mnoz)</code> # 2

# Изменяемые и неизменяемые Типы данных

Изменяемые	Неизменяемые
Список	Числа
Словарь	Строки
Множества	Кортеж
	Булевые
	None

Изменяемые:

```
spisok = [5, 1, 4, 2]
spisok.sort()
print(spisok) # [1,2,4,5]
spisok2 = spisok.sort()
print(spisok2) # None
```

Неизменяемые:

```
stroka = "Привет"
stroka.replace('Омл', 'Прив')
print(stroka) # Привет
stroka = stroka.replace('Омл', 'При')
print(stroka) # Омлет
```

# Инструкции

<https://pythonworld.ru/osnovy/klyuchevye-slova-modul-keyword.html>

Инструкция удаление: `del perem`      `del spisok[1:4]`      `del slovar['ключ']`

Условия: `if`, `else`, `elif`

Проверка в условии: `in`, `is`

Инструкции в условии: `or`, `and`, `not`

Циклы: `for in`, `while`, `continue`, `break`

Подключение пакетов, библиотек: `import`, `from import`

Другие: `global`, `def`, `return`, `class`, `with`



# Блочные инструкции

Блочные инструкции: if, elif, else, while, for, def, class, with

Правила: 1)Блок начинается с :

2)Все, что внутри блока должно иметь отступ

3)Блок завершается возвратом от отступа

# Вложенности в блоке

```
if a>b:  
    print(a)  
    if c>b:  
        print(c)  
    else:  
        print(b)  
else:  
    print(d)
```

# Условия

## Пять форм условия:

1. if a > b:  
    print(a)

---

2. if a > b:  
    print(a)  
    else:  
        print(b)

---

3. if a > b:  
    print(a)  
    elif a < b:  
        print(b)  
    else:  
        print(c)

## Однострочный if:

4. print(a) if a > b

5. print(a) if a > b else print(b)

## Аналог пункта 3:

```
if a > b:  
    print(a)  
else:  
    print(b) if a < b else print(c)
```

# Выражение в условии

Где применяются:

if (True или False):

```
print(True)
```

while (True или False):

```
print(True)
```

Виды выражений:

1)  $a > b$ ,  $a > b > c$ ,  $a \geq b$

2)  $a$  is  $b$

3)  $a$  in  $b$  # stroka = 'мир' 'и' in stroka

4)  $a$

5)  $a > b$  and  $b < c$        $a < b$  and  $c$

6)  $a > b$  or  $b < c$

7) not  $a$       not  $a > b$

# Циклы

`while` (условие):

    блок

`for` (переменная) `in` (итерируемый тип данных):

    блок

`break` - останавливает выполнение итераций

`continue` - пропускает оставшиеся инструкции и выполняет итерации дальше

**Итерируемые типы данных:**

Строки	проходит по символам
Списки	проходит по элементам
Словари	проходит по ключам
Кортежи	проходит по элементам
Множества	проходит по элементам
<code>range()</code>	проходит по диапазону чисел (на самом деле, это тип данных схожий со списком)

# Функции

- 1) Встроенные функции - функции доступные в питоне из коробки
- 2) Функции из библиотек - функции которые импортируем из пакетов python
- 3) Встроенные методы - функции привязанные к типу данных(к объекту)
- 4) Пользовательские функции - наши собственные функции
- 5) Пользовательские методы - наши собственные методы

# Функции

`function(аргумент1, аргумент2)`

функция возвращает ответ подобно выражению.

возвращает ответ инструкцией `return`

ответом может быть любой из типов данных

используя встроенные функции мы их просто вызываем. Описание встроенных функций спрятано от нас.

аргументов может быть и ноль и сколь угодно

# Встроенные функции

<https://pythonworld.ru/osnovy/vstroennye-funkcii.html>

<code>print()</code> , <code>input()</code>	функции ввода и вывода
<code>int()</code> , <code>float()</code> , <code>str()</code> , <code>list()</code> , <code>dict()</code> , <code>list()</code> , <code>set()</code> , <code>tuple()</code>	функции типов данных
<code>min()</code> , <code>max()</code>	минимальный и максимальные элемент послед-ти
<code>len()</code>	возвращает длину строки или массива
<code>abs()</code>	Возвращает абсолютную величину (модуль числа).
<code>sum()</code>	Возвращает сумму элементов последоват-ти
<code>type()</code>	Возвращает тип данных
<code>eval()</code>	Выполняет инструкцию из строки



# Методы

Методы это функции, которые привязаны к типу данных (объекту)

Обращаемся к методам через точку.

```
spisok.sort()
```

# Методы строк

<code>S.find(str, [start],[end])</code>	Поиск подстроки в строке. Возвращает номер первого вхождения или -1
<code>S.rfind(str, [start],[end])</code>	Поиск подстроки в строке. Возвращает номер последнего вхождения или -1
<code>S.index(str, [start],[end])</code>	Поиск подстроки в строке. Возвращает номер первого вхождения или вызывает <code>ValueError</code>
<code>S.rindex(str, [start],[end])</code>	Поиск подстроки в строке. Возвращает номер последнего вхождения или вызывает <code>ValueError</code>
<code>S.replace(шаблон, замена)</code>	Замена шаблона
<code>S.split(символ)</code>	Разбиение строки по разделителю
<code>S.upper()</code> , <code>S.lower()</code>	Преобразование строки к верхнему и нижнему регистру
<code>S.join(список)</code>	Сборка строки из списка с разделителем <code>S</code>
<code>S.format()</code>	форматирование строки
<code>S.strip([chars])</code>	удаление символов.

# Методы списков

<code>list.append(x)</code>	Добавляет элемент в конец списка
<code>list.extend(L)</code>	Расширяет список <code>list</code> , добавляя в конец все элементы списка
<code>list.insert(i, x)</code>	Вставляет на <code>i</code> -ый элемент значение <code>x</code>
<code>list.remove(x)</code>	Удаляет первый элемент в списке, имеющий значение <code>x</code> .
<code>list.pop([i])</code>	Удаляет <code>i</code> -ый элемент и возвращает его
<code>list.index(x, [start [, end]])</code>	Возвращает положение первого элемента со значением <code>x</code>
<code>list.count(x)</code>	Возвращает количество элементов со значением <code>x</code>
<code>list.sort()</code>	Сортирует список
<code>list.reverse()</code>	Разворачивает список
<code>list.clear()</code>	Очищает список

# Методы словарей

<code>dict.clear()</code>	очищает словарь
<code>dict.get(key[, default])</code>	возвращает значение ключа
<code>dict.items()</code>	возвращает список кортежей (ключ, значение)
<code>dict.keys()</code>	возвращает список ключей словаря
<code>dict.values()</code>	возвращает список значений словаря
<code>dict.update([other])</code>	добавляет в словарь ключ значения из другого
<code>dict.pop(key[, default])</code>	удаляет указанный элемент в словаре