

## Лекции 1, 2

# Введение в программирование на языке Python

*Язык Python (произносится как Пайтон или Питон) появился в 1991 году и был разработан Гвидо ван Россумом. Язык назван в честь шоу "Летающий цирк Монти Пайтона".*

*Python – высокоуровневый язык, поддерживающий императивное, объектно-ориентированное и функциональное программирование.*

# Ссылки на ПО

Интерпретатор языка Python 3:  
<https://www.python.org/downloads/>



## Download PyCharm

[Windows](#)

[macOS](#)

[Linux](#)

### Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

[Download](#)

Free trial

### Community

For pure Python development

[Download](#)

Free, built on open-source

Среда программирования  
JetBrains PyCharm (Community):  
[https://www.jetbrains.com/pycharm/  
download](https://www.jetbrains.com/pycharm/download)

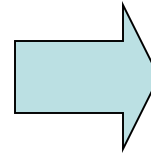
Version: 2021.2.1  
Build: 212.5080.64  
27 August 2021

[System requirements](#)

[Installation Instructions](#)

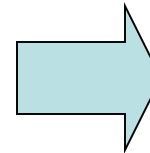
# 1 Последовательность операторов

```
a = 1
b = 2
a = a + b
b = a - b
a = a - b
print(a,b)
print("a=",a,"b=",b)
print(a+b)
```



```
2 1
a= 2 b= 1
3
```

```
a = 1
  b = 2
a = a + b
  b = a - b
a = a - b
  print(a,b)
print("a=",a,"b=",b)
  print(a+b)
```

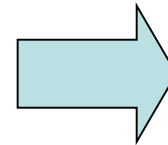


**ОШИБКА!**

Все операторы,  
входящие в  
последовательность  
действий, должны  
иметь один и тот же  
отступ

## 2 Ввод-вывод данных (консоль)

```
1 print("Hello")
2 print("A", "B", "C")
3 print("A" + "B" + "C")
4
5 # замена разделителя (sep)
6 print("A", 'B', "C", sep="#")
7
8 # замена символа перевода строки (end)
9 print("123", end="- ")
10 print("456")
```

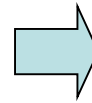


```
Hello
A B C
ABC
A#B#C
123-456
```

```

1 a = input()
2 print ('a =', a)
3
4 val = int(input())
5 print ('val =', val)
6
7 # строка-приглашение
8 s = int(input("input number: "))
9 print ('s =', s)
10
11 #Преобразование строки в список
12 print('Преобразование строки в список:')
13 #разделитель пробел
14 l1 = input().split()
15 print ('l1 =', l1)
16 #разделитель "-"
17 l2 = input().split("-")
18 print ('l2 =', l2)
19 #считывания списка с приведением к int
20 l3 = map(int, input().split())
21 print ('l3 =', list(l3))

```



```

Python
a = Python
44
val = 44
input number: 5
s = 5
Преобразование строки в список:
11 33 gg 4 f
l1 = ['11', '33', 'gg', '4', 'f']
11-33-55-ggg-r-4
l2 = ['11', '33', '55', 'ggg', 'r', '4']
44 55 1 7 4 3
l3 = [44, 55, 1, 7, 4, 3]

```

# 3 Основные типы данных



```
a = 5
print(a, "is of type", type(a))
a = 2.0
print(a, "is of type", type(a))
a = 1+2j
print(a, "is complex number?", isinstance(1+2j, complex))
```



```
5 is of type <class 'int'>
2.0 is of type <class 'float'>
(1+2j) is complex number? True
```

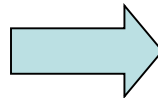
**type()** -  
определение  
класса  
переменной или  
значения

**isinstance()** -  
проверка  
принадлежности  
объекта  
определённому  
классу

## Динамическая типизация объектов:

- любой объект является ссылкой;
- типом объекта является то, на что он ссылается;
- тип объекта может произвольно меняться по ходу выполнения кода, когда ссылка начинает ссылаться на другой объект (например, в результате операции присвоения).

```
a=5.555  
b=3  
print(type(b))  
b=a  
print(type(b))
```



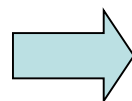
```
<class 'int'>  
<class 'float'>
```

**СПИСКИ (list)** - упорядоченные коллекции объектов *произвольных* ТИПОВ

**Пример** создания списка: `a = [1, 2.2, 'python']`

Оператор `[ ]` – извлечение элемента (“доступ по индексу”) или диапазона элементов (“извлечение среза”)

```
1 a = [5,10,15,20,25,30,35,40]
2 print("a[2] = ", a[2])
3 print("a[0:3] = ", a[0:3])
4 print("a[1:3] = ", a[1:3])
5 print("a[5:] = ", a[5:])
```

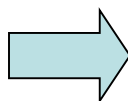


```
a[2] = 15
a[0:3] = [5, 10, 15]
a[1:3] = [10, 15]
a[5:] = [30, 35, 40]
```

```
a = [5,10,15,20,25,30,35,40]

# изменение значения
a[1]='Python'

print(a)
```



```
[5, 'Python', 15, 20, 25, 30, 35, 40]
```



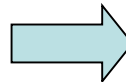
## Некоторые методы списков:

**append()** - добавляет элемент в конец списка;

**reverse()** – разворачивает список;

**count(x)** - возвращает количество элементов со значением x.

```
1 a = [1, 2.2, 'python', 1]
2
3 a.append('PYTHON')
4 print(a)
5 a.reverse();
6 print(a)
7
8 print(a.count('python'))
9 print(a.count(1))
```



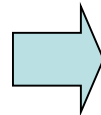
```
[1, 2.2, 'python', 1, 'PYTHON']
['PYTHON', 1, 'python', 2.2, 1]
1
2
```

**КОРТЕЖИ (tuple)** - упорядоченная последовательность неизменяемых элементов

**Пример** создания кортежа: `a = (1, 2.2, 'python')`

Для извлечения элементов используется оператор `[ ]`

```
t = (5, 'program', 1+3j)
print("t[1] =", t[1])
print("t[0:3] =", t[0:3])
```



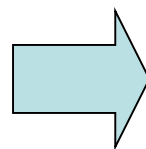
```
t[1] = program
t[0:3] = (5, 'program', (1+3j))
```

```
# Приводит к ошибке, т.к.
# кортежи неизменяемы
t[0] = 10
```

## СТРОКА (str) - последовательность СИМВОЛОВ

### Примеры создания строк:

```
1 s1 = "Строка в двойных кавычках"
2
3 s2 = 'Строка в одинарных кавычках'
4
5 s3 = """Многострочная
6 строка"""
7
8 s4 = '''Ещё одна
9 многострочная
10 строка'''
11
12 s5 = 'очень длинная \
13 строка'
14
15 print(s1)
16 print()
17 print(s4)
18 print()
19 print(s5)
```



```
Строка в двойных кавычках

Ещё одна
многострочная
строка

очень длинная строка
```

# Базовые операции над строками:

## 4) доступ по индексу

### 1) конкатенация

```
S1 = 'Cat'  
S2 = 'Dog'  
print(S1 + S2)
```

CatDog

```
S = 'PythoN'  
print (S[0], S[3], S[-2])
```

P h o

## 5) извлечение среза [X:Y:<ШАГ>]

X – начало среза, а Y – окончание

### 2) дублирование строки

```
print('Str' * 3)
```

StrStrStr

### 3) длина строки

```
print(len('Str' * 3))
```

9

```
1 s = "Hellow, world!"  
2  
3 print(s[3:5])  
4 print(s[2:-2])  
5 print(s[:6])  
6 print(s[1:])  
7 print(s[:])  
8 print()  
9  
10 print(s[::-1])  
11 print(s[1:7:3])  
12 print(s[2::2])
```

```
lo  
llow, worl  
Hellow  
ellow, world!  
Hellow, world!  
  
!dlrow ,wolleH  
eo  
lo, wrd
```

# Функции и методы создают новую строку, а не меняю текущую

## Встроенные функции строк:


**len(S)** – длина строки S;

**ord(C)** - преобразует символ C в целое число;

**chr(D)** - преобразует целое число D в символ;

**str(O)** - изменяет тип объекта O на str.

```
1 S = "Программирование \
2   на языке Python"
3
4 print( len(S) )
5 print( ord(S[5]) )
6 print( chr(65) )
7 print( "Text " + str( 5.54321 ) )
```



```
32
65
A
Text 5.54321
```

```
Программирование на языке Python
Программирование на языке Python
Программирование на языке Python
Программирование на языке python
False
True
```

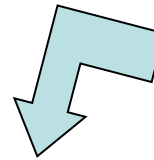
## Некоторые методы строк:

**lstrip()** – удаление пробельных символов в начале строки;

**capitalize()** - переводит первый символ строки в верхний регистр, а все остальные в нижний ;

**isdigit()** - состоит ли строка из цифр?

```
1 S = "    Программирование \
2   на языке Python"
3
4 print( S )
5 print( S.lstrip() )
6 print( S )
7
8 S1 = S.lstrip()
9 print ( S1.capitalize() )
10
11 S2 = '12345'
12 print ( S1.isdigit() )
13 print ( S2.isdigit() )
```



**МНОЖЕСТВО (set)** - неупорядоченная последовательность  
(дубликаты автоматически удаляются)

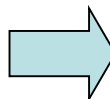
**Пример** создания множества:  $a = \{5, 2, 3, 1, 4\}$

```
1 a = {5, 2, 3, 1, 4, 5, 5}
2
3 # вывод переменной множества
4 print("a =", a)
5
6 # тип данных переменной a
7 print(type(a))
```



```
a = {1, 2, 3, 4, 5}
<class 'set'>
```

```
# множество значений разных типов
my_set = {1.0, "Hello", (1, 2, 3)}
print(my_set)
```

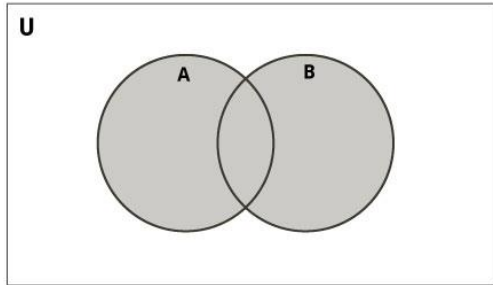


```
{'Hello', 1.0, (1, 2, 3)}
```

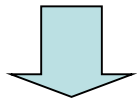
Оператор [ ] нельзя применять

# Операции над множествами

## 1) Объединение

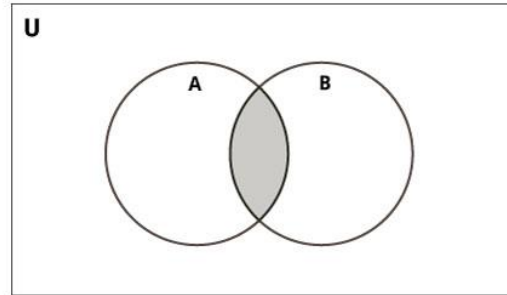


```
1 A = {1, 2, 3, 4, 5}
2 B = {4, 5, 6, 7, 8}
3
4 print(A | B)
5 print(A.union(B))
6 print(B.union(A))
```



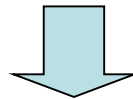
```
{1, 2, 3, 4, 5, 6, 7, 8}
{1, 2, 3, 4, 5, 6, 7, 8}
{1, 2, 3, 4, 5, 6, 7, 8}
```

## 2) Пересечение



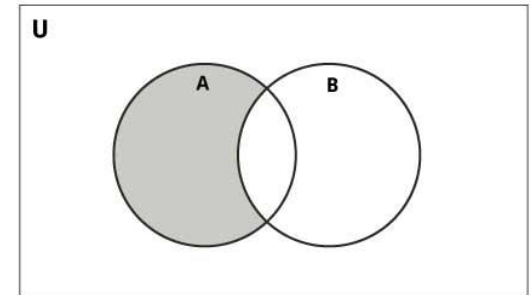
```
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

print(A & B)
print(A.intersection(B))
print(B.intersection(A))
```



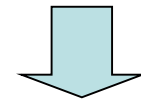
```
{4, 5}
{4, 5}
{4, 5}
```

## 3) Определение разницы множеств



```
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

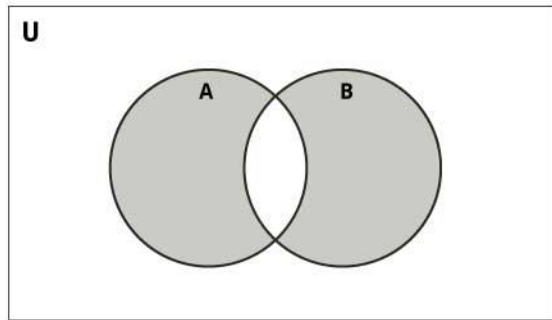
print(A - B)
print(A.difference(B))
print(B.difference(A))
```



```
{1, 2, 3}
{1, 2, 3}
{8, 6, 7}
```

# Операции над множествами

## 4) Симметричная разница множеств



```
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

print(A ^ B)
print(A.symmetric_difference(B))
print(B.symmetric_difference(A))
```

```
{1, 2, 3, 6, 7, 8}
{1, 2, 3, 6, 7, 8}
{1, 2, 3, 6, 7, 8}
```

## 5) Проверка принадлежности к множеству

```
1 # инициализируем my_set
2 my_set = set("Python!!")
3
4 print(my_set)
5
6 print('a' in my_set)
7 print('p' not in my_set)
```

```
{'!', 'n', 'o', 'y', 'P', 't', 'h'}
False
True
```

## 6) Итерация множества

```
for letter in set("apple"):
    print(letter)
```

```
a
p
e
l
```



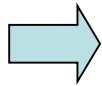
## Некоторые методы множеств:

**add()** - добавляет элемент во множество;

**clear()** - удаляет все элементы из множества;

**remove()** - удаляет элемент из набора. Если элемент не является членом множества, выдает `KeyError`.

```
1 s = {1,2,3}
2 print(s)
3
4 s.add(4)
5 print(s)
6
7 s.remove(2)
8 print(s)
9
10 s.clear()
11 print(s)
```



```
{1, 2, 3}
{1, 2, 3, 4}
{1, 3, 4}
set()
```

## Некоторые встроенные функции для работы с множествами:

**len()** - возвращает длину (количество элементов) множества;

**max()** - возвращает наибольший элемент во множестве;

**sum()** - возвращает сумму всех элементов множества.

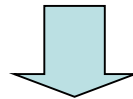
```
s = {1,2,3}
print(len(s))
print(max(s))
print(sum(s))
```



```
3
3
6
```

## НЕИЗМЕНЯЕМОЕ МНОЖЕСТВО (FrozenSet) – неизменяемое множество

```
1  # инициализируем A и B
2  A = frozenset([1, 2, 3, 4])
3  B = frozenset([3, 4, 5, 6])
4  C = A | B
5
6  print(C)
7  print(A.union(B))
8  print(A)
```



```
frozenset({1, 2, 3, 4, 5, 6})
frozenset({1, 2, 3, 4, 5, 6})
frozenset({1, 2, 3, 4})
```

## СЛОВАРИ (dict) - неупорядоченные наборы пар ключ-значение

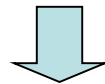
**Пример** создания словаря: `d = {1:'value1', 2:'value2', 3:'value3'}`

```
d = {1:'value1', 2:'value2', 3:'value3'}

print("d[1] =", d[1]);
print("d[3] =", d[3]);
print();

d1 = {1:'value', 'key':2}

print("d1['key'] =", d1['key']);
```

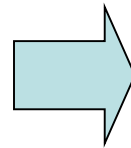


```
d[1] = value1
d[3] = value3

d1['key'] = 2
```

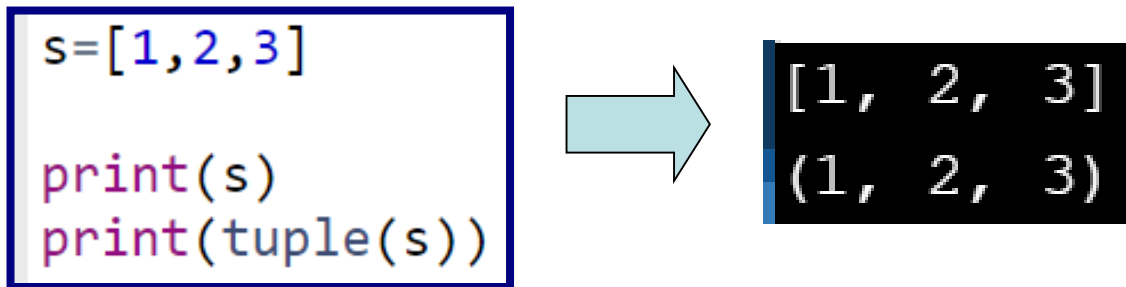
# 4 Преобразование типов

```
1 a = int(390.8)
2 print(int(390.8) =',a)
3
4 a = float(57)
5 print('float(57) =',a)
6
7 a = float('+12.345')
8 print("float('+12.345') =",a)
9
10 #преобразование с помощью деления
11 a = 5 / 2
12 print('5 / 2 =',a)
13
14 a = str(12)
15 print('str(12) =',a)
16
17 #конкатенация
18 print(a + '-' + str(10) + "=" + str(2))
19
20 a='3.33'
21 b="4.44"
22 print(a+b)
23 print(float(a)+float(b))
```

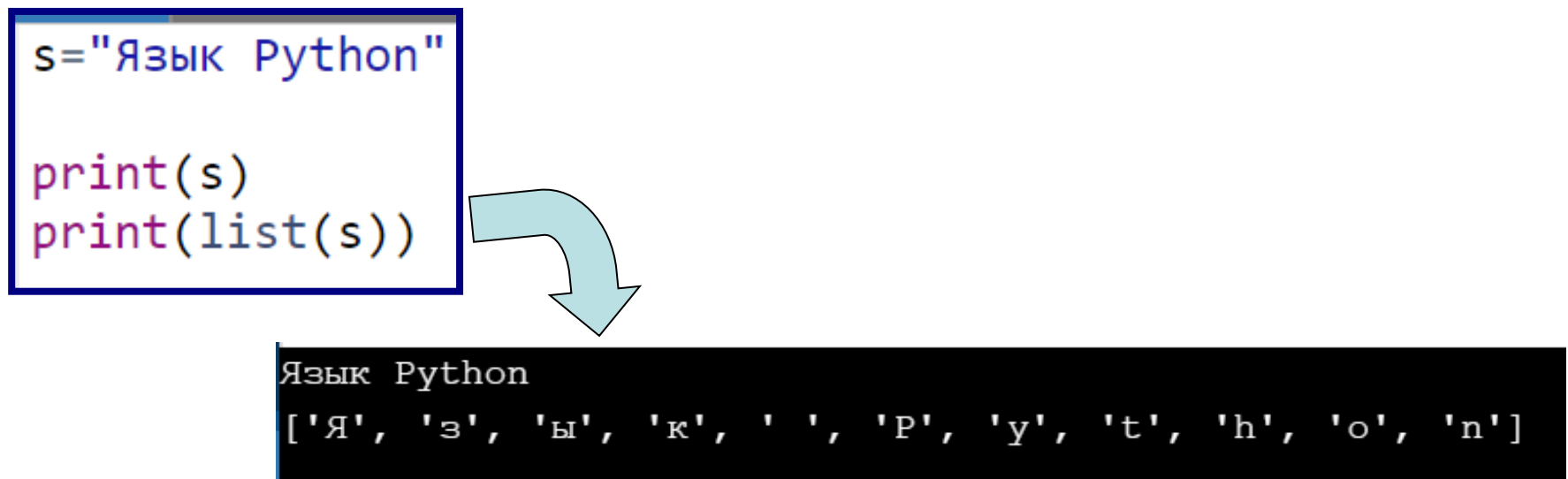


```
int(390.8) = 390
float(57) = 57.0
float('+12.345') = 12.345
5 / 2 = 2.5
str(12) = 12
12-10=2
3.334.44
7.77000000000000005
```

## Преобразование в кортеж: **tuple(...)**



## Преобразование в список: **list(...)**

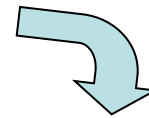


# 5 ФУНКЦИИ

def

lambda

```
def cena(rub, kop=0):  
    return "%i руб. %i коп." % (rub, kop)  
  
print (cena(8, 50))  
print (cena(7))  
print (cena(rub=23, kop=70))
```

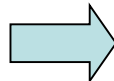


```
8 руб. 50 коп.  
7 руб. 0 коп.  
23 руб. 70 коп.
```

# Переменное количество аргументов функции

`*args` - для неименованных аргументов;  
`**kwargs` - для именованных аргументов.

```
def adder(*nums):  
    sum = 0  
    for n in nums:  
        sum += n  
  
    print("Sum: ", sum)  
  
adder(3, 5)  
adder(4, 5, 6, 7)  
adder(1, 2, 3, 5, 6)
```



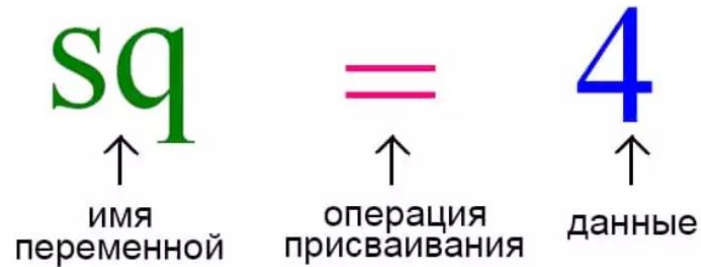
```
Sum: 8  
Sum: 22  
Sum: 17  
>>> |
```

```
Firstname is Sita  
Age is 22  
  
Firstname is Sita  
Lastname is Sharma  
Age is 22  
Phone is 1234567890  
>>>
```

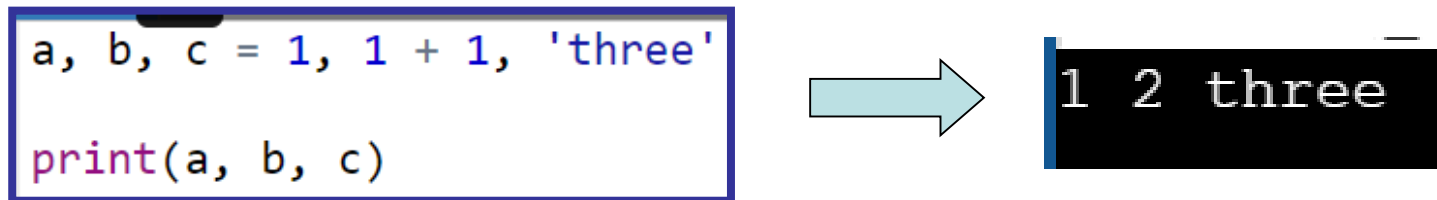


```
def intro(**data):  
    for key, value in data.items():  
        print("{} is {}".format(key, value))  
  
intro(Firstname="Sita", Age=22)  
print()  
intro(Firstname="Sita", Lastname="Sharma", Age=22, Phone=1234567890)
```

# 6 Переменные



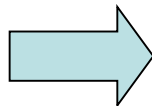
Указать значений для нескольких переменных за одну операцию:





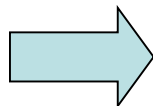
## Локальные и глобальные переменные:

```
1 a = 1
2
3 def f():
4     a = 2
5     print(a)
6
7 f()
8 print(a)
```



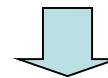
2  
1

```
1 a = 1
2
3 def f():
4     global a
5     a = 2
6     print(a)
7
8 f()
9 print(a)
```



2  
2

```
1 a = 1
2 def f1():
3     a = 2
4     def f2():
5         nonlocal a
6         a = 3
7         print(a) # ВЫВОДИТ 3
8     f2()
9     print(a+2) # ВЫВОДИТ 5
10
11 f1()
12 print(a) # ВЫВОДИТ 1
```



3  
5  
1

# 7 Управляющие конструкции

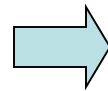
## Конструкция *if*

```
if выражение:  
    инструкция_1  
    инструкция_2  
    ...  
    инструкция_n
```

**ВАЖНО:** блок кода, который необходимо выполнить, в случае истинности выражения, отделяется пробелами слева!

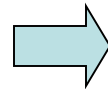
## Примеры

```
if 1:  
    print("hello 1")
```



hello 1

```
a = 3  
if a == 3:  
    print("hello 2")
```



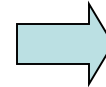
hello 2

## Конструкция *if – else*

```
if выражение:  
    инструкция_1  
    инструкция_2  
    ...  
    инструкция_n  
else:  
    инструкция_a  
    инструкция_b  
    ...  
    инструкция_x
```

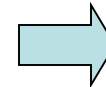
### Примеры

```
a = 3  
if a > 2:  
    print("H")  
else:  
    print("L")
```



H

```
a = 1  
if a > 2:  
    print("H")  
else:  
    print("L")
```



L

## Конструкция *if – elif – else*

```
if выражение_1:  
    инструкции_(блок_1)  
elif выражение_2:  
    инструкции_(блок_2)  
elif выражение_3:  
    инструкции_(блок_3)  
else:  
    инструкции_(блок_4)
```

```
a = int(input("введите число:"))  
if a < 0:  
    print("Neg")  
elif a == 0:  
    print("Zero")  
else:  
    print("Pos")
```

введите число:5  
Pos

введите число:-33  
Neg

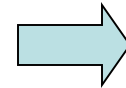
введите число:0  
Zero

# Оператор цикла *while*

```
while выражение:  
    инструкция_1  
    инструкция_2  
    ...  
    инструкция_n
```

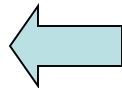
## Примеры

```
a = 0  
while a < 7:  
    print("A")  
    a += 1
```



A  
A  
A  
A  
A  
A  
A

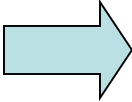
Бесконечный  
цикл



```
a = 0  
while a == 0:  
    print("A")
```

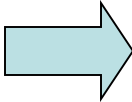
## Оператор цикла *for*

```
for i in range(5):  
    print("Hello")
```




```
Hello  
Hello  
Hello  
Hello  
Hello
```

```
lst = [1, 3, 5, 7, 9]  
for i in lst:  
    print(i * 2)
```



```
2  
6  
10  
14  
18
```

```
word_str = "Hello, world!"  
for l in word_str:  
    print(l)
```



```
H  
e  
l  
l  
o  
,  
  
w  
o  
r  
l  
d  
!
```

## Операторы *break* и *continue*

```
a = 0
while a >= 0:
    if a == 7:
        break
    a += 1
    print("A")
```



```
A
A
A
A
A
A
A
```

```
a = -1
while a < 10:
    a += 1
    if a >= 7:
        continue
    print("A")
```



```
A
A
A
A
A
A
A
```

# 8 Комментарии

```
# Обычный текст комментария
my_var = 'Hi, Mowshon!' # Описываем переменную

"""
Я тоже комментарии
расположенный на
несколько строк
"""

a = 3
```



# 9 Операции

## Арифметические операции

Оператор	Описание
+	Сложение - Суммирует значения слева и справа от оператора
-	Вычитание - Вычитает правый операнд из левого
*	Умножение - Перемножает операнды
/	Деление - Делит левый операнд на правый
%	Деление по модулю - Делит левый операнд на правый и возвращает остаток.
**	Возведение в степень - возводит левый операнд в степень правого
//	Целочисленное деление - Деление в котором возвращается только целая часть результата. Часть после запятой отбрасывается.

## Операции сравнения

Оператор	Описание
<b>==</b>	Проверяет равны ли оба операнда. Если да, то условие становится истинным.
<b>!=</b>	Проверяет равны ли оба операнда. Если нет, то условие становится истинным.
<b>&gt;</b>	Проверяет больше ли значение левого операнда, чем значение правого. Если да, то условие становится истинным.
<b>&lt;</b>	Проверяет меньше ли значение левого операнда, чем значение правого. Если да, то условие становится истинным.
<b>&gt;=</b>	Проверяет больше или равно значение левого операнда, чем значение правого. Если да, то условие становится истинным.
<b>&lt;=</b>	Проверяет меньше или равно значение левого операнда, чем значение правого. Если да, то условие становится истинным.

# Побитовые операции

Оператор	Описание
&	Бинарный "И" оператор, копирует бит в результат только если бит присутствует в обоих операндах.
	Бинарный "ИЛИ" оператор копирует бит, если тот присутствует в хотя бы в одном операнде.
^	Бинарный "Исключительное ИЛИ" оператор копирует бит только если бит присутствует в одном из операндов, но не в обоих сразу.
~	Бинарный комплиментарный оператор. Является унарным (то есть ему нужен только один операнд) меняет биты на обратные, там где была единица становится ноль и наоборот.
<<	Побитовый сдвиг влево. Значение левого операнда "сдвигается" влево на количество бит указанных в правом операнде.
>>	Побитовый сдвиг вправо. Значение левого операнда "сдвигается" вправо на количество бит указанных в правом операнде.

# Логические операторы

Оператор	Описание
<b>and</b>	Логический оператор "И". Условие будет истинным если оба операнда истина.
<b>or</b>	Логический оператор "ИЛИ". Если хотя бы один из операндов истинный, то и все выражение будет истинным.
<b>not</b>	Логический оператор "НЕ". Изменяет логическое значение операнда на противоположное.

## Операторы членства

проверки на наличие элемента в составных типах данных, таких, как строки, списки, кортежи или словари

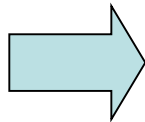
Оператор	Описание	Примеры
<b>in</b>	Возвращает истину, если элемент присутствует в последовательности, иначе возвращает ложь.	"cad" in "cadillac" вернет True. 1 in [2,3,1,6] вернет True. 2 in {"hi":2,"bye":1} вернет False (в словарях проверяется наличие в ключах, а не в значениях).
<b>not in</b>	Возвращает истину если элемента нет в	Результаты противоположны результатам оператора in

# Операторы тождественности

сравнение размещения двух объектов в памяти компьютера

Оператор	Описание	Примеры
<b>is</b>	Возвращает истину, если оба операнда указывают на один объект.	x is y вернет истину, если id(x) будет равно id(y).
<b>is not</b>	Возвращает ложь если оба операнда указывают на один объект.	x is not y, вернет истину если id(x) не равно id(y).

```
1 a = 5
2 b = a
3 print(a is b, "\n")
4
5 a, b = 1, 6
6 print(a is b)
7 print(a is 6)
8 print(a is 1)
9 print(a is not 1)
```



```
True
False
False
True
False
```

id(obj) – возвращает идентификатор объекта obj (идентификатор является адресом объекта в памяти )

# 10 Исключения

**try:**

k = 1 / 0

**except** ZeroDivisionError:

k = 0

print(k)

**try:**

k = 1 / 0

**finally:**

print("Выполнюсь в любом случае! ")

**try:**

res = int(open('a.txt').read()) / int(open('c.txt').read())

print(res)

**except** IOError:

print ("Ошибка ввода-вывода")

**except** ZeroDivisionError:

print ("Деление на 0")

**except** KeyboardInterrupt:

print ("Прерывание с клавиатуры")

**except:**

print ("Ошибка")