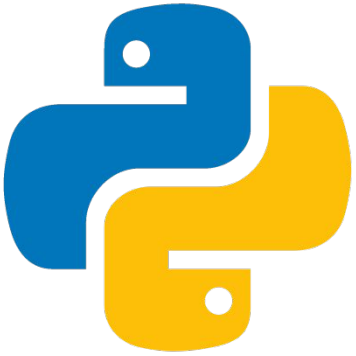


ТЕМА 3: РАБОТА С УСЛОВНЫМИ ОПЕРАТОРАМИ И ОПЕРАТОРАМИ ЦИКЛОВ

Операторы условия в «Python»

1) оператор *if* – определяет условие как истинное или ложное, и если условие истинно, программа выполняет соответствующее действие:

```
>>> balance=-5
>>> if balance<0:
        print ("нет возможности осуществить звонок")
нет возможности осуществить звонок
```

2) оператор *else* – позволяет выполнять определенное действие даже тогда, когда выражение *if* является ложным:

```
>>> balance=5
>>> if balance<0:
        print ("нет возможности сделать звонок")
else: print ("Вы сможете сделать звонок")
Вы сможете сделать звонок
```

3) оператор *elif* – применяется, когда программа должна обрабатывать более двух возможных результатов при этом количество дополнительных условий неограниченно, чего операторы *if* и *else* обеспечить не могут:

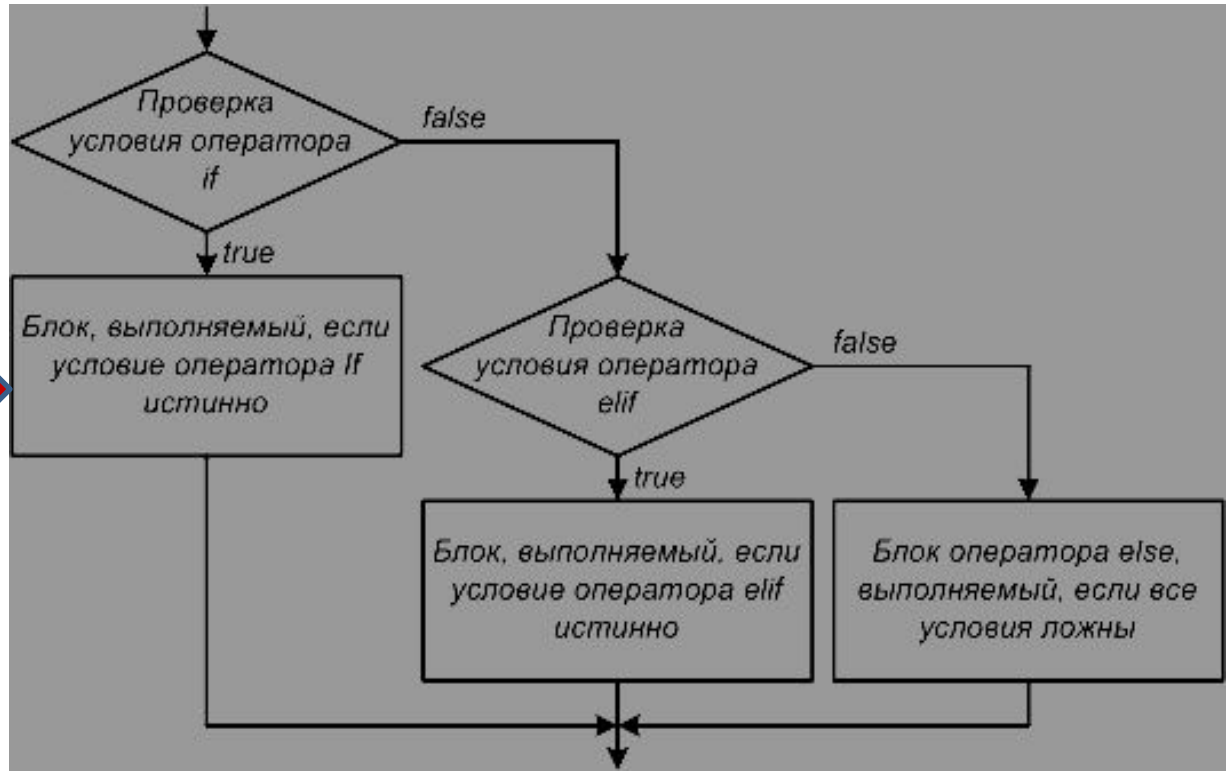
```
>>> balance=5
>>> if balance<0:
        print ("нет возможности сделать звонок")
elif balance==0:
        print ("нет возможности сделать звонок")
else: print ("Вы сможете сделать звонок")

Вы сможете сделать звонок
```

Интегрированная совокупность условных операторов в «Python»

```
if <логическое выражение>:
        <блок, выполняемый, если условие истинно>
elif < логическое выражение>:
        < блок, выполняемый, если условие истинно>
……..
else: < блок, выполняемый, если все условия ложны>:
```

Схема алгоритма работы условных операторов



Оператор цикла for в «Python»

Оператор цикла ***for*** применяют в тех случаях, когда, существует необходимость повторить что-нибудь определенное количество раз

Цикл *for* имеет следующую структуру

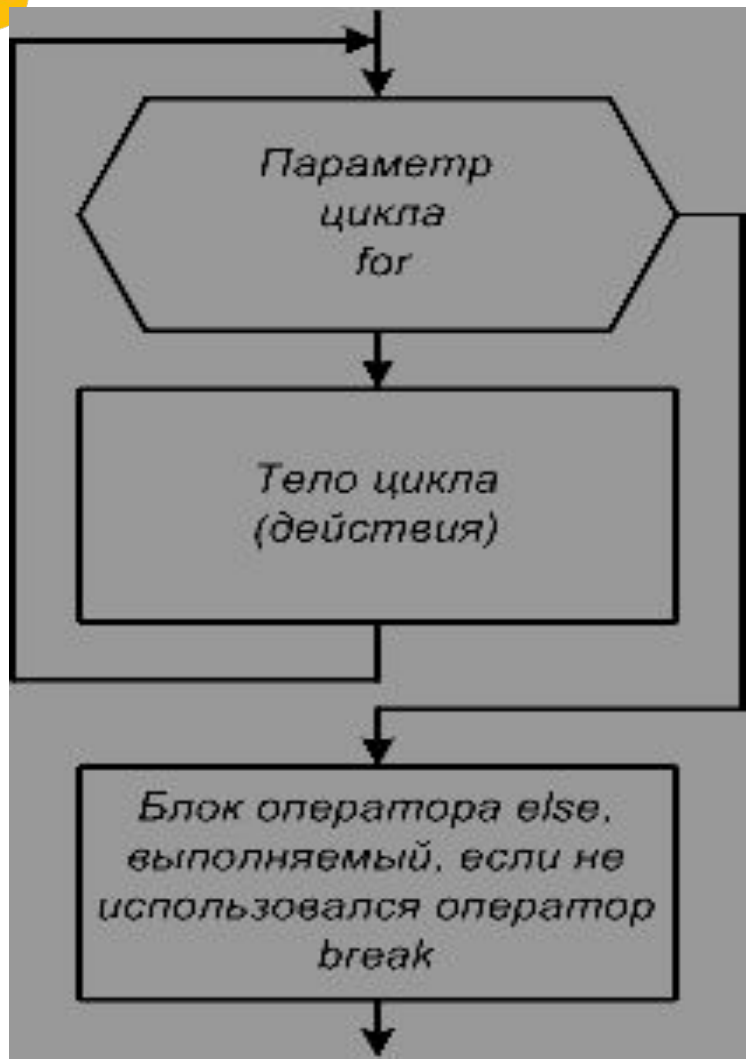
```
for <текущий элемент> in <последовательность>:
        <инструкции внутри цикла>
else:
        <выполняемый блок, если не использовался оператор
break>
```

1)<текущий элемент> – на каждой итерации через параметр доступен текущий элемент последовательности или ключ словаря;

2) *<последовательность>* – объект, поддерживающий механизм итерации (строка, список, кортеж, диапазон, словарь и т.д.);

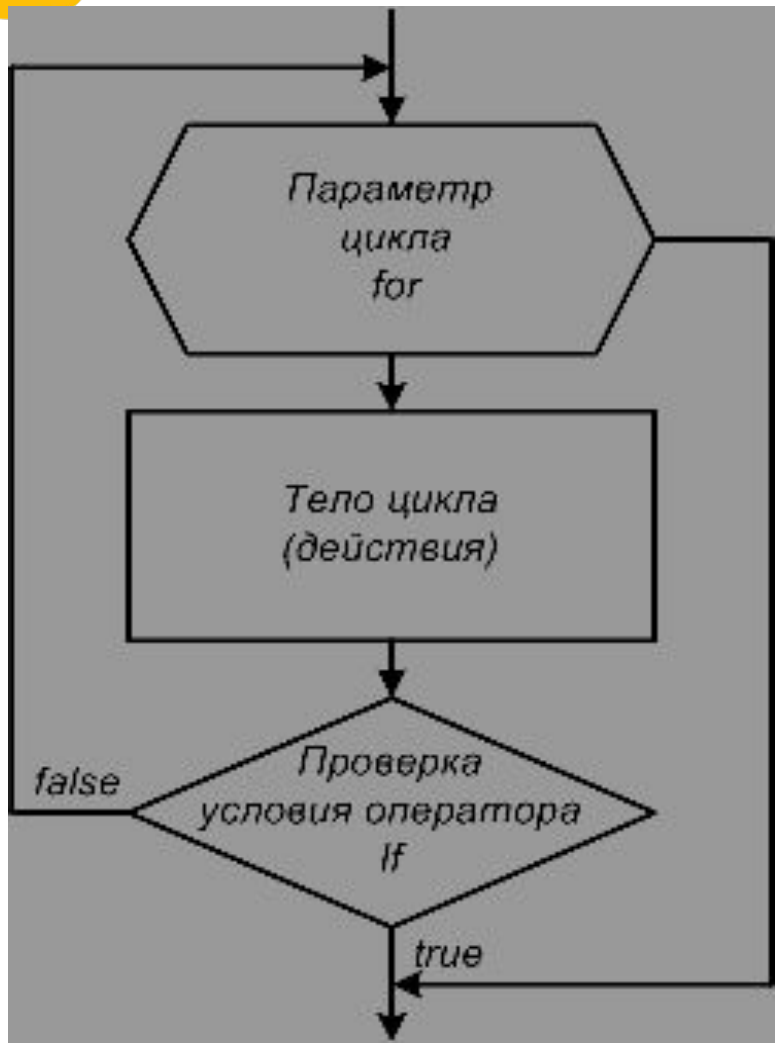
3) *<инструкции внутри цикла>* – блок, выполнение которого будем многократным.

Схема алгоритма работы цикла for без прерывания в «Python»



Если внутри цикла не использовался оператор ***break***, то после завершения выполнения цикла будет выполнен блок в инструкции ***else***, который является необязательным.

Схема алгоритма работы цикла for с прерыванием в «Python»



В случае применения оператора ***break*** цикл, выполняемый в программе, будет прерван, как только выполнится условие, указанное в операторе ***if***.

Оператор цикла for позволяет организовать перебор:

1) букв в слове:

```
>>> for i in "BO":
        print(i, end=" ")
else:
        print("цикл выполнен")
B O цикл выполнен
```

2) списка кортежа:

```
>>> for i in [2, 8]:
        print(i)
2
8
```

3) элементов словаря:

```
>>> arr={"a":1, "b":2}
>>> arr.keys()
dict_keys(['a', 'b'])
>>> for key in arr.keys():
        print(key, arr[key])
a 1
b 2
```

```
>>> arr={"a":1, "b":2}
>>> for key in arr:
        print(key, arr[key])
a 1
b 2
```

метод ***keys()*** возвращает объект ***dict_keys()***, содержащий все ключи словаря

4) элементов списка кортежей:

```
>>> arr=[(5, 8), (24, 56)]
>>> for x, y in arr:
        print (x, y)
5 8
24 56
```

Работа с циклом while в «Python»

Выполнение цикла ***while*** продолжается до тех пор, пока логическое выражение является истинным

Цикл *while* имеет следующую структуру

```
<начальное значение>
while <условие>:
        <инструкции>
        <приращение>
[else:
        <блок, выполняемый, если не использовался оператор break>]
```

Схема алгоритма работы цикла while без прерывания

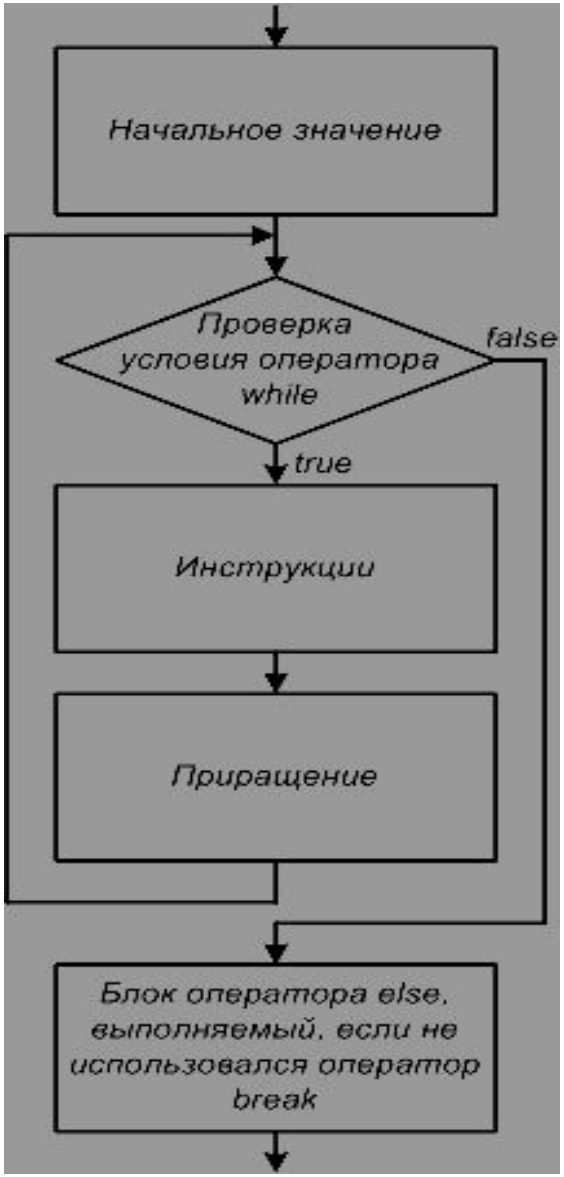
1) переменной (счетчику) присваивается начальное значение;

2) выполняется проверка условия и, если условие истинно, то выполняются инструкции внутри цикла, в противном случае выполнение цикла завершается;

3) переменная (счетчик) изменяется на величину, указанную в параметре <*приращение*>;

4) выполняется переход к пункту 2;

5) если внутри цикла не использовался оператор ***break***, то после завершения выполнения цикла будет выполнен блок в инструкции ***else***, однако следует сказать, что этот блок не является обязательным.



Пример работы с циклом while

Программный код, позволяющий выводить все числа от 1 до 3 с применением цикла ***while***:

```
>>> i=1 # Начальное значение
>>> while i<4: # Условие
        print(i) # Инструкции
        i+=1 # Приращение


1
2
3
```

В случае, если приращение не указано, то цикл будет бесконечным. Прерывание бесконечного цикла осуществляется нажатием клавиш ***<Ctrl>+<C>***

Применение оператора continue в «Python»

Осуществить переход к следующей итерации цикла до завершения выполнения всех инструкций внутри цикла позволяет оператор ***continue***

Рассмотрим пример вывода всех чисел от 1 до 10, за исключением чисел от 3 до 8 включительно:

```
>>> for i in range(1, 11):
        if 2<i<9:
            continue
        print(i)
1
2
9
10
```

Работа с оператором break в «Python»

Выполнять прерывание цикла досрочно позволяет оператор ***break***

Следует помнить о том, что оператор ***break*** прерывает выполнение цикла, а не программы, это говорит том, что инструкция, следующая за циклом, будет выполнена.

```
>>> i=1
>>> while True:
        if i>3: break # Прерывание цикла
        print(i)
        i+=1


1
2
3
```

В качестве примера рассмотрим вывод чисел от 1 до 3:

Поскольку в условии указано значение True, то выражения внутри цикла будут выполняться бесконечно, однако применение оператора break прерывает выполнение цикла, как только число строк достигнет 3.