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».

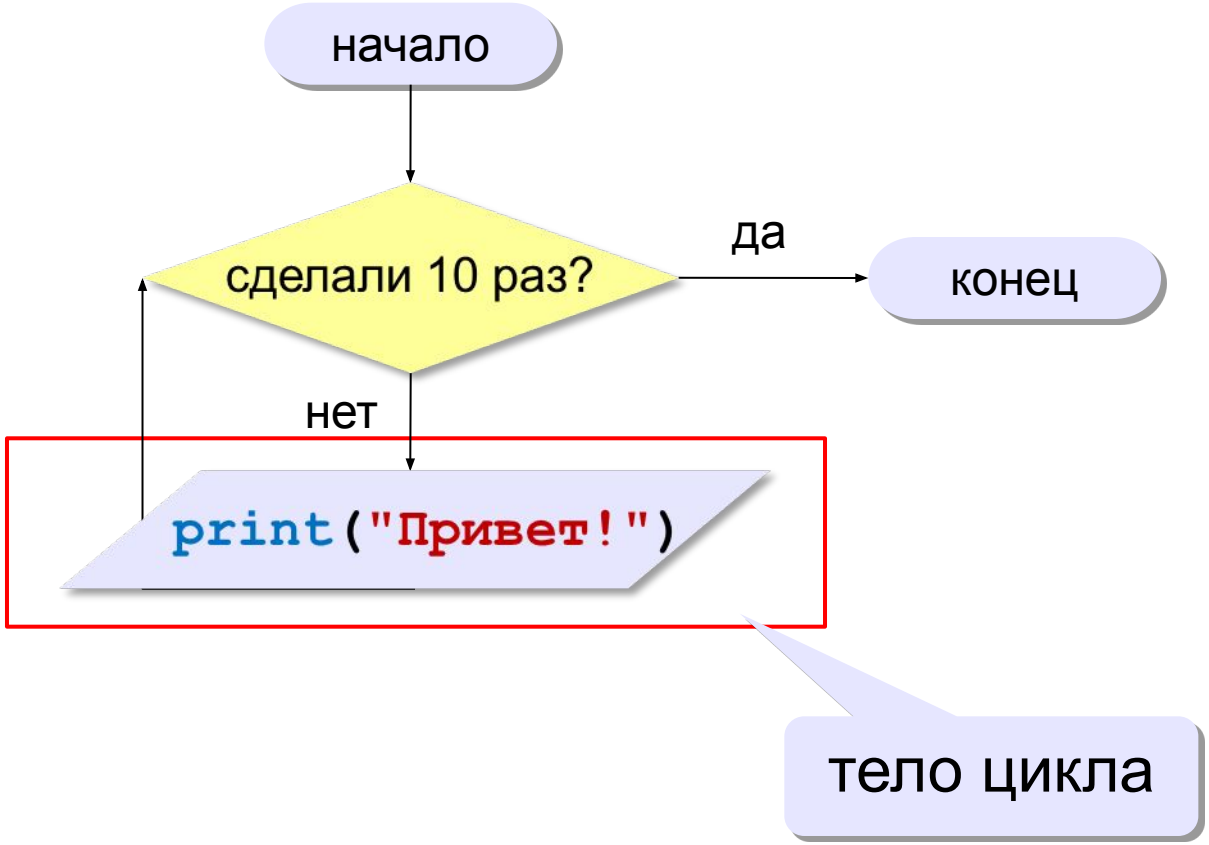
Можно ли решить известными методами?

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: a -= 1
```

зацикливание

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

1 раз
b = -2

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?

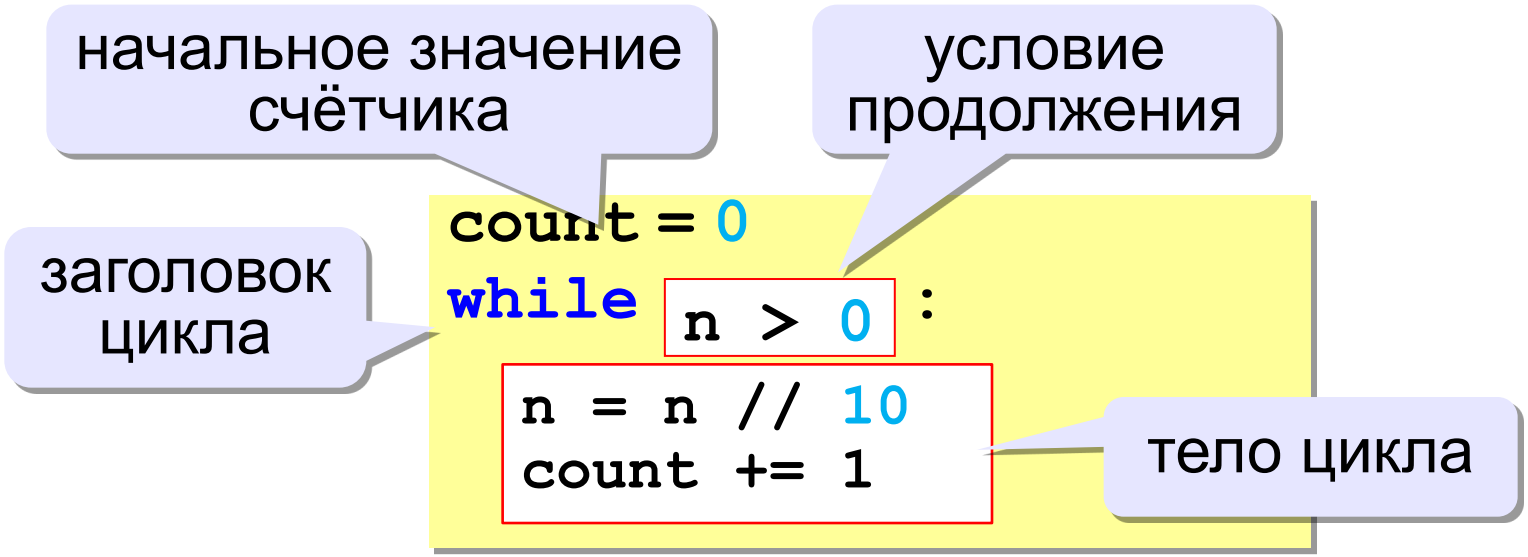
```
n = n // 10
```

? 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
```

Цикл for

Цикл **for**, также называемый циклом с параметром, в языке Питон богат возможностями. В цикле **for** указывается переменная и множество значений, по которому будет пробегать переменная. Множество значений может быть задано списком, кортежем, строкой или диапазоном. Тело цикла записывается с отступом.

В списке значений могут быть выражения различных типов, например:

```
for i in 1, 2, 3, 'one', 'two', 'three':
    print(i) # Здесь обязательно отступ
```

При первых трех итерациях цикла переменная **i** будет принимать значение типа **int**, при последующих трех — типа **str**.

Обработка строк в цикле

Задача. Ввести строку и определить, сколько в ней цифр.

```
счётчик = 0
для каждого символа строки:
  если символ – цифра:
    счётчик += 1
```

```
s = input()
k = 0
for c in s:
  if c.isdigit():
    k += 1
```

для всех символов в строке

если `c` – это цифра

Проверка символов

```
if c.isdigit():
  print("Цифра")
```

```
if c.isalpha():
  print("Буква")
```

```
if c.islower():
  print("Строчная буква")
```

```
if c.isupper():
  print("Заглавная буква")
```

```
if c in ["а","б"]:
  print("Это а или б")
```

Задачи

«2»: Вывести на экран циклом пять строк из нулей, причем каждая строка должна быть пронумерована

Пример:

```
1) 00000
2) 00000
3) 00000
4) 00000
5) 00000
```

Задачи

«3»: Ввести с клавиатуры число в двоичной системе счисления. Определить, сколько в его записи единиц и сколько нулей.

Пример:

```
Введите число:
1010100
Нулей: 4
Единиц: 3
```

«4»: Ввести с клавиатуры символьную строку. Если это правильная запись двоичного числа, вывести сообщение «Да», иначе вывести сообщение «Нет».

Пример:

```
Введите число:          Введите число:
1010100                 abcd10
Да.                     Нет.
```

Задачи

«5»: Ввести с клавиатуры символьную строку и составить новую строку, удалив из исходной все пробелы.

Пример:

```
Введите строку:
Вася пошел гулять.
Васяпошелгулять.
```

Задачи

«6»: Ввести с клавиатуры символьную строку, и составить новую строку, удалив из исходной все заглавные буквы.

Пример:

```
Введите строку:
ЫтВЫоЯСтем.
тотем.
```

Функция range

Для повторения цикла некоторое заданное число раз n можно использовать цикл for вместе с функцией range:

```
for i in range(n):          # i пробегает значения
    Тело цикла              # от 0 до n-1
```

В качестве n может использоваться числовая константа, переменная или произвольное арифметическое выражение (например, 2 ** 10). Если значение n равно нулю или отрицательное, то тело цикла не выполнится ни разу.

Можно задать цикл таким образом:

```
for i in range(a, b):       # i пробегает значения
    Тело цикла              # от a до b-1
```

Если же a≥b, то цикл не будет выполнен ни разу.

Цикл с переменной

Задача. Вывести 10 раз слово «Привет!».

? Можно ли сделать с циклом «пока»?

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

Цикл с переменной:

```
for i in range(10):
    print("Привет!")
```

в диапазоне [0,**10**)

! Не включая **10**!

```
range(10) → 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
```

Цикл с переменной

Задача. Вывести все степени двойки от 2^1 до 2^{10}.

? Как сделать с циклом «пока»?

```
k = 1
while k <= 10
    print ( 2**k )
    k += 1
```

возведение в степень

Цикл с переменной:

```
for k in range(1,11):
    print ( 2**k )
```

в диапазоне [1,**11**)

! Не включая **11**!

```
range(1,11) → 1, 2, 3, 4, 5, 6, 7, 8, 9,
10
```

Цикл с переменной: другой шаг

10,9,8,7,6,5,4,3,2,1

шаг

```
for k in range(10,0,-1):
    print ( k**2 )
```

```
100
81
64
49
36
25
16
9
4
1
```

? Что получится?

1,3,5,7,9

```
for k in range(1,11,2):
    print ( k**2 )
```

```
1
9
25
49
81
```

Сколько раз выполняется цикл?

```
a = 1
for k in range( 3): a += 1
```

a = 4

```
a = 1
for k in range( 3,1): a += 1
```

a = 1

```
a = 1
for k in range( 1,3,-1): a += 1
```

a = 1

```
a = 1
for k in range( 3,1,-1): a += 1
```

a = 3

Самостоятельно

1. Даны два целых числа *A* и *B* (*A* < *B*). Вывести в порядке возрастания все целые числа, расположенные между *A* и *B* (включая сами числа *A* и *B*), а также количество *N* этих чисел.

Ввод: 4 10

Вывод: 4 5 6 7 8 9 10

7

2. Даны два целых числа *A* и *B* (*A* < *B*). Вывести в порядке убывания все целые числа, расположенные между *A* и *B* (не включая числа *A* и *B*), а также количество *N* этих чисел.

Ввод: 15 10

Вывод: 14 13 12 11

4

3. Даны два целых числа *A* и *B* (*A* < *B*). Найти сумму квадратов всех целых чисел от *A* до *B* включительно.

Ввод: -1 3

Вывод: 15

Задачи

«3»: Ипполит задумал трёхзначное число, которое при делении на 15 даёт в остатке 11, а при делении на 11 даёт в остатке 9. Найдите все такие числа.

«4»: Вводится натуральное число N. Программа должна найти **факториал** (обозначается как **N!**) – произведение всех натуральных чисел от 1 до N. Например, 5! = 1 · 2 · 3 · 4 · 5 = 120.

Пример:

```
Введите число:
5
5!=120.
```

Задачи

«5»: Натуральное число называется **числом Армстронга**, если сумма цифр числа, возведенных в N-ную степень (где N – количество цифр в числе) равна самому числу. Например, $153 = 1^3 + 5^3 + 3^3$. Найдите все трёхзначные Армстронга.

Задачи

«6»: Дано 5 целых чисел. Вычислите их сумму. Напишите программу, использующую наименьшее число переменных.

Пример:

```
Введите числа:
5
5
1
3
1
Сумма: 15
```

Задачи

«7»: По данному натуральному n ≤ 9 выведите лесенку из n ступенек, i-я ступенька состоит из чисел от 1 до i без пробелов.

Пример:

```
Введите n:
3
1
12
123
```

Пример:

```
Введите n:
2
1
12
```

Задачи

«8»: Вывести на экран все чётные значения в диапазоне от 1 до 497;

Задачи

«9*»: Для настольной игры используются карточки с номерами от 1 до N. Одна карточка потерялась. Найдите ее, зная номера оставшихся карточек.

Дано число N, далее N − 1 номер оставшихся карточек (различные числа от 1 до N). Программа должна вывести номер потерянной карточки.

Для самых умных: массивами и аналогичными структурами данных пользоваться нельзя.

Пример:

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

Пример:

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

Пример:

```
3
1
3
2
```