ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ PYTHON

1. Общие сведения о языке Python.
2. Вывод информации на экран.
3. Инструкция присваивания. Ввод данных.
4. Условные операторы.
5. Циклы.

Вопрос 1. ОБЩИЕ СВЕДЕНИЯ О ЯЗЫКЕ PYTHON

Краткая история языка Python

Python – высокоуровневый язык программирования. Используется для создания программ различного назначения.

Первая версия языка Python вышла в **1991 г**.

Автор языка – нидерландский программист **Гвидо ван Россум.**

Название Python язык программирования получил в честь комик-группы из Великобритании – Монти Пайтон (Monty Python).

Версии языка программирования: Python 2 (версия 2.0 вышла в 2000 г.) и Python 3 (версия 3.0 вышла в 2008 г.).

СТРУКТУРА ПРОГРАММЫ

Программа состоит из команд, которые называются **инструкциями**.

Правила записи инструкций

1. Конец строки является концом инструкции.

2. Можно записать несколько инструкций в одной строке, разделяя их точкой с запятой (;).

3. Инструкции должны быть записаны с одним и тем же отступом.

Составные инструкции записываются в несколько строк:

```
инструкция1:
    инструкция2
    инструкция3
    …
```

«Внутренние» инструкции записываются с одним и тем же отступом относительно «наружной» инструкции.

Отступ можно сделать, нажав клавишу **Tab** или **4 пробела**.

В конце первой строки составной инструкции указывается символ «**:**».

Алфавит языка Python

Алфавит языка Python (набор допустимых символов) включает:

- ✓ буквы латинского алфавита;
- ✓ цифры;
- ✓ специальные знаки (препинания, арифметические знаки и другие).

Русские буквы могут использоваться при выводе текста на экран и в комментариях к программе.

!!! *Заглавные и строчные буквы различаются.*

Например, переменные с именами dlina и Dlina – две разные переменные.

Служебные слова – цепочки символов, имеющие фиксированное смысловое значение.

Данные в программе представлены константами и переменными.

Константы – величины, не изменяющие своего значения при выполнении программы.

Переменные могут изменять свое значение при выполнении программы.

Переменная *– это ссылка на область памяти, где хранятся данные*.

Переменная имеет **имя**, **тип** и **значение.**

Имя переменной – любая, отличная от служебных слов последовательность латинских букв, цифр и символа подчеркивания "_". Имя не может начинаться с цифры, не может содержать пробел.

Например,

n, n1, massa, massa_tela – правильно;

1n, масса, massa tela – неправильно.

Тип переменной определяет:

- ✓область допустимых значений переменной;
- ✓множество допустимых операций с переменной;
- ✓объем памяти для хранения значения данной переменной.

Основные типы данных в языке Python

Тип	Обозначение
Целый	**int**
Вещественный	**float**
Строковый	**str**
Логический	**bool**

*Целая часть числа от дробной части отделяется **точкой**.*

Строковое значение заключается в двойные или одинарные кавычки.

Тип переменной определяется автоматически в момент присваивания ей значения и может изменяться по ходу выполнения программы.

*«**Python —язык программирования с динамической типизацией, т. е. в ходе выполнения программы одна и та же переменная может хранить значения различных типов**»*

ВЫРАЖЕНИЯ И ОПЕРАЦИИ

Простые выражения − *переменные* и *константы*.

Сложные выражения строятся из простых с помощью *операций, функций* и *скобок.*

АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ

сложение	+
вычитание	−
умножение	*
деление	/
целочисленное деление	//
остаток от деления	%
возведение в степень	**

ОПЕРАЦИИ СРАВНЕНИЯ

равно	= =
не равно	!=
больше	>
меньше	<
больше или равно	>=
меньше или равно	<=

Форма записи выражений – линейная (в одну строку).

Например:

$$\frac{(a+b)h}{2}$$

```
1   2  3
(a+b)*h/2
```

$$v+\frac{at^2}{2}$$

```
4   2    1  3
v+a*t**2/2
```

ФУНКЦИИ

Функции имеют определенное *имя* и один или несколько *аргументов* в скобках.

Некоторые встроенные функции

Функция	Назначение
`abs(x)`	Модуль числа x
`int(x)`	Преобразование *x* в целое число
`round(x, n)`	Округляет число *x* до *n* знаков после запятой

Встроенные функции доступны без дополнительных действий.

Большинство функций языка находятся в стандартной библиотеке.

Функции разбиты на группы по назначению.

Каждая группа записана в отдельном файле, который называется **модулем**.

Подключение модуля осуществляется командой **`import`**.

Пример 1. *Подключаем все функции из модуля* ***math***

```
import math
```

Пример 2. *Подключаем из модуля math функцию*

```
from math import sin
```

Стандартные функции модуля `math`

Функция	Назначение
`sqrt(x)`	Квадратный корень из X
`sin(x)`	Синус X (X указывается в радианах)
`cos(x)`	Косинус X (X указывается в радианах)
`tan(X)`	Тангенс X (X указывается в радианах)
`exp(X)`	Экспонента числа X
`log(X)`	Натуральный логарифм X
`log10(X)`	Десятичный логарифм X
`log(X, n)`	Логарифм X по основанию n
`pi`	Выдаётся число π

Вопрос 2.
ВЫВОД ИНФОРМАЦИИ НА ЭКРАН

Для вывода информации на экран используется инструкция **`print( )`**

В списке вывода (в скобках) может быть:

1. Текст в кавычках	`print('Привет!')` `print("Привет!")`
2. Число	`print(-2)`
3. Имя переменной	`print(x)`
4. Выражение	`print(a * b)`
5. Несколько выражений (в том числе разного типа) через запятую	`print ("Масса равна", m, "кг")`

По умолчанию между значениями списка выводится один пробел.
В качестве разделителя можно использовать другой символ, указав его как параметр **sep** («separator»):

```
print(<список вывода>, sep = ",")
```

Каждая новая инструкция **print()** выводит значения на следующей строке. Чтобы исключить это, используется параметр **end**.:

```
print(<список вывода>, end = "")
```

Примеры

Вариант вывода	Инструкция	На экране
По умолчанию	`print (1, 20, 300)`	`1 20 300`
Без разделителя	`print (1, 20, 300, sep="")`	`120300`
Через запятую и пробел	`print (1, 20, 300, sep=", ")`	`1, 20, 300`
Без перехода на новую строку	`print (1, end="")` `print (20)`	`120`

Вопрос 3.
ИНСТРУКЦИЯ ПРИСВАИВАНИЯ. ВВОД ДАННЫХ

ИНСТРУКЦИЯ ПРИСВАИВАНИЯ

Инструкция присваивания позволяет изменить (или задать впервые) значение переменной.

```
<имя переменной> = <выражение>
```

Например:

```
a = 5
b = a
c = c+1  (или c+=1)
a = a*b  (или a*=b)
```

В Python допускается множественное присваивание

Например,

Запись оператора:	*Равносильная запись:*
`a, b = 0, 1` (количество переменных слева = количеству значений справа)	`a = 0` `b = 1`
`a = b = 0`	`a = 0` `b = 0`

ВВОД ДАННЫХ

Для ввода значений переменных с клавиатуры используется инструкция **input ( )**:

```
<имя_переменной> = input( )
```

В скобках `input()` можно записать сообщение-подсказку.

Например,

```
a = input ("Введите число: ")
```

Пример:

Программа на Python

```
a = input ('Введите целое число:')
b = input ('Введите целое число:')
c = a + b
print (c)
```

Результат вывода на экран

```
Введите целое число: 2
Введите целое число: 3
23
```

Важно!!! `input` всегда воспринимает введенные значения как строку.

Если вводится не строка, а число, необходимо выполнить преобразование типов с помощью функций `int` (для целых) и `float` (для вещественных).

Правильно:

```
a = int (input("Введите целое число: "))
b = int (input("Введите целое число: "))
```

Комментарии в программе

Комментарии - тексты, помогающие читающему код программы понять ее особенности.

Комментариями считается любой текст после символа **#**.

При выполнении программы комментарии игнорируются.

Пример:

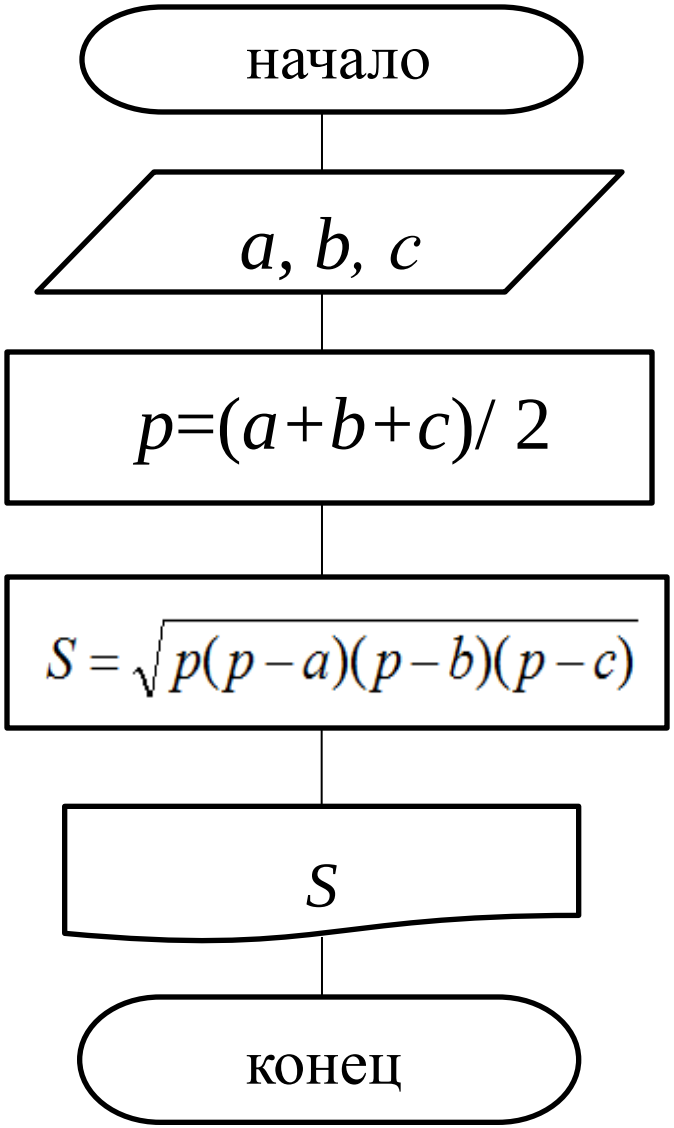
```
# Длина окружности и площадь круга
r = float(input("Введите радиус: "))
c = 2*3.14*r                    # длина окружности
s = 3.14*r**2                   # площадь круга
print ("c=", c)
print ("s=", s)
```

Задача 1

Составить программу для вычисления площади треугольника по известным длинам его сторон.

Формула Герона: $$S = \sqrt{p \cdot (p - a) \cdot (p - b) \cdot (p - c)}$$

Блок-схема алгоритма



```
# Площадь треугольника по формуле Герона
print ("Введите длины сторон треугольника: ")
a = float(input("a="))
b = float(input("b="))
c = float(input("c="))
p = (a+b+c)/2
from math import sqrt                  # подключаем модуль math
s = sqrt(p*(p-a)*(p-b)*(p-c))          # формула Герона
print ("Площадь треугольника=", s)
```

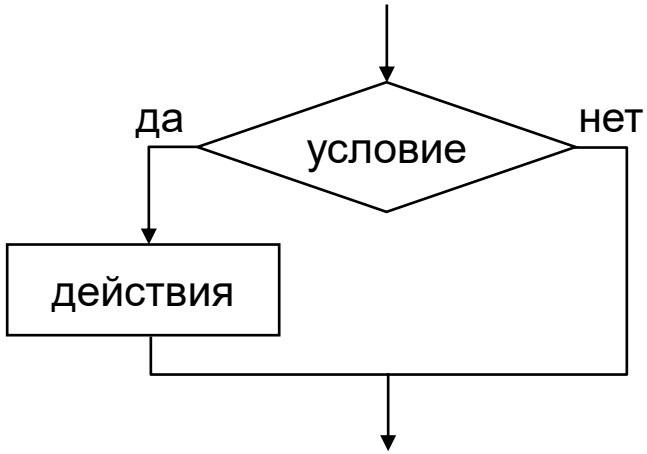
Вопрос 4.
УСЛОВНЫЕ ОПЕРАТОРЫ

ФОРМЫ УСЛОВНОГО ОПЕРАТОРА

1. Неполная форма условного оператора:

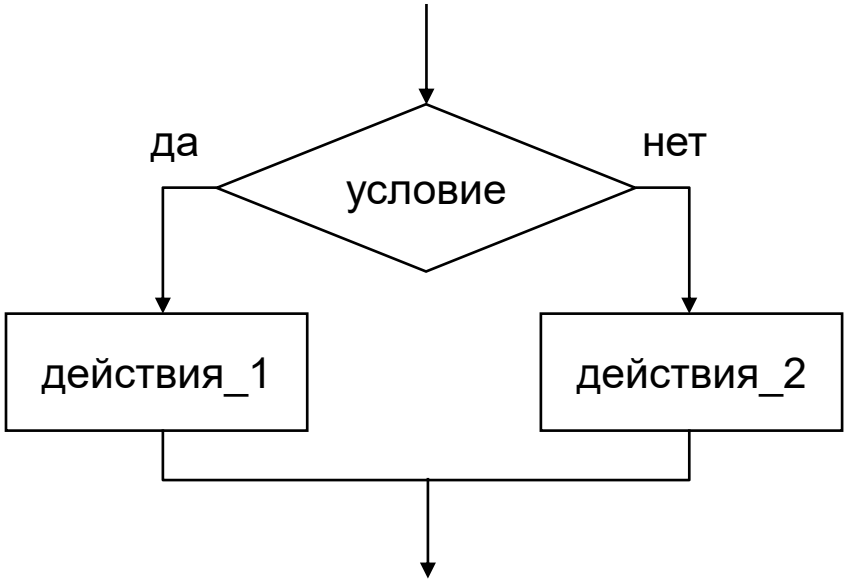
```
if <логическое выражение> :
    <действия, выполняемые, когда логическое
    выражение True>
```

Действия после двоеточия выполняются, если логическое выражение истинно.
В противном случае – переход к следующему оператору программы.



2. Полная форма условного оператора:

```
if  <логическое выражение>:
        <действия, выполняемые, когда логическое
        выражение True>
else :
        <действия, выполняемые, когда логическое
        выражение False>
```



3. Форма с дополнительным оператором elif

```
if <первое логическое выражение>:
    <действия, выполняемые, если логическое
    выражение True>
elif <второе логическое выражение>:
    <действия, выполняемые, если второе
    логическое выражение True>
elif <третье логическое выражение>:
    <действия, выполняемые, если третье
     логическое выражение True>
. . .
else :
    <действия, выполняемые, если ни одно
    из логических выражений не принимает значение
    True>
```

Как только некоторое условие окажется истинным, соответствующий блок выполняется, и дальнейшие условия не проверяются.

Важно!

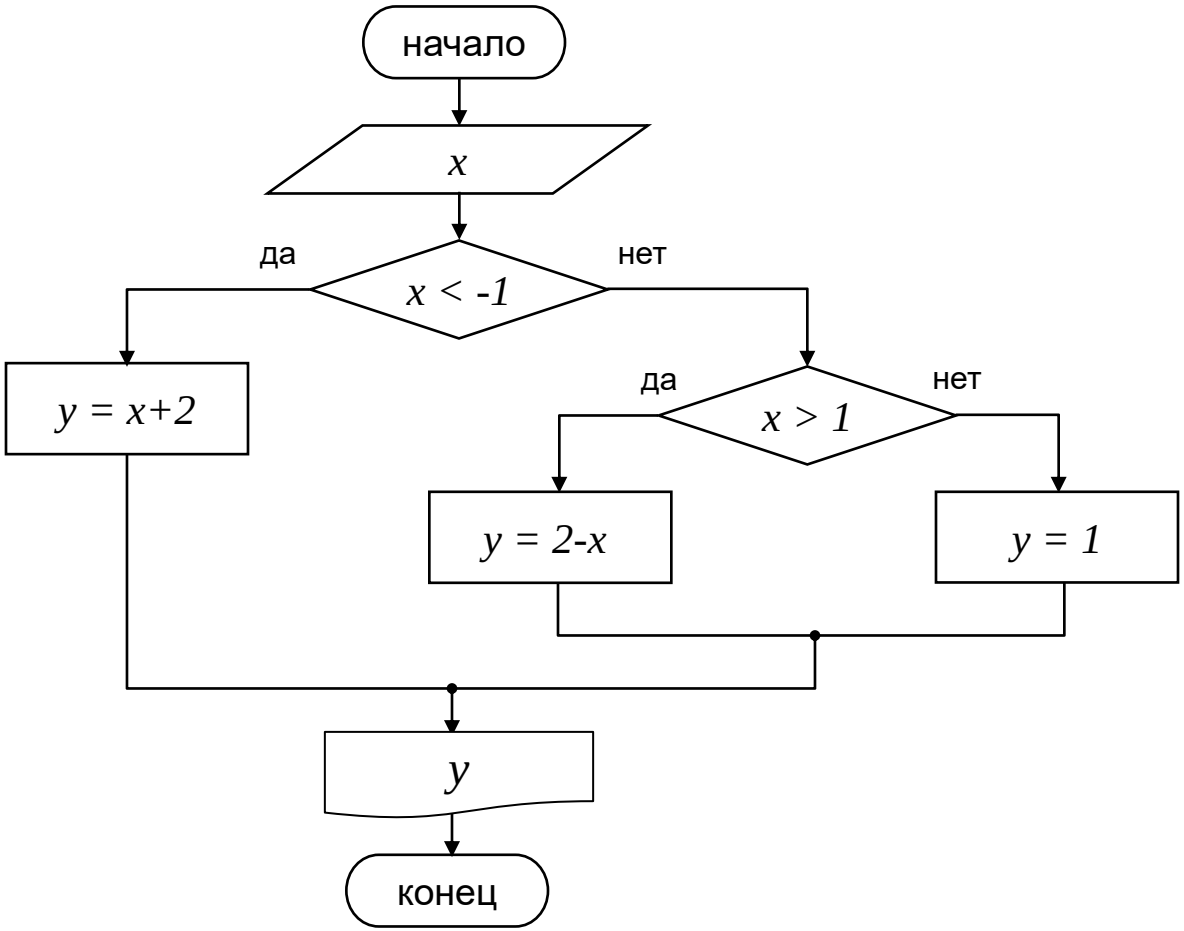
После двоеточия в конструкциях типа `if, else, elif` всегда идёт блок, выделенный отступом вправо.

То, что выделено отступами - *тело оператора*,

то, что записано до двоеточия, называется *заголовком*.

Задача 2. Найти значение функции для любого значения аргумента:

$$y = \begin{cases} x + 2, & при\ x < -1 \\ 1, & при\ -1 \le x \le 1 \\ 2 - x, & при\ x > 1 \end{cases}$$



$$y = \begin{cases} x+2, & при\ x < -1 \\ 1, & при\ -1 \le x \le 1 \\ 2-x, & при\ x > 1 \end{cases}$$

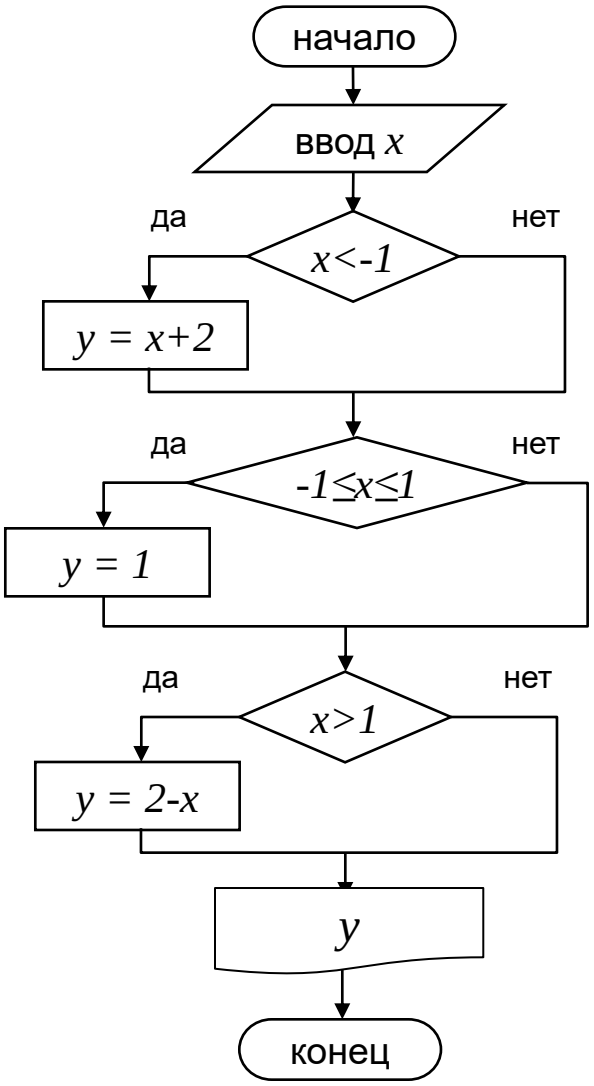
```
# Значение функции
x = float(input("Введите x: "))
if x < -1:
      y = x+2
elif x > 1:
      y = 2-x
else:
      y = 1
print ("y=", y)
```

Примечание: в языке Python разрешены двойные неравенства, например:

```
if -1 <= x <= 1:
      y = 1
```

Задача 2 (2 способ)

Найти значение функции для любого значения аргумента.



$$y = \begin{cases} x+2, & при\ x < -1 \\ 1, & при\ -1 \le x \le 1 \\ 2-x, & при\ x > 1 \end{cases}$$

```
# Значение функции
x = float(input("Введите x: "))
if x < -1:
        y = x+2
if x >= -1 and x <= 1:
        y = 1
if x > 1:
        y = 2-x
print ("y=", y)
```

СОСТАВНЫЕ УСЛОВИЯ

В инструкции `if` возможно использование «сложных условий», записанных с помощью логических операций:

- **логическое умножение `and` («И»)**

```
x>3 and y==4
```

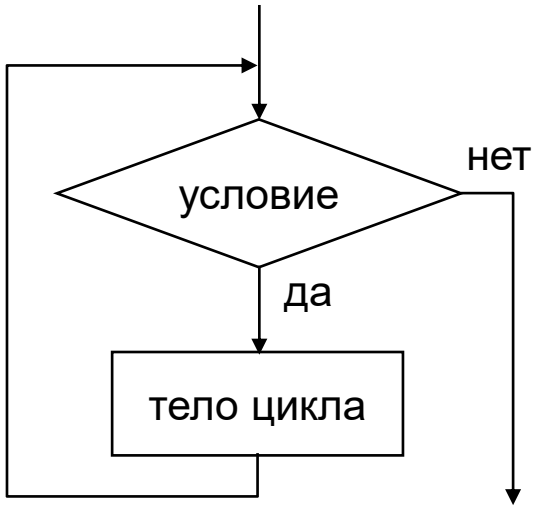
- **логическое сложение `or` («ИЛИ»).**

```
x<3 or y=!4
```

Вопрос 5. ЦИКЛЫ

1. Инструкция while

(цикл с предусловием)




```
while <условие>:
        <операторы_тела_цикла>
```

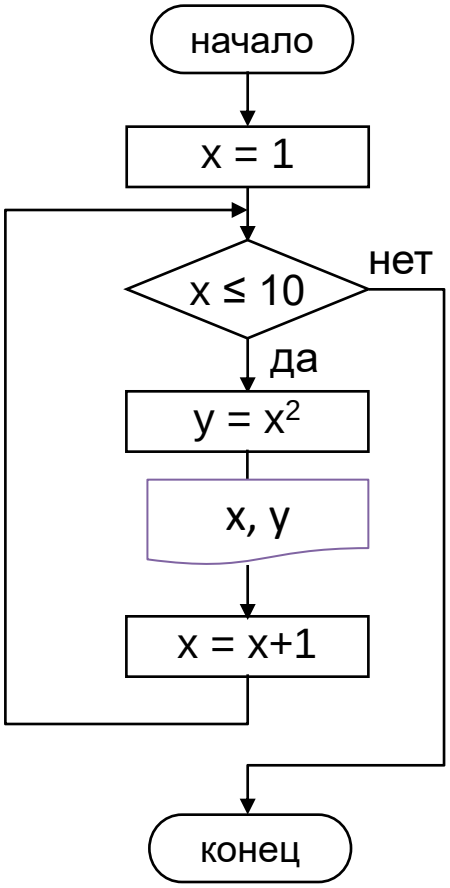
Пока **<условие>** **истинно**,
<операторы_тела_цикла> **повторяются**.

Если **<условие>** **ложно**, то управление передается следующему после цикла оператору.

Если **<условие>** изначально ложно, операторы тела цикла не выполнятся ни разу.

Если **<условие>** никогда не станет ложным, то программа «зациклится».

Задача 3. Напечатать таблицу значений функции $y=x^2$ для x = 1; 2; 3; …; 10.




```
print("Таблица:")
x = 1                    # начальное значение x
while x<=10:             # пока x<=10 повторять:
    y = x**2             # вычисление функции
    print(x, y, sep="    ")
    x = x+1              # следующее значение x
```

2. Инструкция for

```
for <параметр> in range(<диапазон>):
    <операторы_тела_цикла>
```

Операторы_тела_цикла повторяются *фиксированное число раз* для каждого значения параметра.
Параметр – переменная целого типа.

Функция **range**(<диапазон>) задаёт количество повторений тела цикла и содержит от одного до трёх чисел.

- одно число **(k)** – параметр цикла изменяется *от **0** до **k-1** с шагом **1***.
- два числа **(n, k)** – параметр цикла изменяется *от **n** до **k-1** с шагом **1***.
- три числа **(n, k, s)** – параметр цикла изменяется *от **n** до **k-1** с шагом **s***.

Примеры записи оператора цикла `for`

```
for i in range(10):
    print(i, end=" ")
```

в диапазоне [0,**10**), не включая **10**

```
0 1 2 3 4 5 6 7 8 9
```

```
for i in range(1,10):
    print(i, end=" ")
```

```
1 2 3 4 5 6 7 8 9
```

```
for i in range(1,10,2):
    print(i, end=" ")
```

```
1 3 5 7 9
```

```
for i in range(9,0,-2):
    print(i, end=" ")
```

```
9 7 5 3 1
```

Задача 3 (цикл с `for`)

Напечатать таблицу значений функции $y=x^2$ для **x=1; 2; 3; …; 10**.

```
for x in range (1,11):
    y = x**2
    print(x, y, sep="    ")
```