Программирование на языке Python

Циклические алгоритмы

Что такое цикл?

Цикл – это многократное выполнение одинаковых действий.

Два вида циклов:

- цикл с **известным** числом шагов (сделать 10 раз)
- цикл с **неизвестным** числом шагов (делать, пока не надоест)

Задача. Вывести на экран 10 раз слово «Привет».

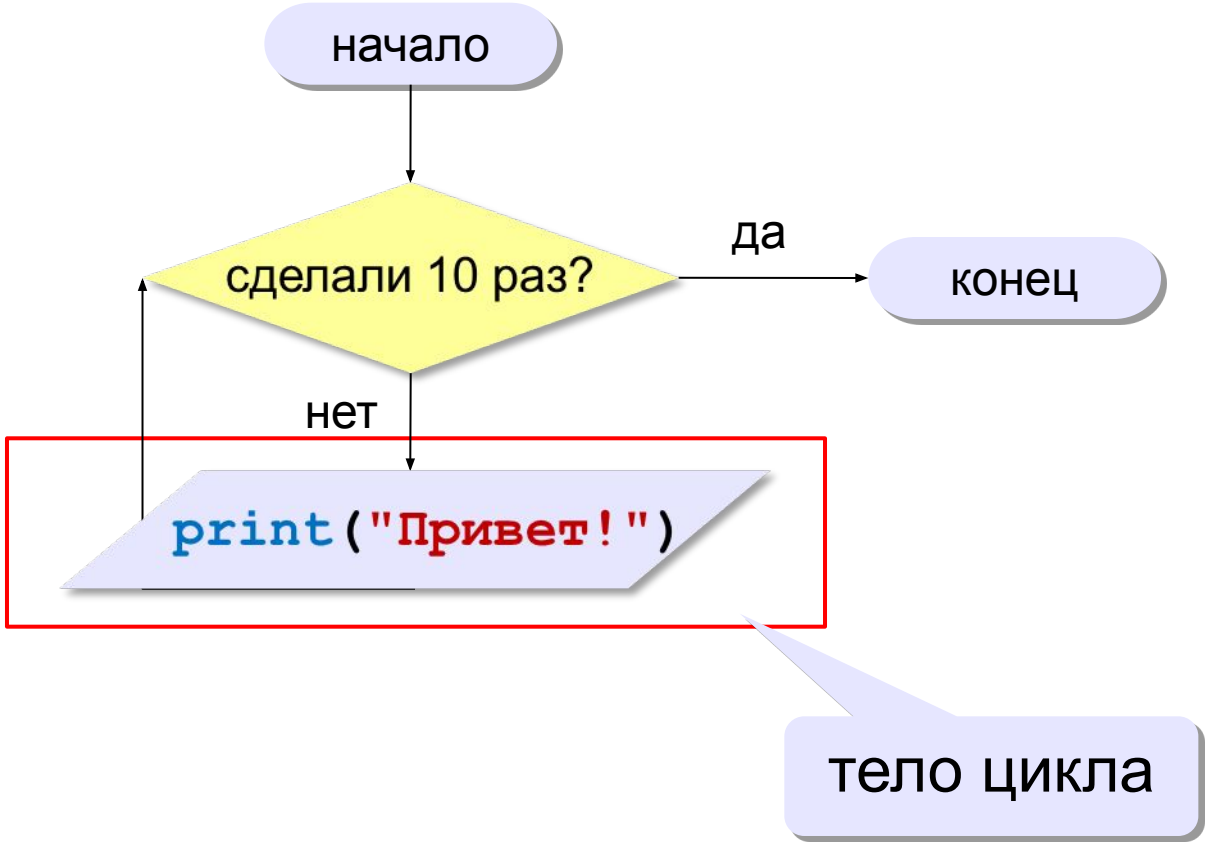
Можно ли решить известными методами?

Повторения в программе

```
print("Привет")
print("Привет")
...
print("Привет")
```

Что плохо?

Блок-схема цикла



Как организовать цикл?

```
счётчик = 0
пока счётчик < 10:
  print("Привет")
  увеличить счётчик на 1
```

```
k = 0
while k < 10:
  print ( "Привет" )
  k += 1
```

Как по-другому?

```
счётчик = 10
пока счётчик > 0:
  print("Привет")
  уменьшить счётчик на 1
```

```
k = 10
while k > 0:
  print ( "Привет" )
  k -= 1
```

Цикл while

Цикл **while (“пока”)** позволяет выполнить одну и ту же последовательность действий, пока проверяемое условие истинно. Условие записывается до тела цикла и проверяется до выполнения тела цикла. Как правило, цикл **while** используется, когда невозможно определить точное значение количества проходов исполнения цикла.

Синтаксис цикла **while** в простейшем случае выглядит так:

while *условие*:

    блок инструкций

Сколько раз выполняется цикл?

```
a = 4; b = 6
while a < b: a += 1
```

2 раза
a = 6

```
a = 4; b = 6
while a < b: a += b
```

1 раз
a = 10

```
a = 4; b = 6
while a > b: a += 1
```

0 раз
a = 4

```
a = 4; b = 6
while a < b: b = a - b
```

1 раз
b = -2

```
a = 4; b = 6
while a < b: a -= 1
```

зацикливание

Цикл с условием

Задача. Определить **количество цифр** в десятичной записи целого положительного числа, записанного в переменную **n**.

```
счётчик = 0
пока n > 0:
   отсечь последнюю цифру n
   увеличить счётчик на 1
```

n	счётчик
1234	0

? Как отсечь последнюю цифру?

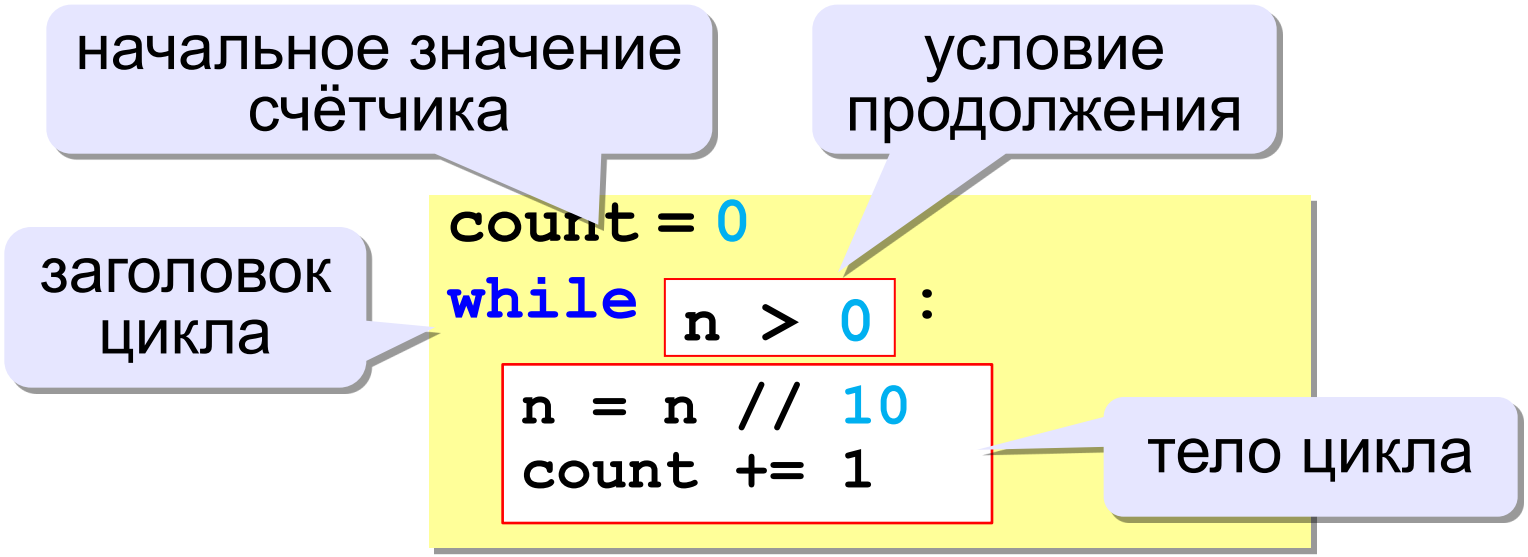
```
n = n // 10
```

? Как увеличить счётчик на 1?

```
счётчик = счётчик + 1
```

```
счётчик += 1
```

Цикл с условием



! Цикл с предусловием – проверка на входе в цикл!

Задачи

«3»: Ввести с клавиатуры количество повторений и вывести столько же раз какое-нибудь сообщение.

Пример:

```
Сколько раз:
5
Привет!
Привет!
Привет!
Привет!
Привет!
```

Задачи

«4»: Ввести с клавиатуры натуральное число и определить, сколько раз в его записи встречается цифра 1.

Пример:

```
Введите число:
51211
3
```

«5»: Ввести с клавиатуры натуральное число и найти сумму значений его цифр.

Пример:

```
Введите число:
1234
Сумма цифр 10
```

Задачи

«6»: Ввести натуральное число и определить, верно ли, что в его записи есть две одинаковые цифры, стоящие рядом.

Пример:

```
Введите натуральное число:
12342
Нет.
```

Пример:

```
Введите натуральное число:
12245
Да.
```

Задачи

«7»: По данному целому числу N распечатайте все квадраты натуральных чисел, не превосходящие N, в порядке возрастания.

Пример:

```
Введите натуральное число:
50
1 4 9 16 25 36 49
```

Пример:

```
Введите натуральное число:
5
1 4
```

Задачи

«7»: Программа получает на вход последовательность целых неотрицательных чисел, каждое число записано в отдельной строке. Последовательность завершается числом 0, при считывании которого программа должна закончить свою работу и вывести количество членов последовательности (не считая завершающего числа 0). Числа, следующие за числом 0, считывать не нужно. (доп – сумма и среднее)

Пример:

```
Введите последовательность (0 - конец):
1
7
9
0
5
3
```

Задачи

«8»: Дано целое число *N* (> 1). Если оно является *простым*, т. е. не имеет положительных делителей, кроме 1 и самого себя, то вывести 1, иначе вывести 0.

Пример:

```
Введите число:
17
1
```

Алгоритм Евклида

Алгоритм Евклида. Чтобы найти НОД двух натуральных чисел, нужно вычитать из большего числа меньшее до тех пор, пока они не станут равны. Это число и есть НОД исходных чисел.

НОД(14,21) = НОД(14,7) = НОД(7, 7) = 7

```
пока a != b:
  если a > b:
    a -= b  # a = a - b
  иначе:
    b -= a  # b = b - a
```

```
while a != b:
  if a > b:
    a -= b
  else:
    b -= a
```

НОД(1998,2) = НОД(1996,2) = … = НОД(2, 2) = 2

Алгоритм Евклида

Модифицированный алгоритм Евклида. Заменять большее число на остаток от деления большего на меньшее до тех пор, пока меньшее не станет равно нулю. Другое (ненулевое) число и есть НОД чисел.

НОД(1998,2) = НОД(0,2) = 2

```
пока a!=0 and b!=0:
  если a > b:
    a = a % b
  иначе:
    b = b % a
```

? Какое условие?

```
если a != 0:
  вывести a
иначе:
  вывести b
```

? Как вывести результат?

Задачи

«3*»: Ввести с клавиатуры два натуральных числа и найти их НОД с помощью алгоритма Евклида.

Пример:

```
Введите два числа:
21 14
НОД(21,14)=7
```

«4»: Ввести с клавиатуры два натуральных числа и найти их НОД с помощью **модифицированного** алгоритма Евклида. Заполните таблицу:

a	64168	358853	6365133	17905514	549868978
b	82678	691042	11494962	23108855	298294835
НОД(a,b)					

Задачи

«5»: Ввести с клавиатуры два натуральных числа и сравнить количество шагов цикла для вычисления их НОД с помощью обычного и модифицированного алгоритмов Евклида.

Пример:

```
Введите два числа:
1998 2
НОД(1998,2)=2
Обычный алгоритм: 998
Модифицированный: 1
```