

# Python

## Циклы

### While. For

# Повторение

---

1. Что такое условие?
2. Что такое условный оператор?
3. Для чего используется переменная?
4. Как выглядит оператор сравнения?
5. Какая команда вызывает диалоговое окно?

# Циклы

- Циклы **for** и **while** позволяют выполнить одно и то же действие несколько раз подряд.
- **for** используется если известно количество повторений
- **while** используется в случаях, если количество повторений цикла заранее неизвестно.

# Циклы While и For

While

```
while <condition>:  
    <action>
```

For

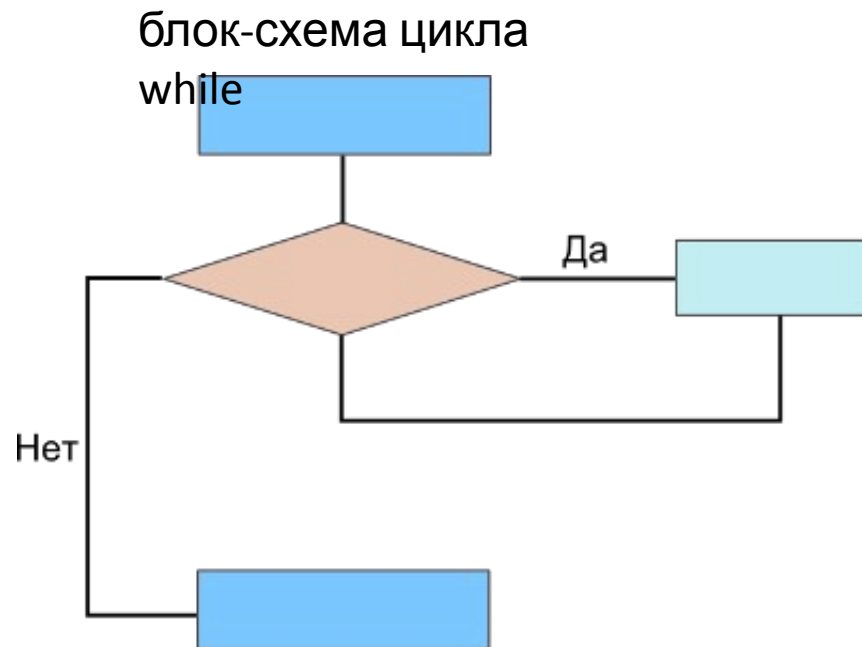
```
for <item> in <sequence>:  
    <action>
```

# Цикл While

Инструкция `while` в Python повторяет указанный блок кода до тех пор, пока указанное в цикле условие будет оставаться истинным.

# Цикл while

Цикл **while**  
повторяет  
команды, пока  
верно условие.



**while** условие:



} заголовок

} тело цикла

WHILE – «пока» в переводе с английского

Русским языком:

Пока <выполняется условие>: делать какие-то действия.

Пример:

```
n=0
while n<3:
    n=n+1
```



4 пробела!

Чему будет равно n  
после завершения  
цикла?

Запустить программу  
на компьютере:

```
n=0
while n<5:
    n=n+1
    print(n)
```

## Запись в тетрадь!

WHILE – «пока» в переводе с английского

Общая форма записи:

```
while <условие>:  
    <действие 1>  
    <действие 2>  
    и т.д.
```

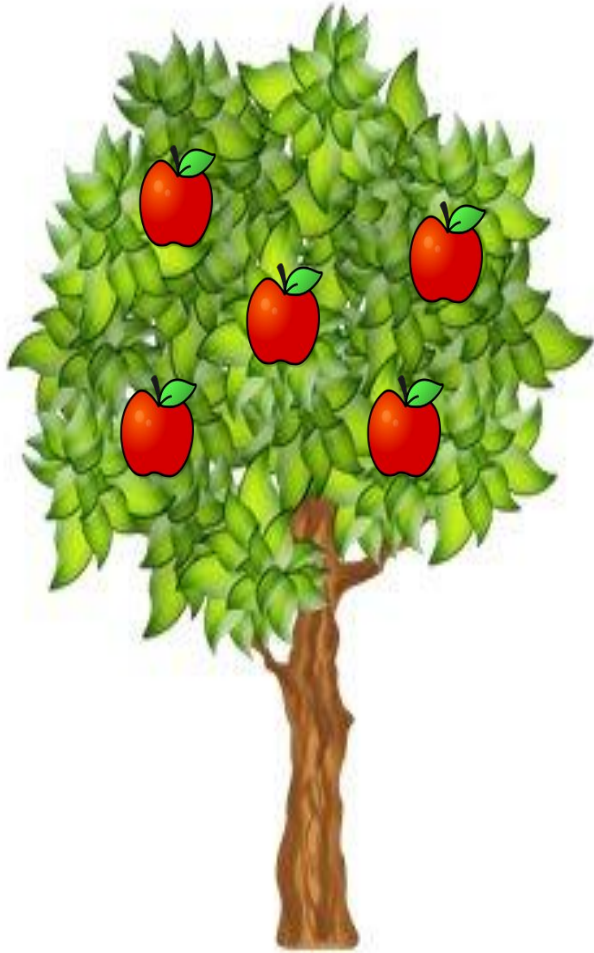
Пример:

```
n=0  
while n<5:  
    n=n+1  
    print(n)
```



# Сбор яблок

---



Сколько яблок нужно сорвать с дерева, чтобы собрать весь урожай?

# Сбор яблок

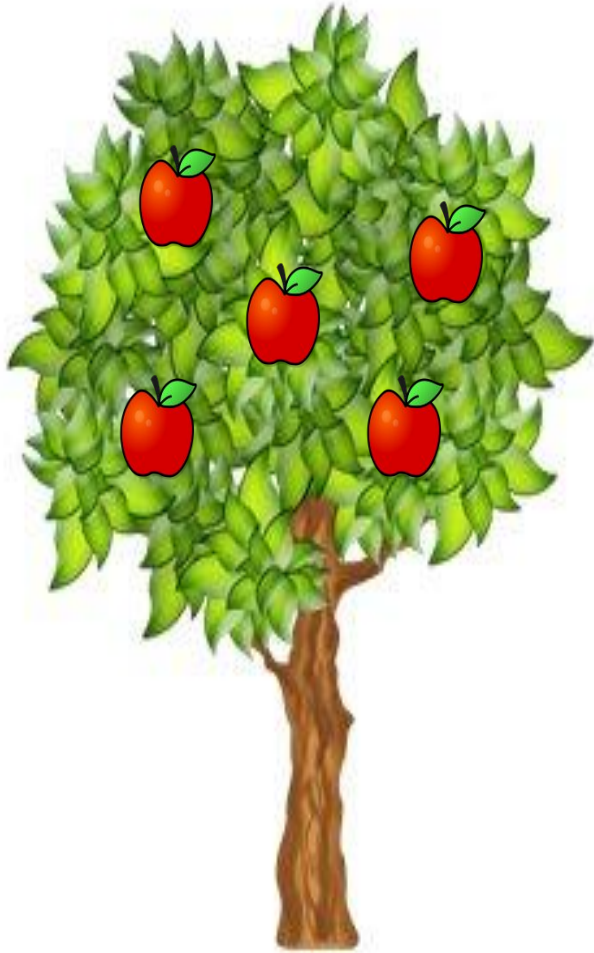
---



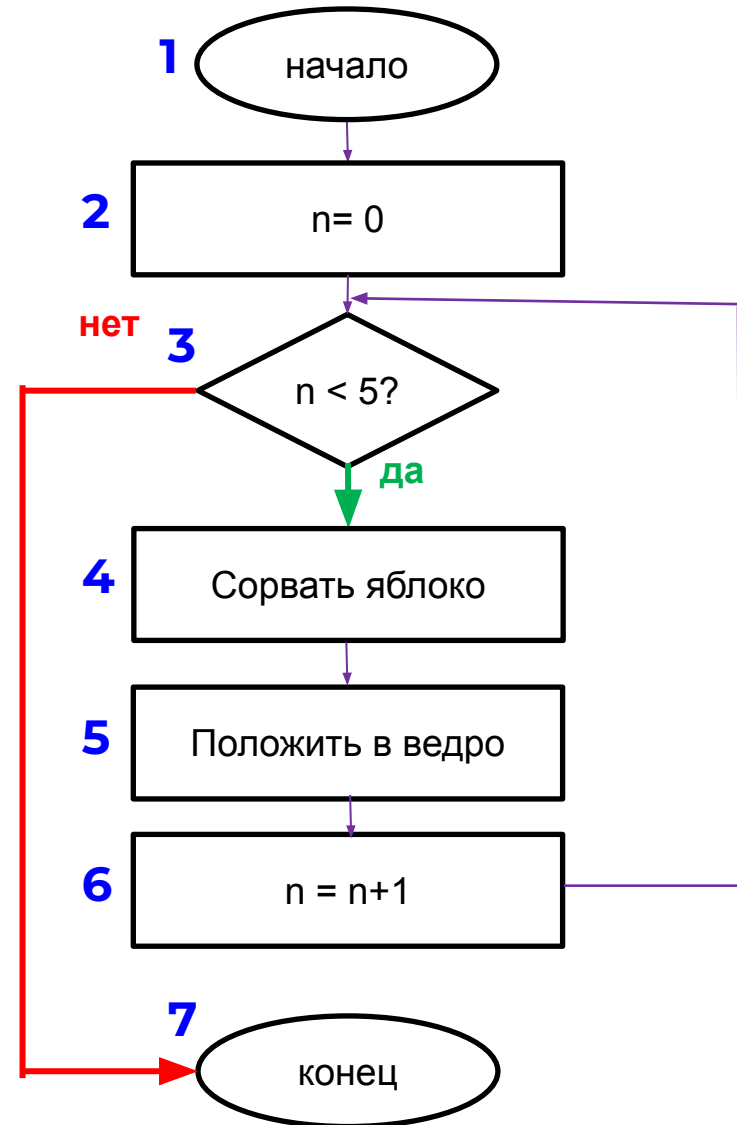
Сколько яблок нужно сорвать с дерева, чтобы собрать весь урожай?

5

# Алгоритм «Сбор яблок»



$n$  = кол-во собранных яблок



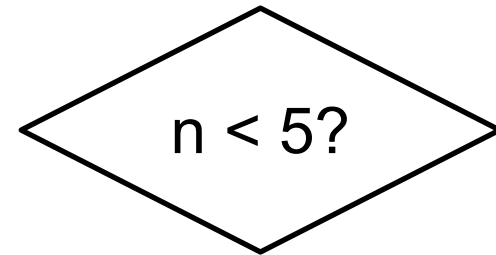
# Алгоритм «Сбор яблок»

---



**$n$  = кол-во собранных яблок**

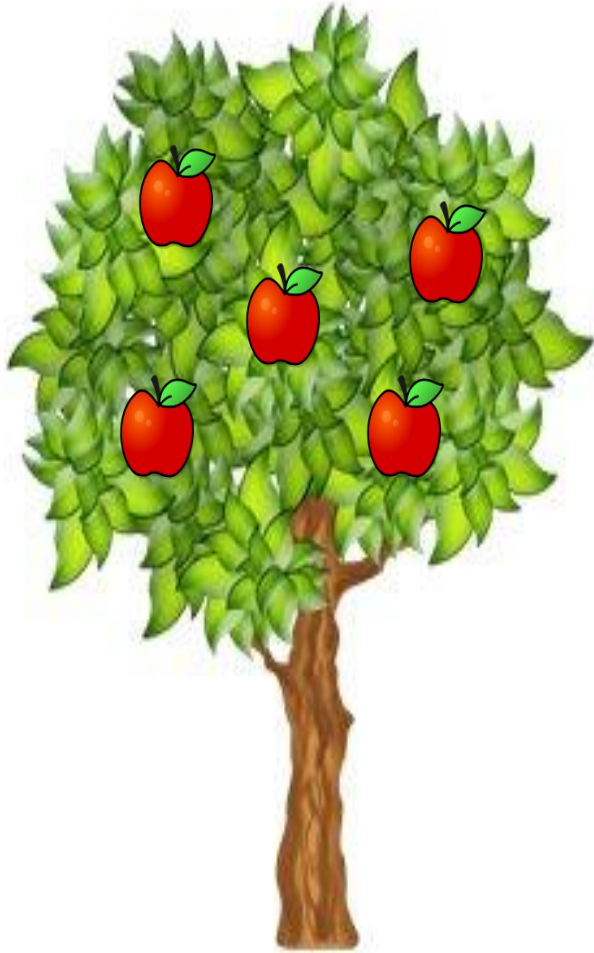
Условие, завершающее сбор  
яблок



Условие – выражение/вопрос,  
на которое можно однозначно  
ответить да или нет.

# Алгоритм «Сбор яблок»

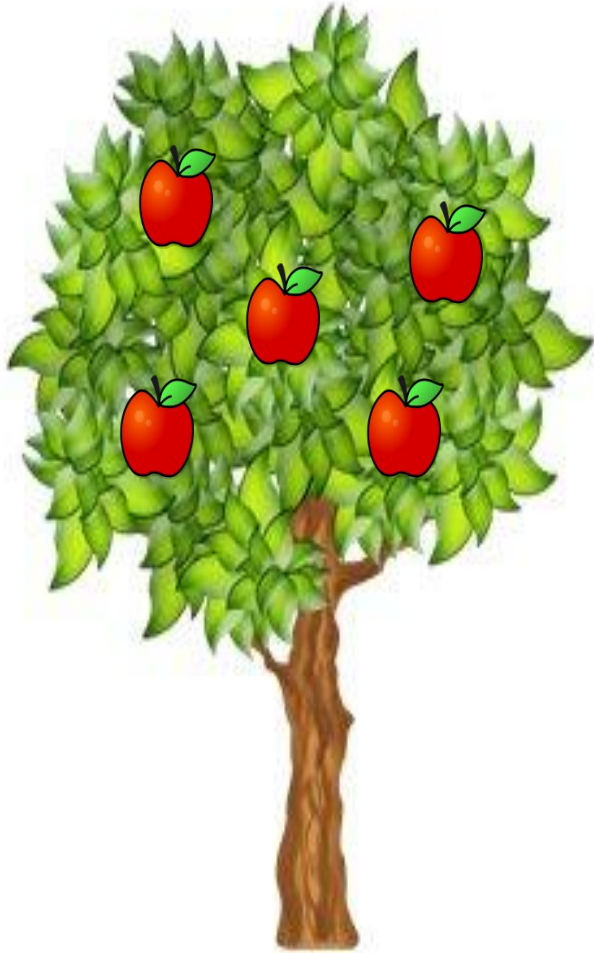
---



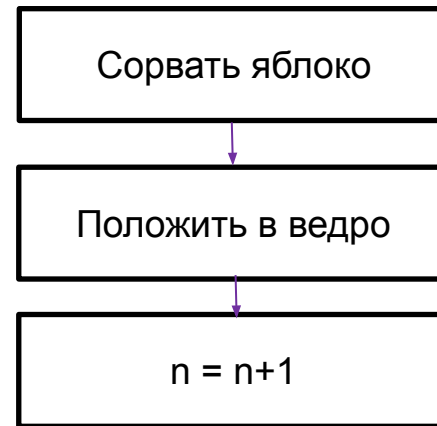
Какие команды мы выполняли,  
чтобы сорвать яблоко?

# Алгоритм «Сбор яблок»

---

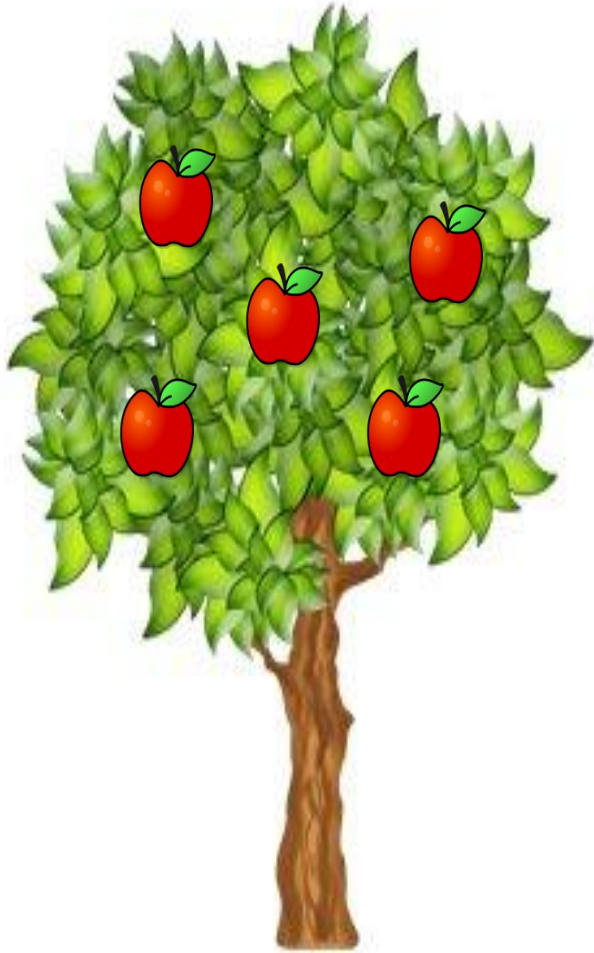


Какие команды мы выполняли, чтобы сорвать яблоко?

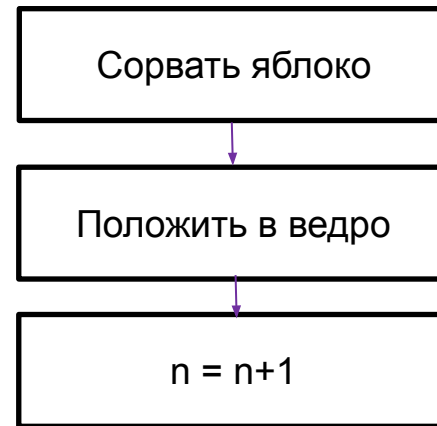


# Алгоритм «Сбор яблок»

---



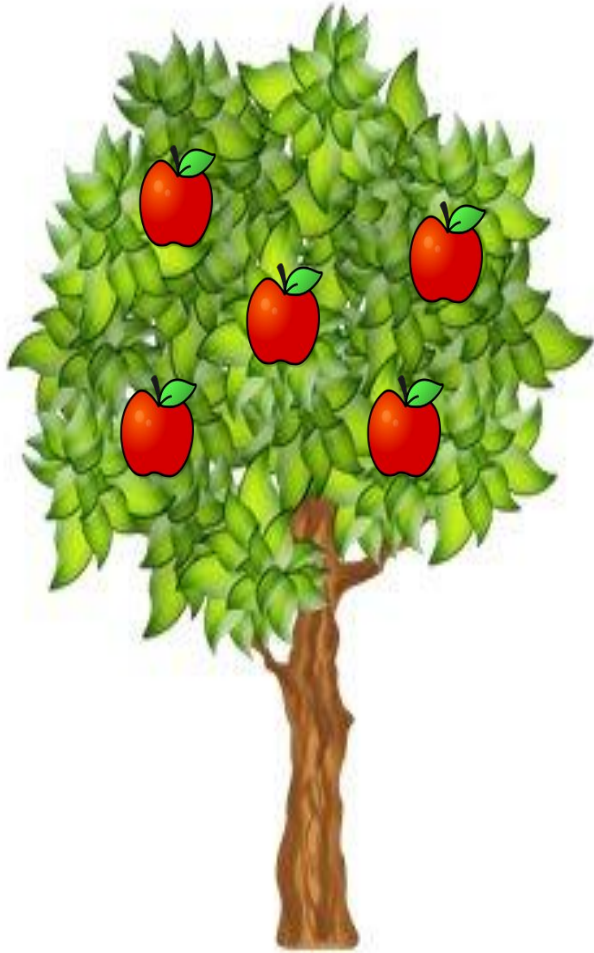
Сколько раз мы повторили следующие команды:



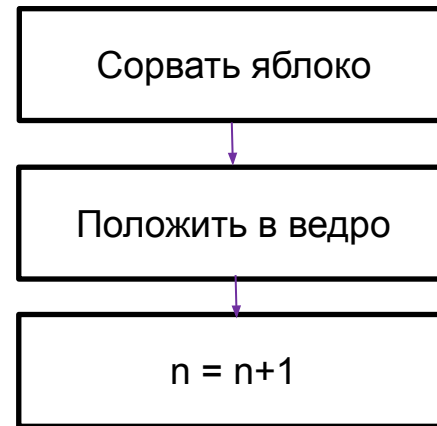


# Алгоритм «Сбор яблок»

---



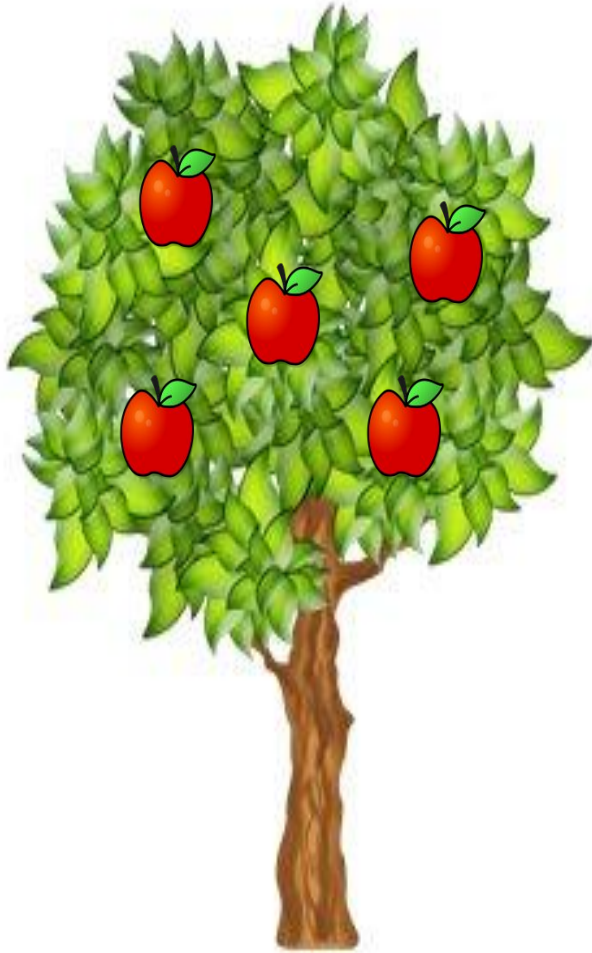
Сколько раз мы повторили следующие команды:



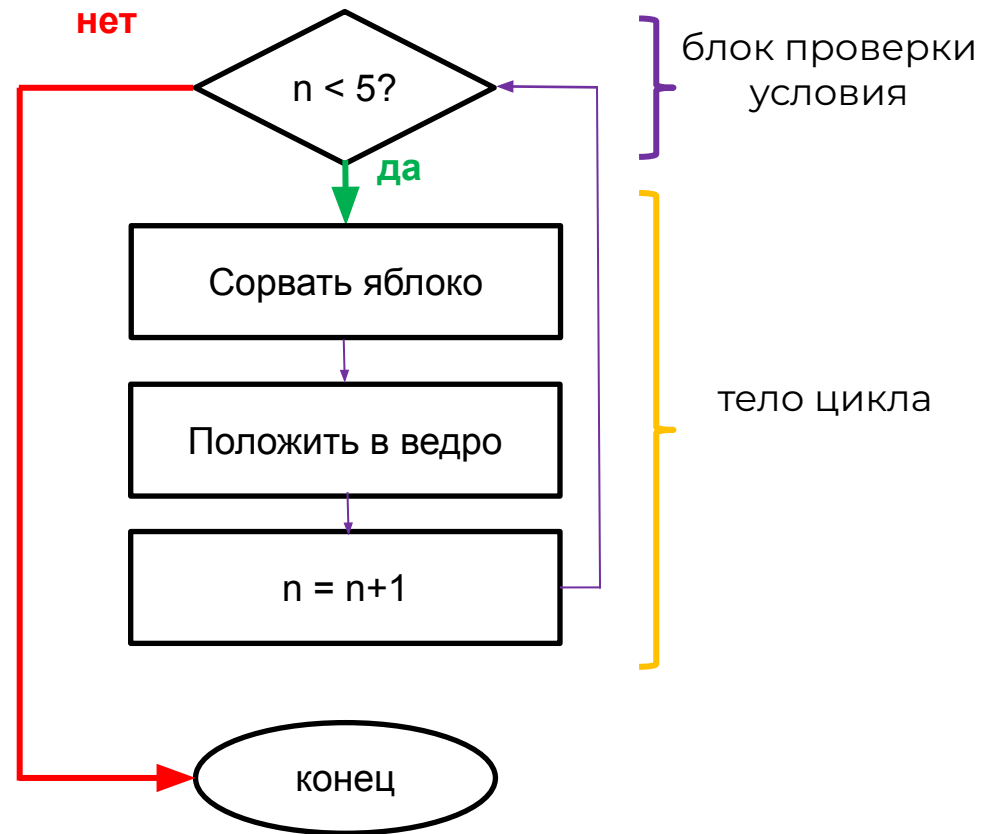
5



# Алгоритм «Сбор яблок»



**Цикл** – это участок алгоритма, осуществляющий повторение тела цикла. Цикл состоит из тела и блока проверки условия.



# Задание

---

Вывести на экран 3 раза слово Hello.

- Для чего нужна переменная?
- Чему равна переменная в момент запуска программы?
- Сколько слов изначально было напечатано?
- Какая команда отвечает за вывод информации?
- Какое условие нужно написать?
- Какие действия будут повторяться?

# Цикл в Python. Пример

---

Вывести на экран 3 раза слово Hello.

```
i = 0
while i < 3:
    print("Hello")
    i = i+1
```

# Цикл `while`

С помощью цикла `while` мы можем выполнять действия, пока условие верно.

Выводим `i`, до тех пор, пока `i` будет меньше 6:

```
i = 1
while i < 6:
    print(i)
    i += 1
```

Вывод:

1  
2  
3  
4  
5

# Цикл while

```
n = 5
while n > 0:
    n -= 1
    print(n)
```

Вывод:

4  
3  
2  
1  
0

# Бесконечный цикл

- Цикл `while` становится бесконечным в случае, когда условие цикла никогда не становится ложным.
- Примером задачи, для реализации которой необходим бесконечный цикл, может быть, например, создание программы "Часы", которая бесконечно будет обновлять и отображать время.
- Однако, часто бесконечный цикл является ошибкой начинающего программиста, который забыл добавить изменение условия цикла. Например:

```
num = 1
while num < 10 :
    print('У нас бесконечный цикл!')
```

- Не спешите запускать данный цикл, иначе ваша программа начнет бесконечное выполнение.

# Бесконечный цикл

Пример:

```
while True:
```

```
    print("У попа была собака, он её любил.")  
    print("Она съела кусок мяса, он её убил,")  
    print("В землю закопал и на камне написал:")
```

Запустить эту программу на компьютере.

(Чтобы приостановить выполнение программы, можно щёлкнуть по тексту правой кнопкой мыши).

Запись в тетрадь!

Бесконечный цикл:

```
while True:
```

```
    <действия>
```

# Цикл `while`. Прерывание цикла

С помощью оператора `break` мы можем остановить цикл, даже если условие `while` ИСТИННО:

Выходите из цикла когда он равен 3:

```
i = 1
while i < 6:
    print(i)
    if i == 3:
        break
    i += 1
```

Вывод:

```
1
2
3
```



# Досрочный выход из цикла

## break

Пример:

```
import random
while True:
    a=random.randint(1,10)
    print(a)
    if a==7:
        break
```

Запустить программу на компьютере.



## Задачи:

- 1) С помощью цикла `while` вывести любую строчку из стихотворения или песни 7 раз.
- 2) Написать программу, которая получает два целых числа  $A$  и  $B$  ( $0 < A < B$ ) и выводит все натуральные числа в интервале от  $A$  до  $B$ .
- 3) С помощью бесконечного цикла реализовать игру «Куп слона». Программа должна выводить фразу «Куп слона!», получать ответ пользователя и выводить: «Все говорят “фраза, введённая пользователем”. А ты купи слона!» Затем программа получает новый ответ от пользователя и так до бесконечности.



## Задачи:

- 4) С помощью бесконечного цикла и оператора break реализовать игру «Угадай число». Программа генерирует случайное число в диапазоне от 1 до 7 и говорит пользователю: «Угадай число от 1 до 7!». В бесконечном цикле программа считывает ответы пользователя и подсказывает ему «больше!» или «меньше!», а в случае правильного ответа цикл завершается, и выводятся сердечные поздравления с победой.
- 5) Доработать программу «Угадай число» так, чтобы она не заканчивалась, то есть, чтобы после угадывания одного числа, начиналось угадывание следующего. Увеличить диапазон с 7 до 15 и ввести ограничение на количество попыток (например, 3 попытки). Если пользователь не справился за 3 попытки, вывести «GAME OVER».



## Домашнее задание

Написать программы:

- 1) С помощью цикла `while` вывести повторяющуюся строку из любой песни 25 раз.
- 2) Написать программу, которая получает два целых числа  $A$  и  $B$  ( $0 < A < B$ ) и выводит квадраты всех натуральных чисел в интервале от  $A$  до  $B$ .
- 3) Дано целое число  $N$  ( $>0$ ). Используя операции деления нацело и взятия остатка от деления, вывести все его цифры, начиная с самой правой.
- 4) \*\* Напишите программу, которая бы «подбрасывала» условную монету 100 раз и сообщала, сколько раз выпал орел, а сколько — решка.

# Цикл while. Оператор continue

С помощью оператора **continue** мы можем остановить текущую итерацию и перейти к выполнению следующей:

Продолжайте до следующей итерации пока  $i$  равна 3:

```
i = 0
while i < 6:
    i += 1
    if i == 3:
        continue
    print(i)
```

Вывод:

1  
2  
4  
5  
6

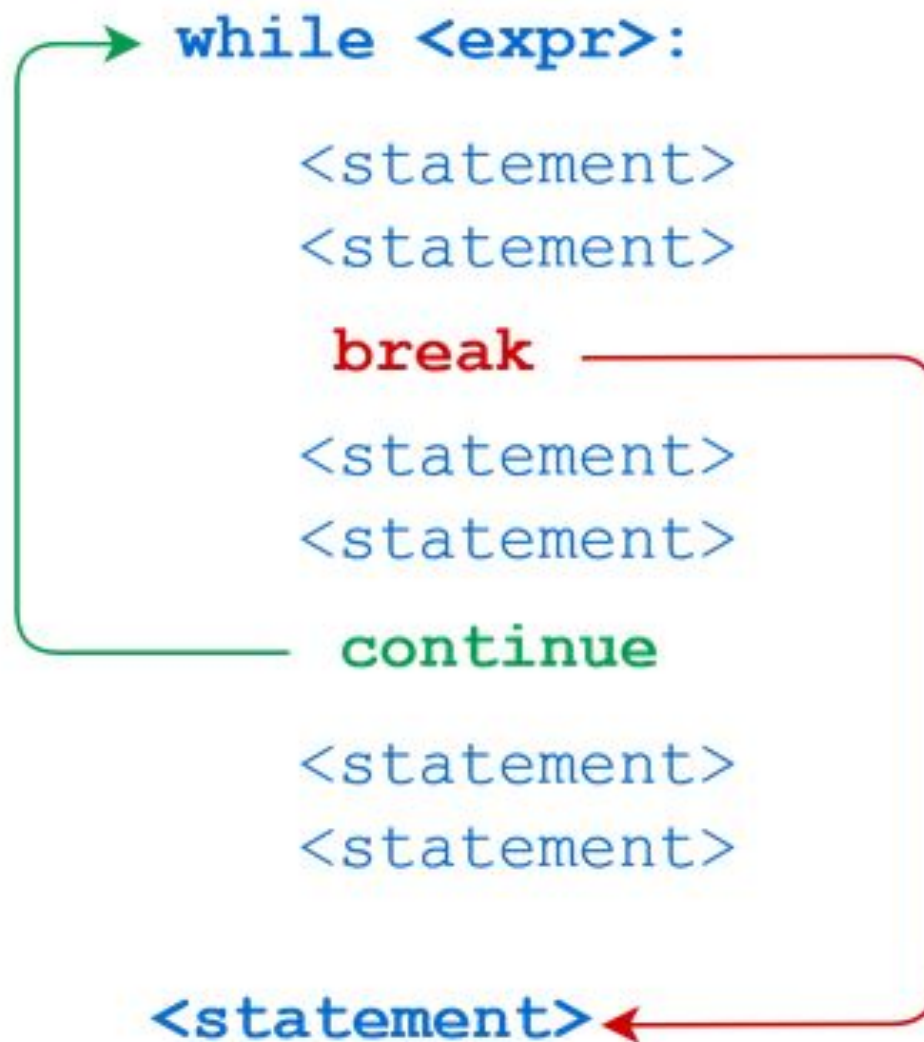
# Пример условия else в цикле while

```
i = 0
while i < 3:
    print(i)
    i += 1
else:
    print('done')
```

```
0
1
2
done
```

Условие **else** выполняется только после завершения цикла **while**, когда его условное выражение становится ложным

# Цикл while. break и continue



# Цикл For

Цикл for преимущественно используется когда известно необходимое количество итераций.



# Цикл for

- Цикл **for** преимущественно используется для итерации по последовательности (**list**, **tuple**, **dict**, **set** или **str**).
- Общий синтаксис **for...in** в python выглядит следующим образом:

```
for <переменная> in  
<последовательность>:    <действие>  
else: <действие>
```

# Цикл For. Итерация по строке

- Даже строки являются итерируемыми объектами и содержат последовательность СИМВОЛОВ.
- Получим буквы слова “Банан”:

```
for x in "Банан":  
    print(x)
```

Вывод:

Б  
а  
н  
а  
н

# Цикл For. Итерация по строке

```
word = 'Слово'  
for letter in word:  
    print(letter)
```

Вывод:

С  
Л  
О  
В  
О

# Цикл For. Перебираем элементы словаря

```
# Перебираем элементы словаря
# По-умолчанию цикл проходится по ключам словаря
# Значение по ключу получаем самостоятельно seasons[s]
seasons = {1: 'Зима', 2: 'Весна',
3: 'Лето', 4: 'Осень'}
for s in seasons:
    print('Номер сезона:', s)
    print('Название:', seasons[s])
```

```
Номер сезона: 1
Название: Зима
Номер сезона: 2
Название: Весна
Номер сезона: 3
Название: Лето
Номер сезона: 4
Название: Осень
```

# Цикл for. Перебираем элементы списка

```
fruits = ["яблоко", "банан", "вишня"]  
for x in fruits:  
    print(x)
```

Вывод:

яблоко  
банан  
вишня

# Цикл for. Перебираем элементы списка

```
languages = ["C", "C++", "Java",  
             "Python"]  
  
for x in languages :  
    print(x)
```

Вывод:

C

C++

Java

Python

# Цикл For. Оператор break

- Благодаря оператору **break** мы можем остановить цикл прежде чем он закончится по всем элементам:
- Завершим из цикл когда x — “банан”:

```
fruits = ["яблоко", "банан", "вишня"]  
for x in fruits:  
    print(x)  
    if x == "банан":  
        break
```

Вывод:

яблоко  
банан

# Цикл For. Оператор `break` и `continue`

- **`break`** – прерывает цикл и выходит из него;
- **`continue`** – прерывает текущую итерацию и переходит к следующей.



# Цикл For. Оператор break

- Выйдем из цикла когда `x` — “банан”, но в этот раз `if` будет перед выводом:

```
fruits = ["яблоко", "банан", "вишня"]
for x in fruits:
    if x == "банан":
        break
    print(x)
```

Вывод:

яблоко

# Цикл For. Оператор continue

- С помощью оператора **continue** мы можем остановить текущую итерацию цикла и перейти к следующей
- Пропустим вывод "банан":

```
fruits = ["яблоко", "банан", "вишня"]  
for x in fruits:  
    if x == "банан":  
        continue  
    print(x)
```

Вывод:

яблоко  
вишня

# Цикл For. Функция range()

- Функция `range()` применяется что бы выполнить действия заданное количество раз.
- Она возвращает последовательность чисел, начиная с 0 (по умолчанию) увеличивает число на 1 (по умолчанию) и заканчивая указанным числом.

```
for x in range(6):  
    print(x)
```

Вывод:

```
0  
1  
2  
3  
4  
5
```

# Цикл For. Функция range()

```
# скажем Миру привет целых пять раз!  
for i in range(5):  
    print("Hello World!")
```

```
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!
```

# Цикл For. Функция range()

# равносильно инструкции for i in 0, 1, 2, 3

```
for i in range(4) :
```

```
    # здесь можно выполнять  
    циклические действия
```

```
        print(i)
```

```
        print(i ** 2)
```

```
# цикл закончился, поскольку  
закончился блок с отступом
```

```
print('Конец цикла')
```

Вывод:

0

0

1

1

2

4

3

9

Конец

цикла

# Цикл For. Функция range()

- **range ()** можно представлять, как функцию, что возвращает последовательность чисел, регулируемую количеством переданных в неё аргументов.
- Их может быть 1, 2 или 3:
  - **range (finish) ;**
  - **range (start, finish) ;**
  - **range (start, finish, step) .**
- Здесь **start** – это первый элемент последовательности (включительно),
- **finish** – последний (не включительно),
- а **step** – разность между следующим и предыдущим членами последовательности.

# Цикл For. Функция range()

Функция `range()` по умолчанию начинается с 0, однако можно изменить начальное значение, добавив параметр: `range(2, 6)`, что означает значения от 2 до 6 (но не включая 6):

```
for x in range(2, 6):  
    print(x)
```

Вывод:

```
2  
3  
4  
5
```

# Цикл For. Функция range()

Функция `range()` по умолчанию увеличивает последовательность на 1, однако можно указать значение приращения, добавив третий параметр: `range(2, 30, 3)` :

```
for x in range(2, 30, 3):  
    print(x)
```

Вывод:

2  
5  
8  
11  
14  
17  
20  
23  
26  
29



# Цикл For. Функция range()

```
# выведем числа от 100 до 1000 с шагом 150
for nums in range(100, 1000, 150):
    print(nums)
```

100  
250  
400  
550  
700  
850

# Цикл For. Функция range()

```
# выведем числа от 45 до 50 по убыванию  
# для этого установим step -1  
for nums in range(50, 44, -1):  
    print(nums)
```

50  
49  
48  
47  
46  
45

# Цикл For. Enumerate

**Enumerate** - позволяет автоматически считать итерации цикла

Функция **enumerate** также принимает необязательный аргумент (значение начала отсчета, по умолчанию 0), который делает ее еще более полезной.

```
fruits = ["яблоко", "банан", "вишня", "персик"]  
for c, value in enumerate(fruits, 1):  
    print(c, value)
```

Вывод

```
1 яблоко  
2 банан  
3 вишня  
4 персик
```

# Цикл For. Enumerate

```
for index, item in enumerate(['один', 'два',  
'три', 'четыре']):  
    print(index, '::', item)
```

```
0 :: один  
1 :: два  
2 :: три  
3 :: четыре
```

# Else в цикле For

- Ключевое слово **else** в цикле **for** включает блок кода, который должен быть выполнен после завершения цикла:

```
for x in range(6):  
    print(x)  
else:  
    print("Цикл завершен!")
```

Вывод:

0

1

2

3

4

5

Цикл

завершен!

# Вложенный цикл

- **Вложенный цикл** — это цикл в цикле. Он будет запускаться при каждой итерации основного цикла.
- Выведем все фрукты с каждым прилагательным:

```
adj = ["желтый", "большой", "вкусный"]
```

```
fruits = ["апельсин", "банан",
```

```
for x in adj:  
    for y in fruits:  
        print(x, y)
```

Вывод:

```
желтый апельсин  
желтый банан  
желтый ананас  
большой  
апельсин  
большой банан  
большой ананас  
вкусный апельсин  
вкусный банан  
вкусный ананас
```

# Дополнительные примеры программ с циклами

# Посчитаем количество символов (с пробелами) в строке

```
myText = "Посчитаем сколько символов в строке  
с пробелами."
```

```
# с помощью цикла for посчитаем количество  
символов (с пробелами) в строке
```

```
# зададим счетчик
```

```
count = 0
```

```
# будем посимвольно обходить весь текст
```

```
for letter in myText:
```

```
    # на каждой новой итерации:
```

```
    # в переменной letter будет храниться  
    следующий символ предложения;
```

```
    # увеличиваем счетчик на 1;
```

```
    count += 1
```

```
print(count)
```

48



# Посчитаем количество слов в строке

```
s = "Посчитаем количество слов в строке"  
count = 0  
flag = 0  
for i in range(len(s)):  
    if s[i] != ' ' and flag == 0:  
        count += 1  
        flag = 1  
    else:  
        if s[i] == ' ':  
            flag = 0  
print(count)
```

# Посчитаем количество слов в строке

```
c = 0
s = "Посчитаем количество слов в строке"
for i in s:
    if i == ' ':
        c += 1
print(c+1)
```