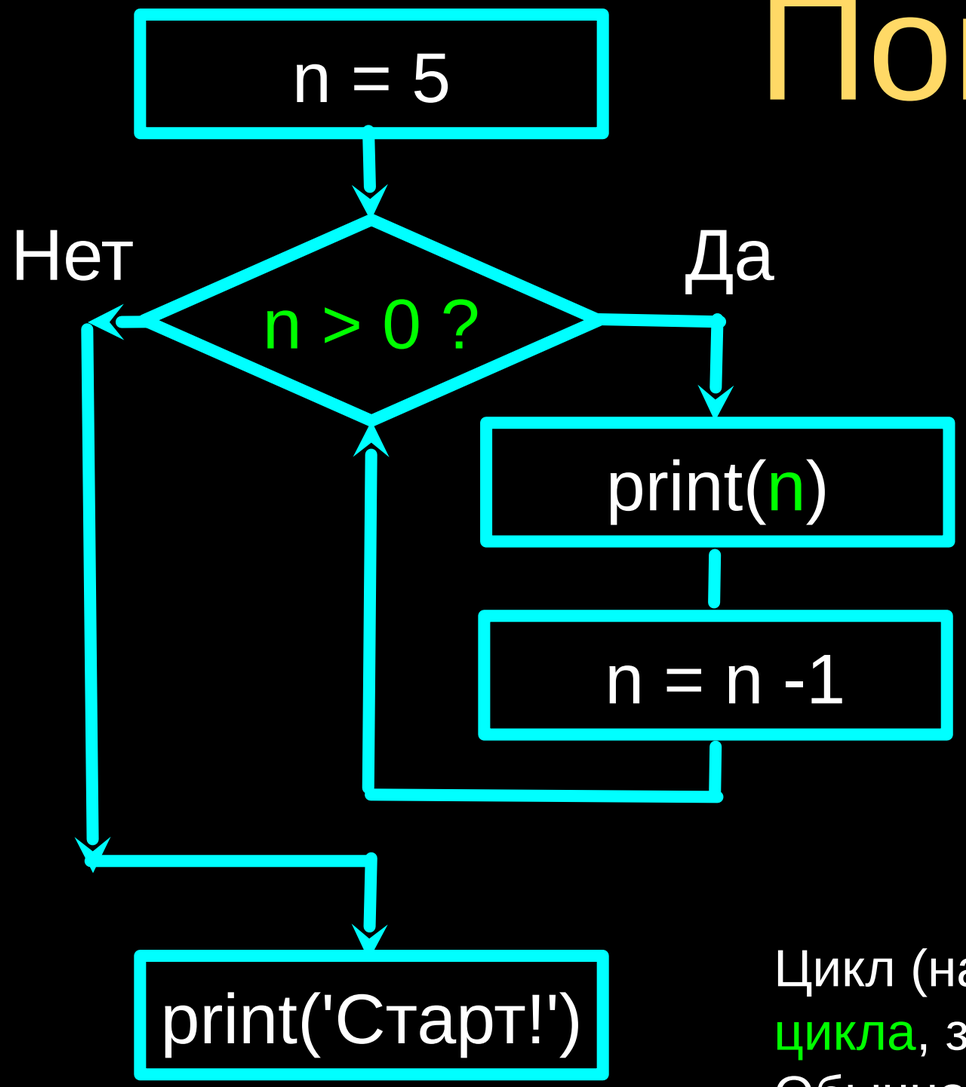
Циклы и повторения (итерации)

Глава 5

Пайтон для всех

www.py4e.com

Повторяющиеся шаги



Программа:

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

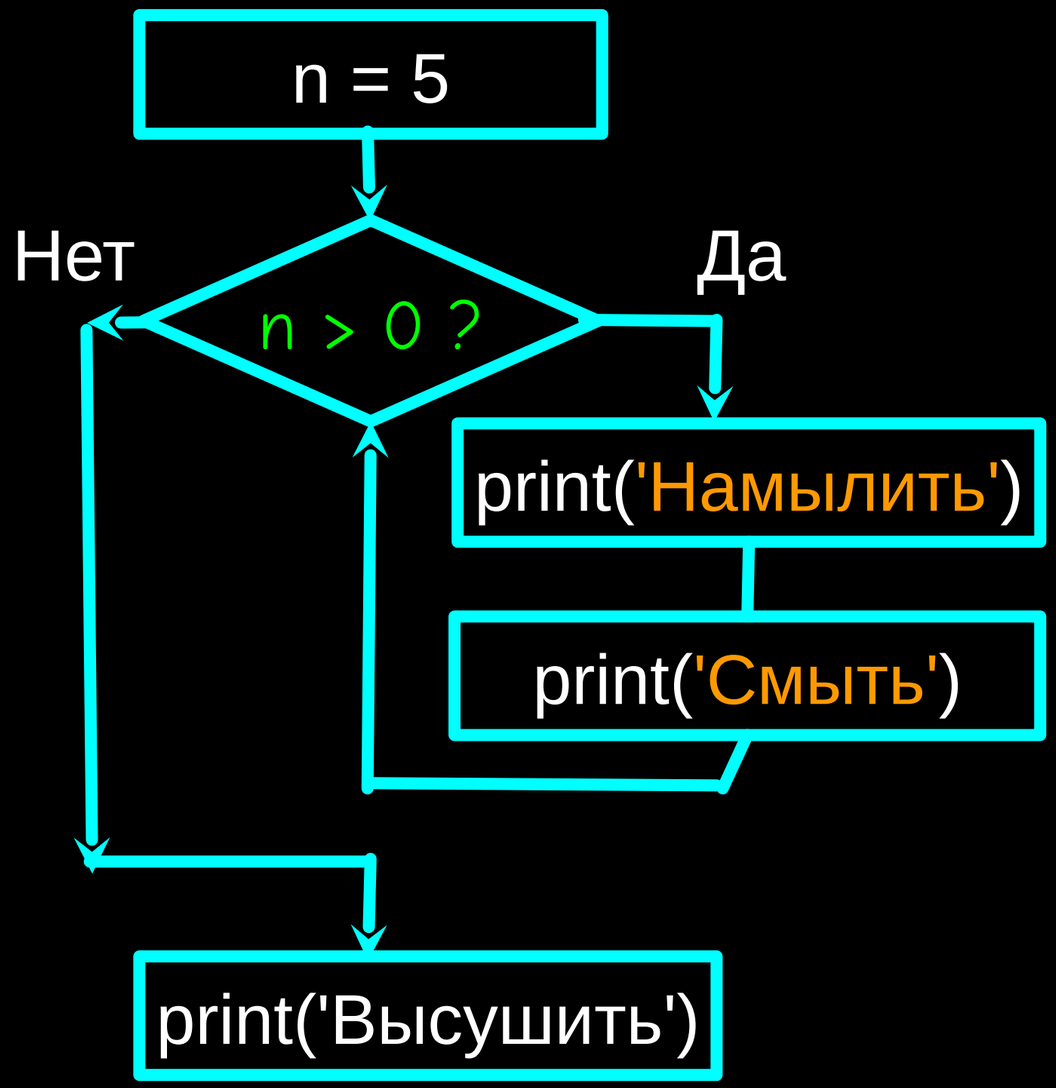
Результат:

```
5
4
3
2
1
Старт!
0
```

Цикл (набор повторяющихся шагов) имеет переменную — счетчик цикла, значение которой изменяется с каждым проходом цикла. Обычно счетчик цикла / переменная цикла проходит через последовательность чисел.

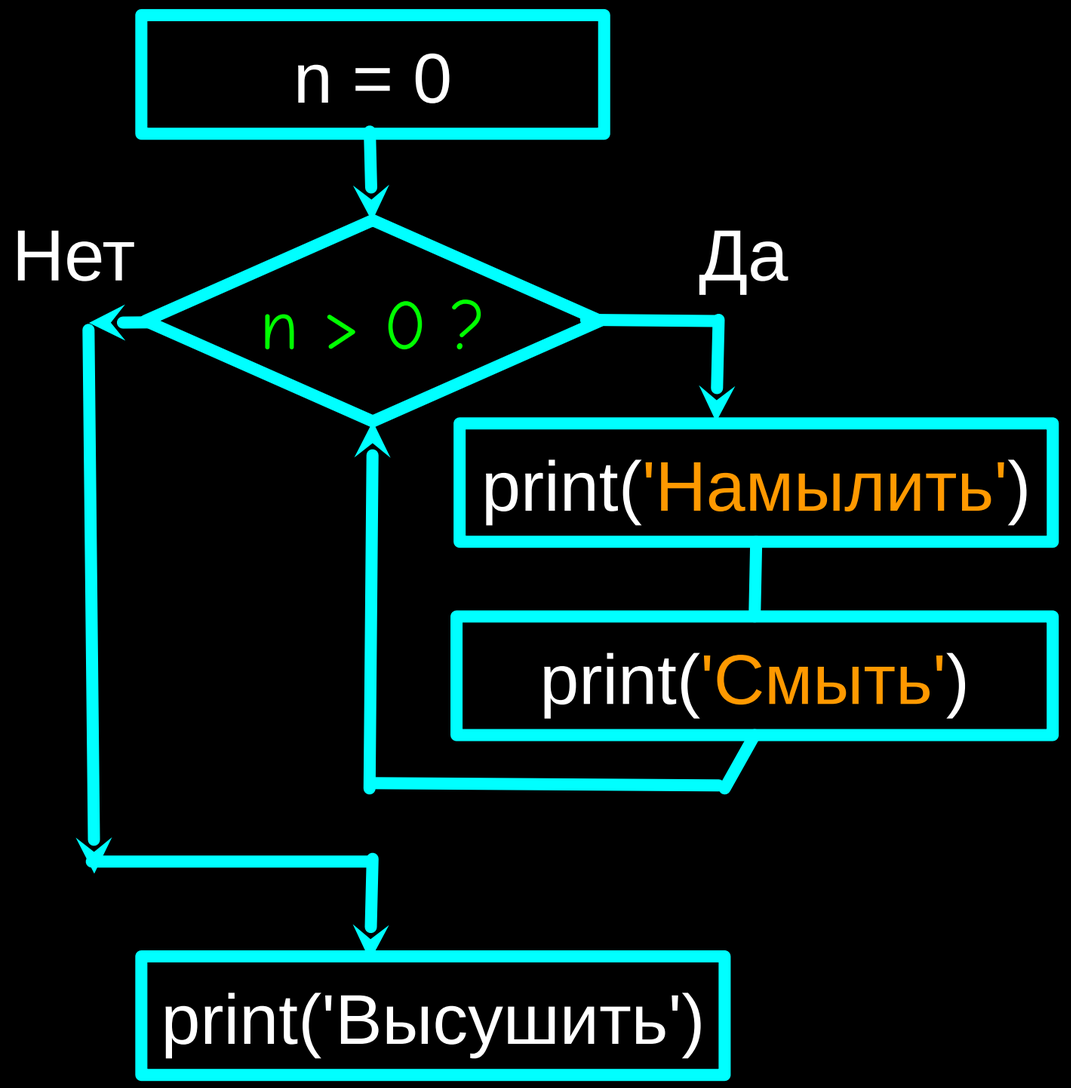
.

Бесконечный цикл




```
n = 5
while n > 0 :
    print('Намылить')
    print('Смыть')
print('Высушить')
```

Что не так с этим циклом?



Другой цикл

```
n = 0
while n > 0 :
    print('Намылить')
    print('Смыть')
print('Высушить')
```

Что делает этот цикл?

Выход из цикла

- Оператор break завершает/прерывает текущий цикл и переходит к оператору, следующему сразу после цикла
- Подобное тестирование цикла может быть произведено в любом месте цикла

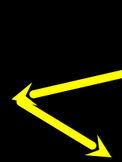
```
while True:
    line = input('> ')
    if line == 'готово' :
        break
    print(line)
print('Готово!')
```

```
> привет
привет
> сделано
сделано
> готово
Готово!
```

Выход из цикла

- Оператор break завершает/прерывает текущий цикл и переходит к оператору, следующему сразу после цикла
- Подобное тестирование цикла может быть произведено в любом месте цикла

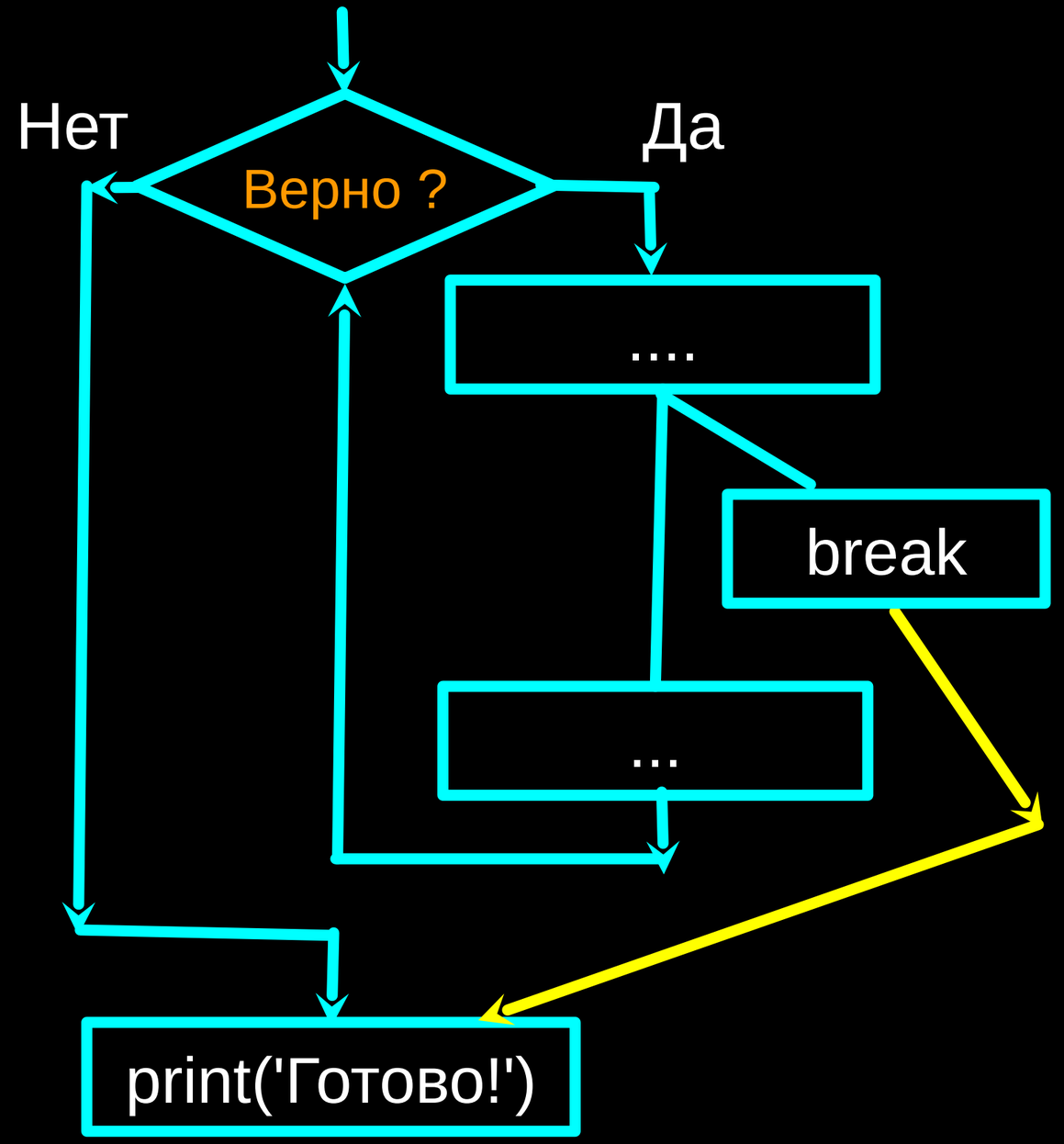
```
while True:
    line = input('> ')
    if line == 'готово' :
        break
    print(line)
print('Готово!')
```



```
> привет
привет
> сделано
сделано
> готово
Готово!
```

```
while True:
    line = input('> ')
    if line == 'готово' :
        break
    print(line)
print('Готово!')
```

http://en.wikipedia.org/wiki/Transporter_(Star_Trek)



Окончание итерации с помощью continue

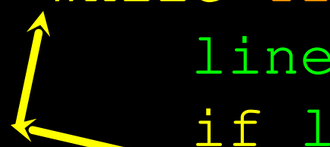
Оператор continue завершает текущий шаг цикла, переходит в начало цикла и начинает следующую итерацию цикла

```
while True:
    line = input('> ')
    if line[0] == '#' :
        continue
    if line == 'готово' :
        break
    print(line)
print('Готово!')
```

```
> привет
привет
> # не печатай это
> напечатай это!
напечатай это!
> готово
Готово!
```

Окончание итерации с помощью continue

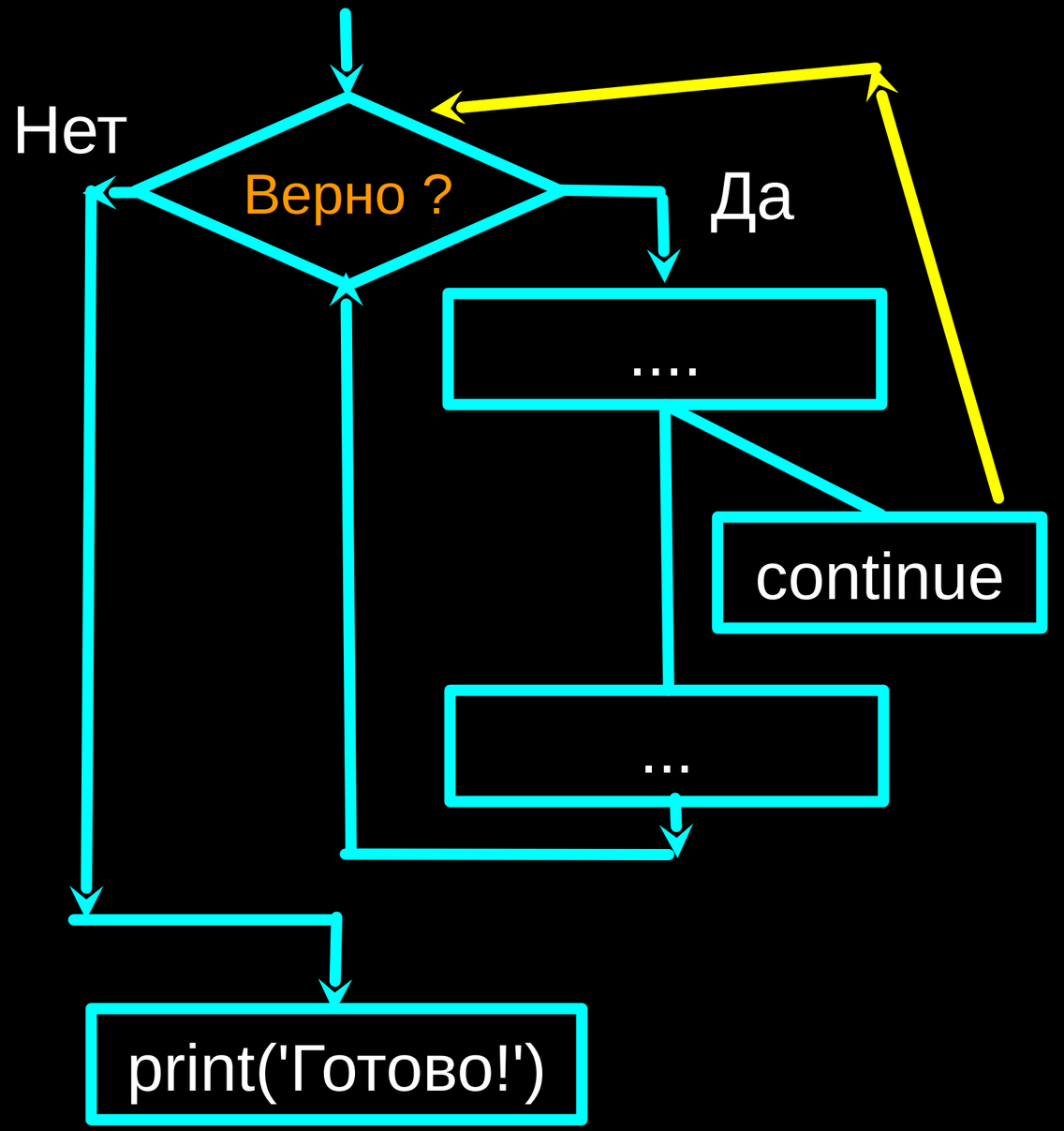
Оператор continue завершает текущий шаг цикла, переходит к началу цикла и начинает следующую итерацию цикла



```
while True:
    line = input('> ')
    if line[0] == '#' :
        continue
    if line == 'готово' :
        break
    print(line)
print('Готово!')
```

```
> привет
привет
> # не печатай это
> напечатай это!
напечатай это!
> готово
Готово!
```

```
while True:
    line = raw_input('> ')
    if line[0] == '#' :
        continue
    if line == 'готово' :
        break
    print(line)
print('Готово!')
```



Неопределенные циклы

- Циклы с оператором While называются «неопределенными циклами», потому что они продолжают свое выполнение до тех пор, пока логическое условие не станет равно False/Ложь
- Циклы, с которыми мы сталкивались до сих пор, достаточно просты, чтобы проверить являются ли они «бесконечными» или завершатся
- Но иногда бывает сложно понять, завершится цикл или нет

Циклы со счетчиком

Итерация по набору элементов…

Циклы со счетчиком

- Достаточно часто у нас есть некий список элементов, например, строк в файле, то есть фактически определенный конечный набор элементов
- Мы можем написать цикл, запускающийся однократно для каждого из элементов списка, используя оператор for
- Такие циклы называются «циклами со счетчиком», так как они выполняются определенное количество раз
- Мы говорим, что «циклы со счетчиком проходят через элементы набора»

Простейший цикл со счетчиком

```
for i in [5, 4, 3, 2, 1] :
    print(i)
print('Старт!')
```

```
5
4
3
2
1
Старт!
```

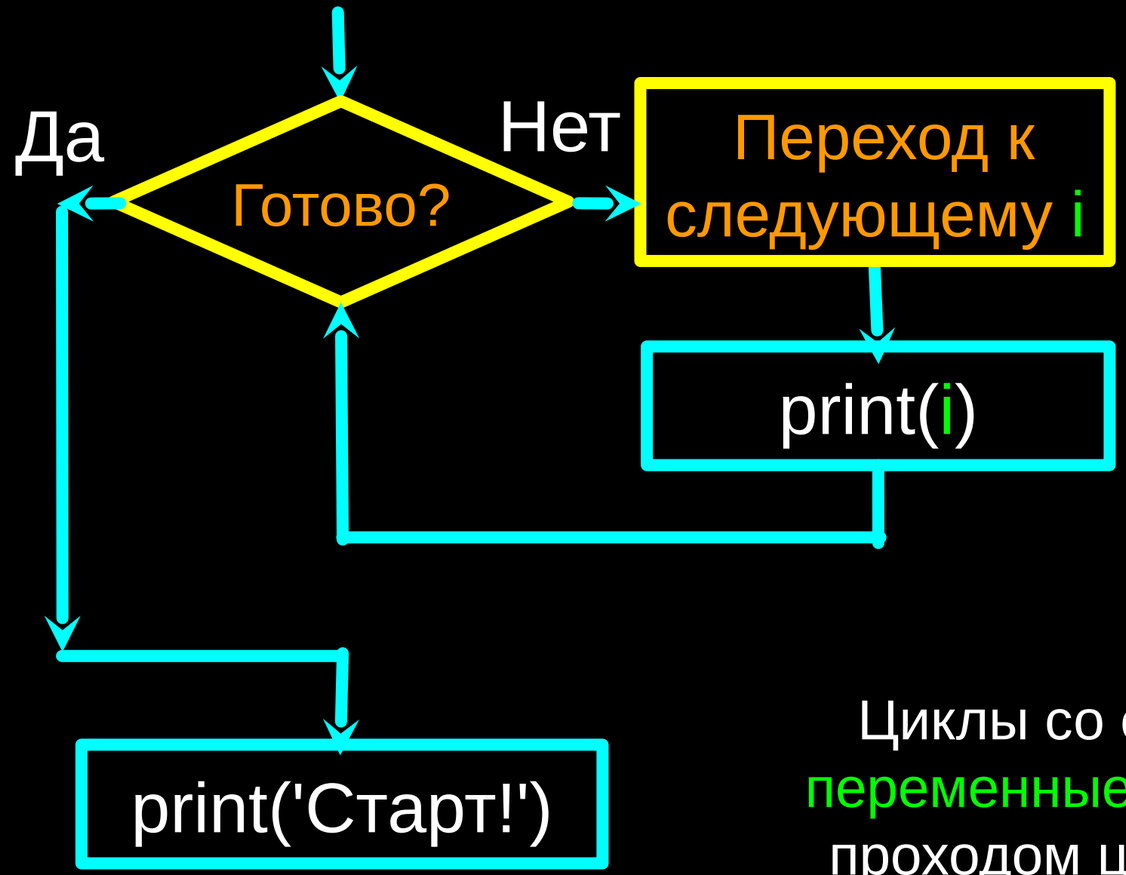
Цикл со счетчиком и элементами типа строка

```
friends = ['Василий', 'Дмитрий',
'Татьяна']
for friend in friends :
    print('С Новым Годом:', friend)
print('Готово!')
```

```
С Новым Годом: Василий
С Новым Годом: Дмитрий
С Новым Годом: Татьяна

Готово!
```

Простой цикл со счетчиком



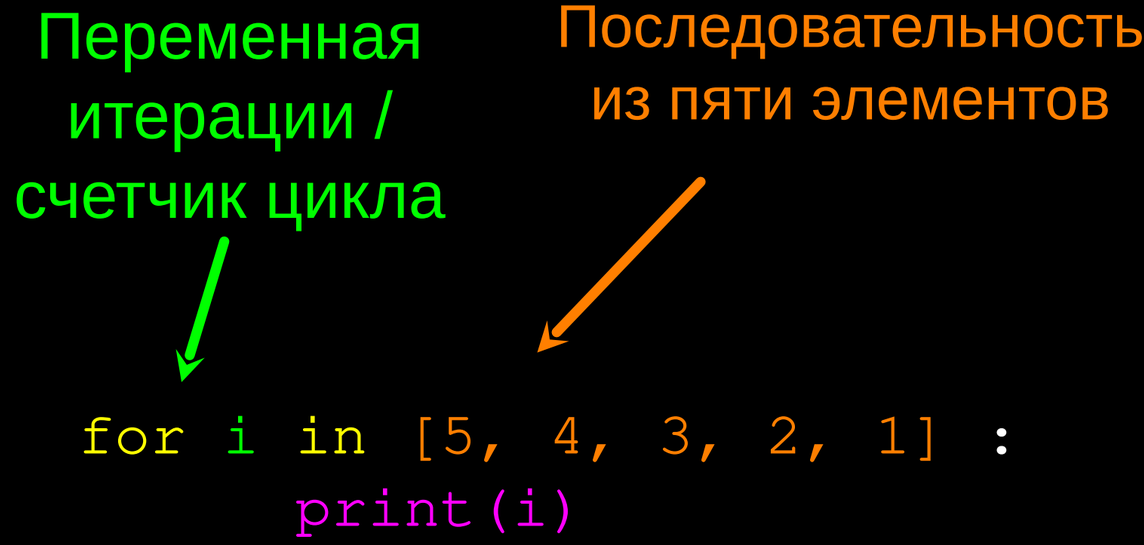

```
for i in [5, 4, 3, 2, 1] :
    print(i)
print('Старт!')
```

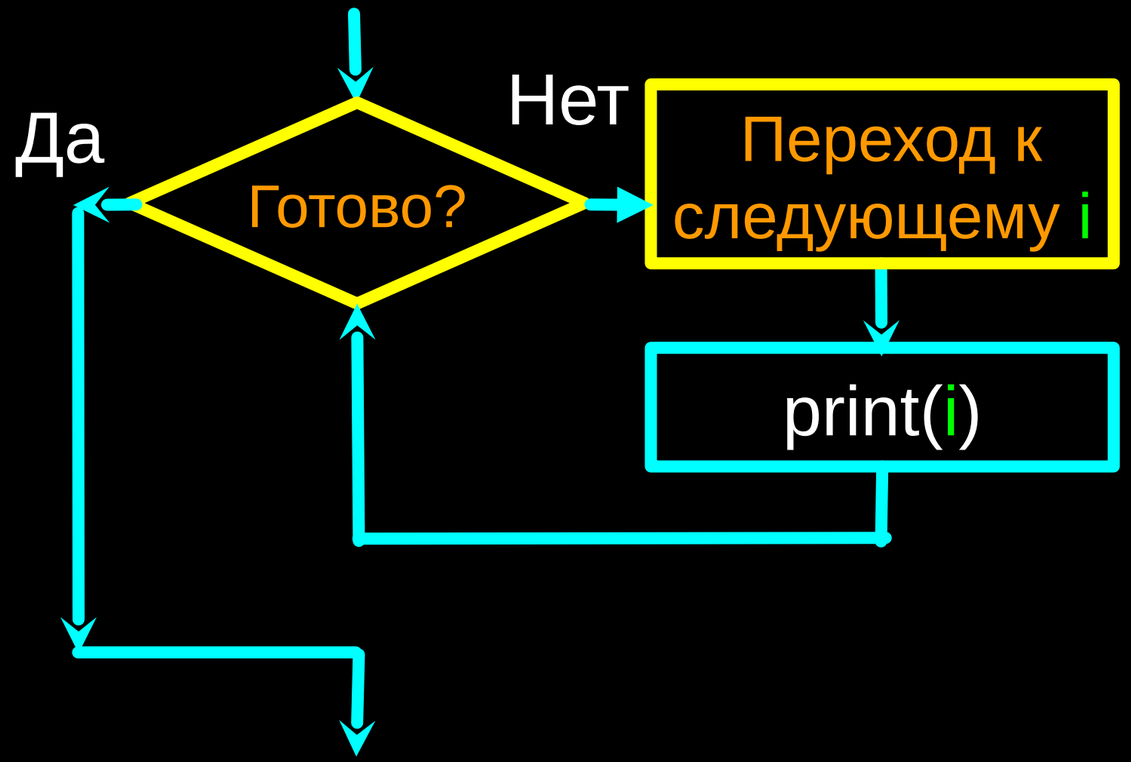
```
5
4
3
2
1
Старт!
```

Циклы со счетчиком (циклы с for) имеют заданные переменные итерации, которые изменяются с каждым проходом цикла. Переменные итерации проходят по последовательности или набору элементов

Конструкция с in...

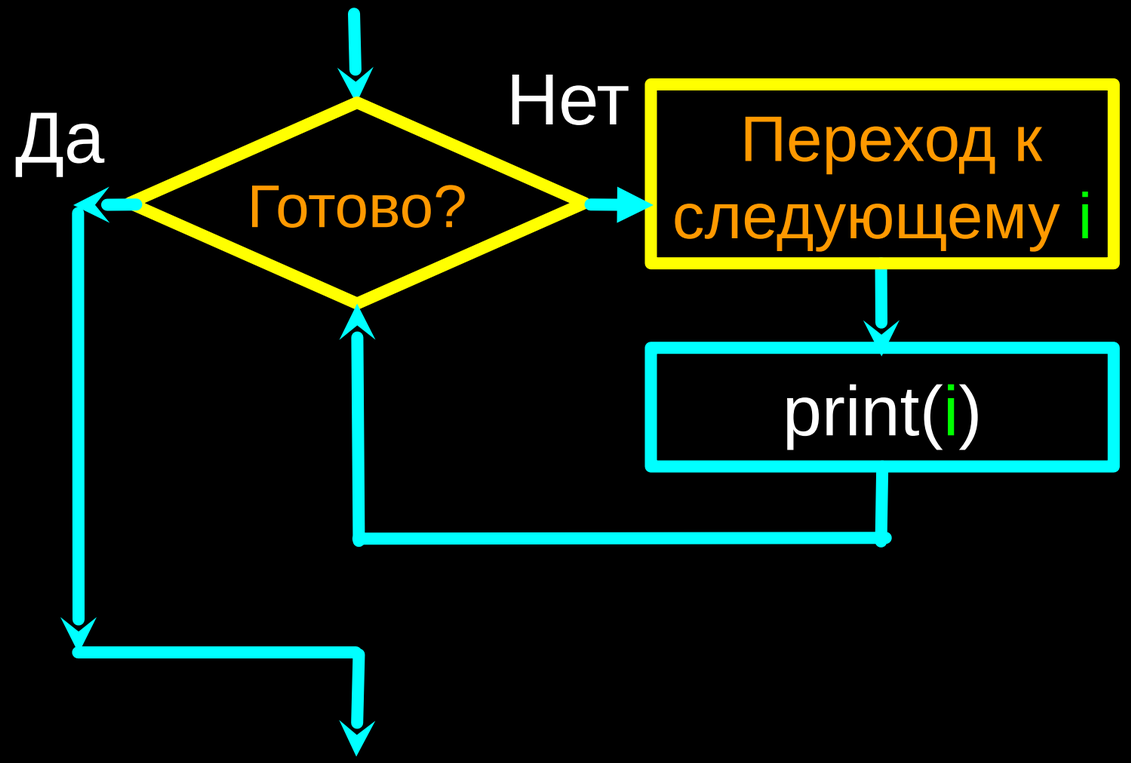
- Переменная итерации / счетчик цикла «движется» через последовательность (упорядоченный набор)
- Блок (тело) кода выполняется однократно для каждого значения в (in) последовательности
- Переменная итерации «движется» по всем значениям в последовательности



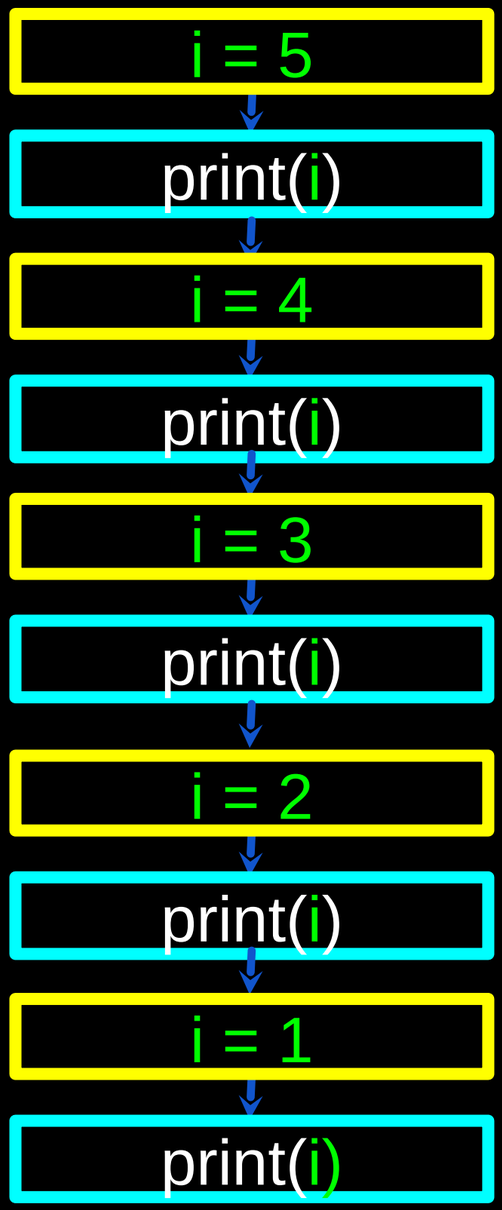



```
for i in [5, 4, 3, 2, 1] :
    print(i)
```

- Переменная итерации / счетчик цикла «движется» через последовательность (упорядоченный набор)
- Блок (тело) кода выполняется однократно для каждого значения в (in) последовательности
- Переменная итерации «движется» по всем значениям в последовательности




```
for i in [5, 4, 3, 2, 1] :
    print(i)
```



Язык циклов:
что мы делаем в циклах

Примечание: несмотря на то, что эти примеры просты, данные шаблоны применимы ко всем видам циклов

Пишем «умные» циклы

Трюк в том, чтобы заранее «знать» нечто обо всем цикле, когда вы застряли при написании кода, который видит только одну запись за раз

Установите для некоторых переменных начальные значения

для (for) элемента в (in) данных:

Ищите какое-либо значение или делайте что-нибудь с каждым элементом по-отдельности, обновляя переменную

Смотрите на переменные

Итерация по набору элементов

```
print('Начало')
for thing in [9, 41, 12, 3, 74, 15] :
     print(thing)
print('Конец')
```

```
$ python basicloop.py
Начало
9
41
12
3
74
15
Конец
```

Какое из чисел больше?

Какое из чисел больше?

3

Какое из чисел больше?

41

Какое из чисел больше?

12

Какое из чисел больше?

9

Какое из чисел больше?

74

Какое из чисел больше?

15

Какое из чисел больше?

Какое из чисел больше?

3 41 12 9 74 15

Какое из чисел больше?

largest_so_far -1

Какое из чисел больше?

3

largest_so_far 3

Какое из чисел больше?

41

largest_so_far 41

Какое из чисел больше?

12

largest_so_far 41

Какое из чисел больше?

9

largest_so_far 41

Какое из чисел больше?

74

largest_so_far 74

Какое из чисел больше?

15

74

Какое из чисел больше?

3 41 12 9 74 15

74

Нахождение наибольшего значения

```
largest_so_far = -1
print('В начале', largest_so_far)
for the_num in [9, 41, 12, 3, 74, 15] :
    if the_num > largest_so_far :
        largest_so_far = the_num
    print(largest_so_far, the_num)

print('В конце', largest_so_far)
```

```
$ python largest.py
В начале -1
9  9
41  41
41 12
41  3
74  74
74  15
В конце 74
```

Мы создаем переменную, которая хранит наибольшее значение, которое мы видели к этому моменту. Если текущее число, на которое мы смотрим, больше, то оно становится новым наибольшим значением, которое мы видели к этому моменту

Больше паттернов циклов…

Счет внутри цикла

```
zork = 0
print('В начале', zork)
for thing in [9, 41, 12, 3, 74, 15] :
    zork = zork + 1
    print(zork, thing)
print('В конце', zork)
```

```
$ python countloop.py
В начале 0
1 9
2 41
3 12
4 3
5 74
6 15
В конце 6
```

Чтобы подсчитать сколько раз выполняется цикл, мы вводим переменную-счетчик, значение которой стартует с 0. Значение этой переменной с каждым проходом цикла увеличивается на единицу

Суммирование в цикле

```
zork = 0
print('В начале', zork)
for thing in [9, 41, 12, 3, 74, 15] :
    zork = zork + thing
    print(zork, thing)
print('В конце', zork)
```

```
$ python countloop.py
В начале 0
9 9
50 41
62 12
65 3
139 74
154 15
В конце 154
```

Чтобы сложить значения, с которыми мы сталкиваемся в цикле, вводим переменную суммы, с начальным значением 0, а затем добавляем значение к сумме каждый раз по ходу цикла.

Нахождение среднего арифметического в цикле

```
count = 0
sum = 0
print('В начале', count, sum)
for value in [9, 41, 12, 3, 74, 15] :
    count = count + 1
    sum = sum + value
    print(count, sum, value)
print('В конце', count, sum, sum /
count)
```

```
$ python averageloop.py
В начале 0 0
1 9 9
2 50 41
3 62 12
4 65 3
5 139 74
6 154 15
В конце 6 154 25.666
```

Нахождение среднего арифметического объединяет операции подсчета, суммирования и деления по завершению цикла

Фильтрация в цикле

```
print('Начало')
for value in [9, 41, 12, 3, 74, 15] :
    if value > 20:
        print('Наибольшее
число',value)
print('Конец')
```

```
$ python search1.py
Начало
Наибольшее число 41
Наибольшее число 74
Конец
```

Мы используем оператор if в цикле, чтобы поймать / отфильтровать значения, которые нам нужны

Поиск с использованием логических (булевых) переменных

```
found = False
print('В начале', found)
for value in [9, 41, 12, 3, 74, 15] :
    if value == 3 :
        found = True
    print(found, value)
print('В конце', found)
```

```
$ python search1.py
В начале False
False 9
False 41
False 12
True 3
True 74
True 15
В конце True
```

Если нужно просто выполнить поиск и узнать, было ли найдено значение, мы используем переменную с начальным значением False (Ложь), которое изменится на True (Правда), как только мы нашли то, что искали

Как найти наименьшее значение

```
largest_so_far = -1
print('В начале', largest_so_far)
for the_num in [9, 41, 12, 3, 74, 15] :
    if the_num > largest_so_far :
        largest_so_far = the_num
    print(largest_so_far, the_num)

print('В конце', largest_so_far)
```

```
$ python largest.py
В начале -1
9 9
41 41
41 12
41 3
74 74
74 15
В конце 74
```

Как нам изменить код, чтобы он нашел наименьшее значение в списке?

Нахождение наименьшего значения

```
smallest_so_far = -1
print('В начале', smallest_so_far)
for the_num in [9, 41, 12, 3, 74, 15] :
    if the_num < smallest_so_far :
        smallest_so_far = the_num
    print(smallest_so_far, the_num)

print('В конце', smallest_so_far)
```

Мы изменили имя переменной на smallest_so_far, а также заменили > на <

Нахождение наименьшего значения

```
smallest_so_far = -1
print('В начале', smallest_so_far)
for the_num in [9, 41, 12, 3, 74, 15] :
    if the_num < smallest_so_far :
        smallest_so_far = the_num
    print(smallest_so_far, the_num)

print('В конце', smallest_so_far)
```

```
$ python smallbad.py
В начале -1
-1  9
-1  41
-1 12
-1  3
-1  74
-1  15
В конце -1
```

Мы изменили имя переменной на smallest_so_far, а также заменили > на <

Нахождение наименьшего значения

```
smallest = None
print('Начало')
for value in [9, 41, 12, 3, 74, 15] :
    if smallest is None :
        smallest = value
    elif value < smallest :
        smallest = value
    print(smallest, value)
print('Конец', smallest)
```

```
$ python smallest.py
Начало
9 9
9 41
9 12
3 3
3 74
3 15
Конец 3
```

У нас есть переменная с именем smallest. При первом проходе цикла переменная smallest имеет значение None. Таким образом, наименьшим значением становится первое значение из списка

Операторы is (то же самое) и is not (не то же самое)

```
smallest = None
print('Начало')
for value in [3, 41, 12, 9, 74, 15] :
    if smallest is None :
        smallest = value
    elif value < smallest :
        smallest = value
    print(smallest, value)

print('Конец', smallest)
```

- Пайтон оператор is может использоваться в логических выражениях
- Подразумевает «то же самое»
- Похож на оператор ==, но является более строгим сравнением
- is not — тоже логический оператор

Резюме

- Циклы с оператором while (неопределенные циклы)
- Бесконечные циклы
- Использование оператора break
- Использование оператора continue
- Константы и переменные типа None
- Циклы с оператором for (циклы со счетчиком)
- Счетчик цикла
- Язык циклов
- Наибольше и наименьшее значение

Авторы / Благодарности

Исходная разработка: Чарльз Северанс, Школа Информации Мичиганского Университета.

Перевод выполнила Фомкина Виолетта.

… Добавьте сюда новых авторов и переводчиков

...