

Условный оператор. Конструкция if

if *a* логический_оператор *b* :

выражения, выполняемые при результате True
в логическом выражении

Пример:

if numbig < 100: # если значение переменной numbig меньше 100,
то ...

*c = a**b* # возвести значение переменной *a* в степень *b*,
результат присвоить переменной *c*.

Схема программы
с конструкцией **if**



Условный оператор. Конструкция if с условием else

if *a* логический_оператор *b* :

выражения, выполняемые при результате True в логическом выражении

else :

выражения, выполняемые при результате False

в

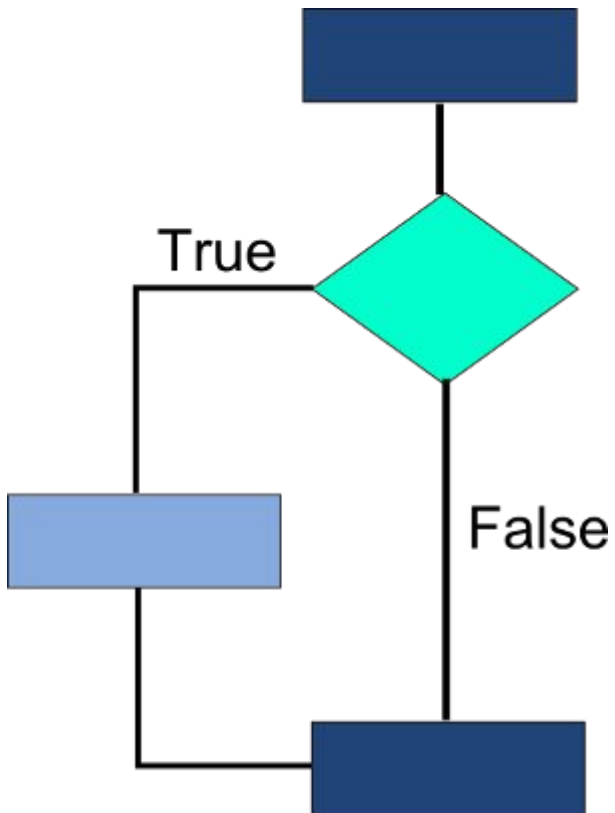
логическом выражении

Пример:

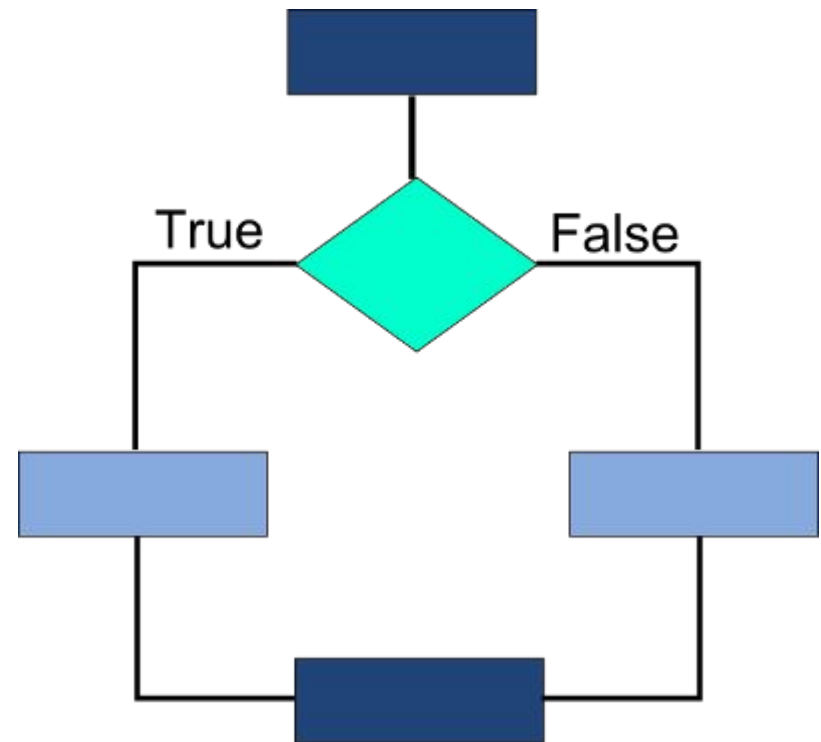
```
print ( "Привет " )
товар1 = 50
товар2 = 32
if товар1+ товар2 > 99 :
    print ("Сумма не достаточна ")
else:
    print ("Чек оплачен ")
print ("Пока ")
```

Условный оператор. Блок-схемы

Конструкция – **if**



Конструкция **if - else**



Условный оператор. Конструкция if с условием elif

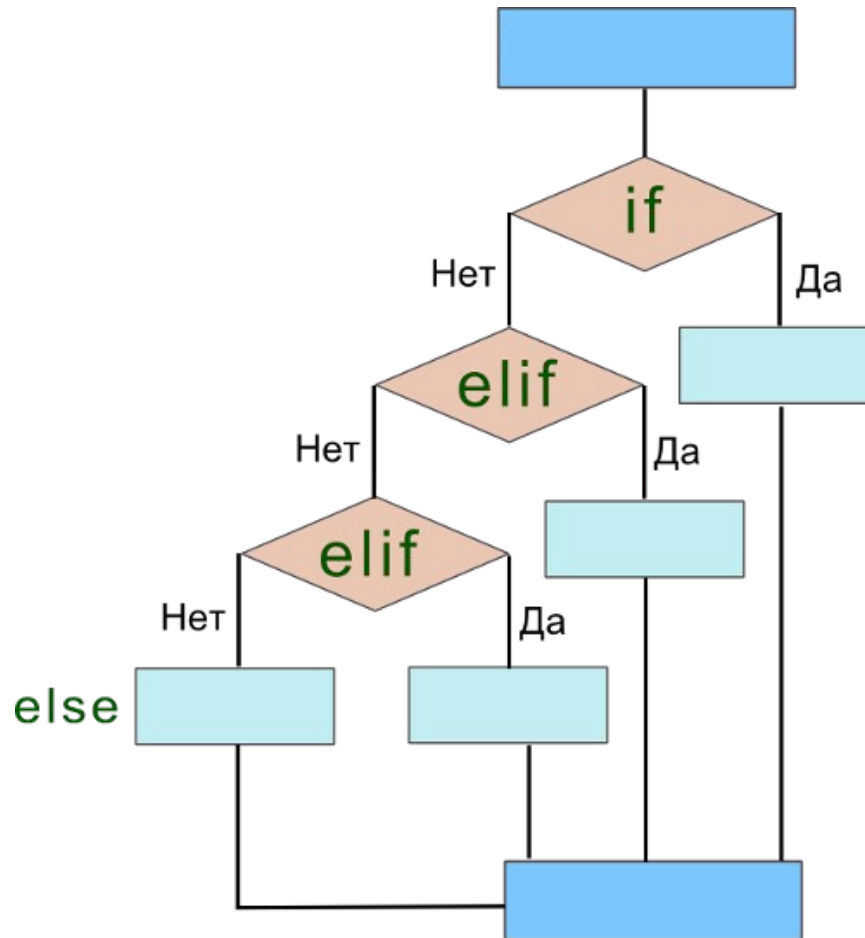
Примеры скриптов:

```
x = -10
if x > 0:
    print (1)
elif x < 0:
    print (-1)
else:
    print (0)
```

```
result = "no result"
num1 = 3
if num1 == 0:
    result = 0
elif num1==1:
    result = 1
elif num1==2:
    result = 2
elif num1==3:
    result = 3
elif num1==4:
    result = 4
elif num1==5:
    result = 5
else:
    print ("Error")
print (result)
```

Условный оператор. Конструкция if с условием elif

Блок-схема **if -elif - else**



Задание 1(а). (Исходный код)

Линейная программа

```
a = int(input("Введите a = "))  
b = int(input("Введите b = "))  
k = int(input("Введите k = "))  
m = int(input("Введите m = "))
```

```
from math import *
```

```
if k-m == 0 or a*b*k == 0:
```

```
    print("\nЗнаменатель равен НУЛЮ!!! Введите другие  
значения.")
```

```
else:
```

```
    C = sqrt((a-b)**2/abs(k-m))
```

```
    print("C = ",C)
```

```
    A = sin(pi/6)*C**2-C*(a-b)/(a*b*k)
```

```
    print("A = ",A)
```

```
input("\n\nНажмите Enter чтобы выйти.")
```

Цикл while

Циклы — это инструкции, выполняющие одну и ту же последовательность действий, пока действует заданное условие.

while **a** логический_оператор **b**:

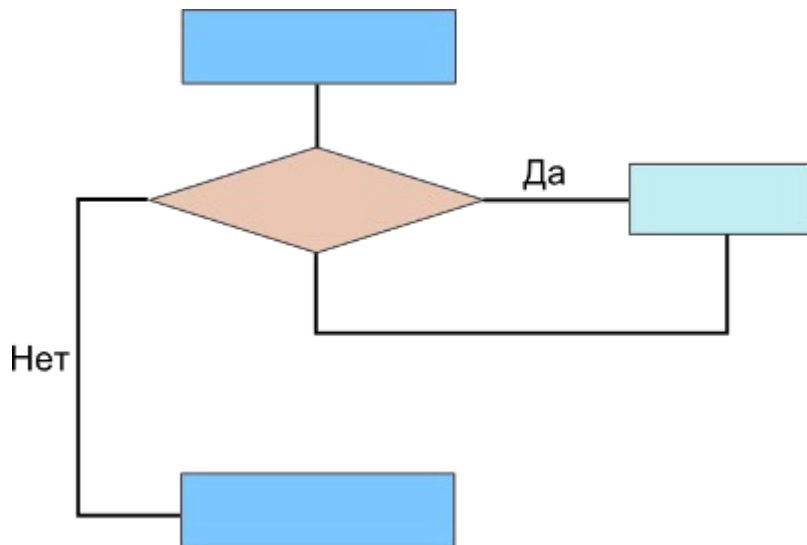
действие(я)

изменение **a**

```
Пример: i = 1
while i <= 10:
    print(i)
    i += 1    # i = i + 1
```

Цикл while

Блок-схема цикла while



Пример:

```
fib1 = 0
```

```
fib2 = 1
```

```
print (fib1)
```

```
print (fib2)
```

```
n = 10
```

```
i = 0
```

```
while i < n:
```

```
    fib_sum = fib1 + fib2
```

```
    print (fib_sum)
```

```
    fib1 = fib2
```

```
    fib2 = fib_sum
```

```
    i = i + 1
```


Цикл `while`.

Инструкции управления циклом

`break` – немедленное прекращение цикла

`continue` – продолжение цикла

Пример:

```
count = 0
```

```
while True:
```

```
    count += 1
```

```
    if count > 10: # завершить цикл, если count
```

```
        break # принимает значение больше 10
```

```
    if count == 5: # пропустить 5
```

```
        continue
```

```
    print (count)
```

```
input("\n\nНажмите Enter, чтобы выйти.")
```

Цикл for

Циклы **for** используются:

- для повторения какой-либо последовательности действий заданное число раз (совместно с функцией **range**)

```
for i in range(n): # n != 0, n > 0
    Тело цикла
```

Пример: `for i in range(10):`
`print(i)`

Вывод:

0
1
2
3
4
5
6
7
8

Цикл for

- либо для изменения значения переменной в цикле от некоторого начального значения до некоторого конечного

а) `for i in range(a, b):` # переменная `i` будет принимать значения от `a` до `b - 1`, `a <= b`
Тело цикла

б) `for i in range(a, b, c):` # переменная `i` будет принимать значения от `a` до `b - 1`,
Тело цикла `c` – шаг индексной переменной

- используется для обхода заданного множества элементов (символов строки, объектов списка или словаря)

Цикл for.

Примеры использования цикла for

1. # Вывод 'Hello' 5 раз и
'There' один раз

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

2. # Вывод 'Hello', 'There'
5 раз

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

3. # Два способа вывода
цифр от 1 до 10

```
for i in range(1, 11):  
    print (i, end=" ")  
for i in range(10):  
    print (i+1)
```

4. # Два способа вывода
четных цифр от 2 до 10

```
for i in range(2, 12, 2):  
    print (i)  
for i in range(5):  
    print ((i+1)*2)
```

5. # Вывод цифр от 10 до 1

```
for i in range(10, 0, -1):  
    print(i)
```

6. # Что выводится?
Почему?

```
for i in range(3):  
    print ("a")  
    for j in range(3):  
        print ("b")
```

Строки (str)

Строка — это неизменяемая последовательность символов.

Строки могут заключаться как в одиночные ('Game Over'), так и двойные кавычки ("Game Over"). Однако, начало и конец строки должны обрамляться одинаковым типом кавычек.

Пример:

```
print("Программа 'Game Over' 2.0")
print("Программа", " 'Game Over' 2.0 ")
print("Программа",
      " 'Game Over' 2.0 ")
print("Программа", end=" ")
print("""Game Over' 2.0 """)
```

Вывод результата: Программа 'Game Over' 2.0
Программа 'Game Over' 2.0
Программа 'Game Over' 2.0
Программа 'Game Over' 2.0

Для вывода псевдографики используются тройные кавычки

Строки (str).

Escape-последовательности

| Последовательность | Описание |
|--------------------|---|
| \\ | Обратный слеш. Выводит один знак обратного слеша |
| \' | Апостроф, или одиночная кавычка. Выводит один апостроф |
| \" | Двойные кавычки. Выводит одну такую кавычку |
| \n | Пустая строка. Перемещает курсор в начало следующей строки |
| \t | Горизонтальный отступ – символ табуляции. Перемещает курсор вправо на один отступ |




Строки (str). Специальные функции

Функция **len()** определяет длину строки.

Для строк существуют операции **конкатенации** (+) и **дублирования** (*).

Оператор **in** определяет, является ли какой-либо символ элементом строки.

Пример:

| | | |
|---|--|--------------------------|
| <pre>>>> len ('It is a long string')</pre> |  | определение длины строки |
| <pre>19</pre> | | |
| <pre>>>> '!!!' + ' Hello World ' + '!!!'</pre> |  | конкатенация |
| <pre>'!!! Hello World !!!'</pre> | | |
| <pre>>>> '#' * 20</pre> |  | дублирование |
| <pre>'#####'</pre> | | |
| <pre>>>> if 'e' in ' Hello World ':</pre> | | |
| <pre> print ('встречается в тексте')</pre> | | |
| <pre>else:</pre> | | |
| <pre> print ('не встречается в тексте')</pre> | | |
| <pre>встречается в тексте</pre> | | |

Строки (str). Индексация

Индекс – уникальный порядковый номер символов в строке (а также в других структурах данных: списках, кортежах).

| | | | | | |
|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 |
| И | Н | Д | Е | К | С |
| -6 | -5 | -4 | -3 | -2 | -1 |

Примеры:

1.

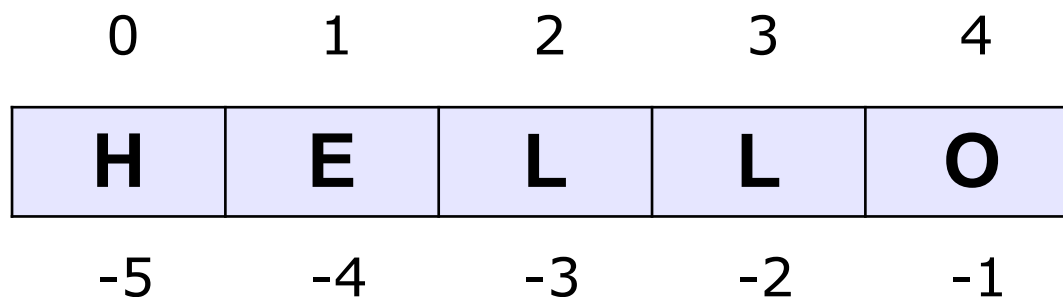
```
>>> 'индекс' [0]
'и'
>>> 'индекс' [-1]
'с'
>>> 'индекс' [-3]
'е'
>>>
```

2.

```
>>> tday = 'morning, afternoon, night'
>>> tday [4]
'i'
>>> a = tday[1]
>>> a
'o'
>>>
```


Строки (str). Срезы (slices)

Срезы (slices) – извлечение из данной строки одного символа или некоторого фрагмента (подстроки)



Оператор извлечения среза из строки выглядит так: [X:Y].
X – индекс **начала среза**,
Y – индекс **окончания среза** (символ с номером Y в срез не входит).

```
>>> s = 'hello'
>>> s[1:4]      ИЛИ      >>> s = 'hello'
'ell'           >>> s[-4:-1]
>>>             >>> 'ell'
```

Строки (str). Срезы (slices)

Если отсутствует первый индекс, то срез берется от начала до второго индекса

```
>>> a = 'very big string'  
>>> a[:4]  
'very'  
>>>
```

При отсутствии второго индекса, срез берется от первого индекса до конца строки

```
>>> a = 'very big string'  
>>> a[9:]  
'string'  
>>>
```

Если оператор извлечения среза из строки выглядит так: [X:Y:Z], **Z** – шаг, через который выбирают элементы

```
>>> a = 'very big string'  
>>> a[::3]  
'vyisi'  
>>>
```

Строки (str).

Строковые методы

Метод - это функция, применяемая к объекту (в данном случае - к строке).

Вызов метода:

имя_объекта.имя_метода(параметры)

Метод `find` находит в данной строке данную подстроку (которая передается в качестве параметра).

Функция возвращает индекс первого вхождения искомой подстроки. Если же подстрока не найдена, то метод возвращает значение -1.

Например: `>>> s = 'Hello'`

```
>>> print(s.find('e'))
```

```
1
```

```
>>> print(s.find('ll'))
```

```
2
```

```
>>> print(s.find('L'))
```

```
-1
```

Строки (str).

Строковые методы

Метод `rfind` возвращает индекс последнего вхождения данной строки (“поиск справа”).

Если вызвать метод `find` с тремя параметрами `s.find(T, a, b)`, то поиск будет осуществляться в срезе `s[a:b]`. Если указать только два параметра `s.find(T, a)`, то поиск будет осуществляться в срезе `S[a:]`, то есть начиная с символа с индексом `a` и до конца строки.

Например:

```
1. >>> s = 'Hello'
>>> print(s.find('l'))
2          2
>>> print(s.rfind('l'))
3          -1
>>>
```

```
2. >>> s='Hello'
>>> print(s.find('l',1,4))
>>> print(s.find('H',1))
```

Строки (str).

Строковые методы

Метод `replace` – `s.replace(old, new)` — заменяет в строке `s` все вхождения подстроки `old` на подстроку `new`.

Пример: `>>> 'Hello'.replace('l', 'L')`
`'HeLlo'`

Формат: `s.replace(old, new, count)` – заменены будут не все вхождения, а только не больше, чем первые `count` из них.

Пример: `>>> 'Abrakadabra'.replace('a', 'A', 2)`
`'AbrAkAdabra'`

Метод `count` `s.count(T)` возвращает число вхождений строки `T` внутри строки `S`.

Пример: `>>> 'Abracadabra'.count('a')`
`4`

При указании трех параметров `s.count(T, a, b)`, будет выполнен подсчет числа вхождений строки `T` в срез `S[a:b]`.

Строки (str).

Строковые методы

| Метод | Описание |
|---------------------|---|
| upper() | Возвращает строку, символы которой приведены к верхнему регистру |
| lower() | Возвращает строку, символы которой приведены к нижнему регистру |
| swapcase() | Возвращает новую строку, в которой регистр всех символов обращен: верхний становится нижним и наоборот |
| capitalize() | Возвращает новую строку, в которой первая буква прописная, а остальные – строчные |
| title() | Возвращает новую строку, в которой первая буква каждого слова прописная, а остальные – строчные |
| strip() | Возвращает строку, из которой убраны все интервалы(табуляция, пробелы, символы пустых строк) в начале и в конце |

Задание 2. (Исходный код)

Введите текст. Создайте из введенного текста новый текст без гласных и выведите его на печать.

Создание новых строк из исходных с помощью for-цикла

```
message = input("Введите текст: ")
```

```
new_message = ""
```

```
glasn = "aeiouaёеиоуыэюя"
```

```
print ()
```

```
for letter in message:
```

```
    if letter.lower() not in glasn:
```

```
        new_message += letter
```

```
        print ("Создана новая строка:", new_message)
```

```
print ("\nНовый текст с изъятými гласными буквами:", new_message)
```

```
input ("\n\nНажмите Enter, чтобы выйти.")
```