Семинар 3

Введение в программирование на Python

Папулин С.Ю.
papulin_hse@mail.ru

2015

Семинар 2

1. Существующие типы данных (int, float, string)
2. Переменные в Python (присваивание значений)
3. Операции над значениями переменных
4. Блок-схемы алгоритмов
5. Условные выражения в Python (if)

План семинара 3

1. Циклы в Python
2. Функция range()
3. Другие полезные возможности
4. Примеры задач
5. Самостоятельная работа

Семинар 2. Присваивание значений

```
#Assignment
x=6
print("x = " + str(x))
#Reassignment
x=4
print("x = " + str(x))
#Assignment
y=x
print("y = " + str(x))
#Update
x=x+5
print("x = " + str(x))
#Update
x=x+y
print("x = " + str(x))
```

1. x = 6
2. x = 4
3. y = 4
4. x = 9
5. x = 13

Циклы в Python

Для повторения одного набора операций (тела цикла) используются *циклы*. *Итерация* – один проходов в цикле (одно повторение).

Пример

На предприятии 500 сотрудников. Руководство решает выписать премию каждому сотруднику в размере $1000, если его зарплата меньше $1000000.

Для каждого из 500 сотрудников

Сотрудник_i
Если Зарплата(Сотрудник_i) меньше $1000000,
то Зарплата(Сотрудник_i) = Зарплата(Сотрудник_i) + $1000

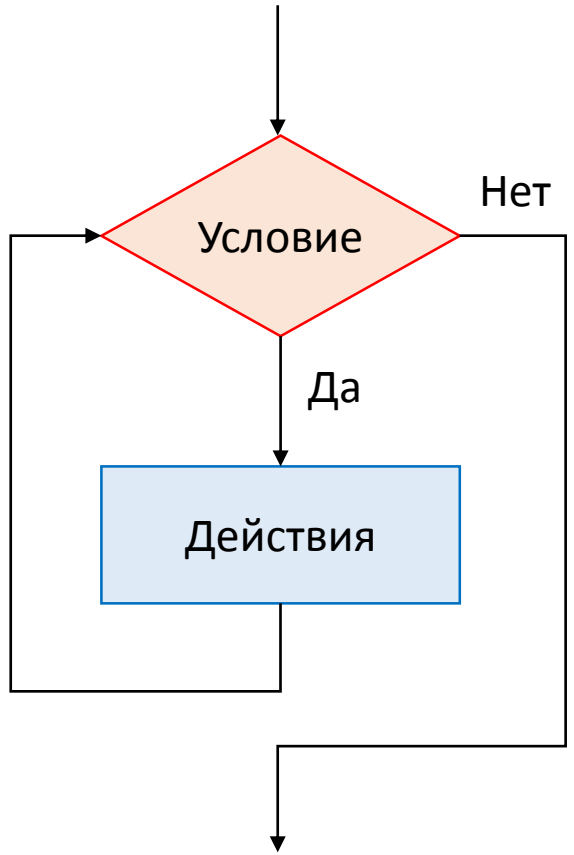
В Python’е есть два вида циклов – WHILE и FOR.

Цикл WHILE

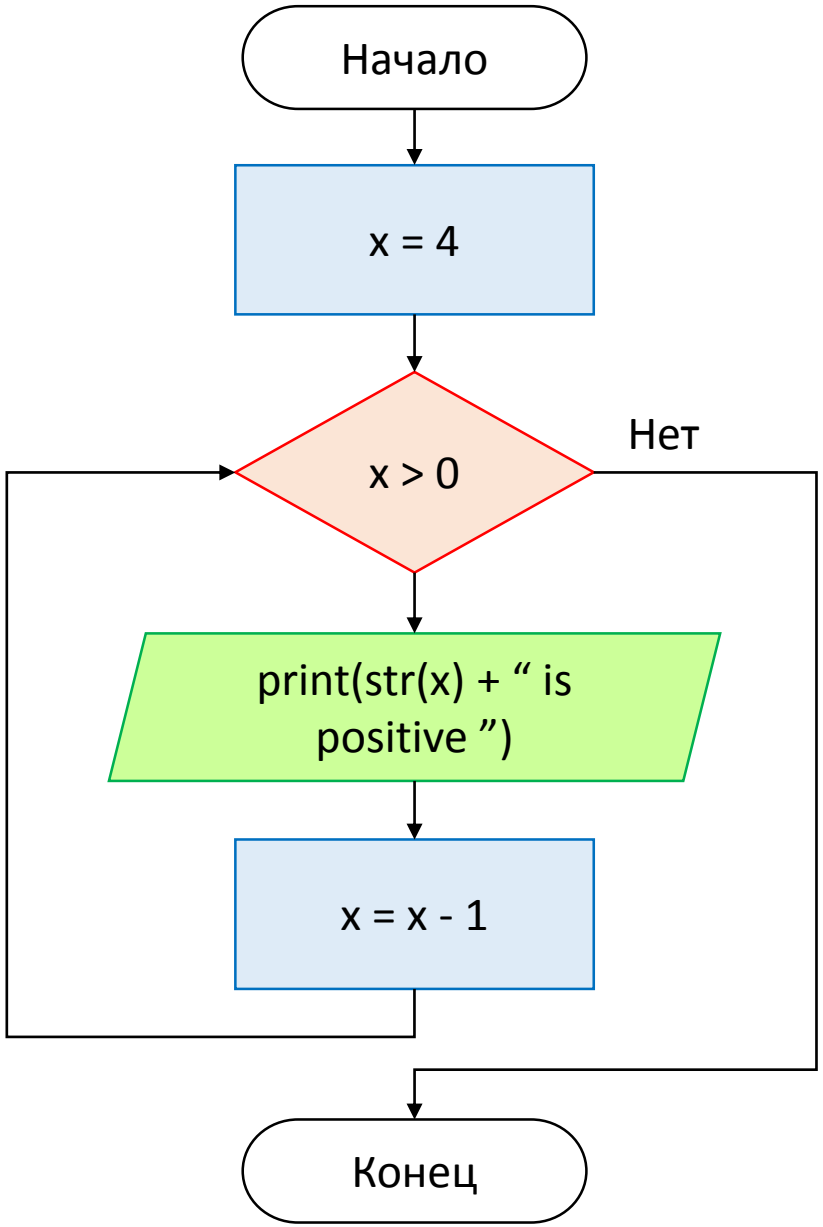
Цикл WHILE выполняет действия, определенные в теле цикла, до тех пор, пока указанное условие истинно.

while *условие*:
    действия

Условие = {TRUE, FALSE}



Пример цикла WHILE



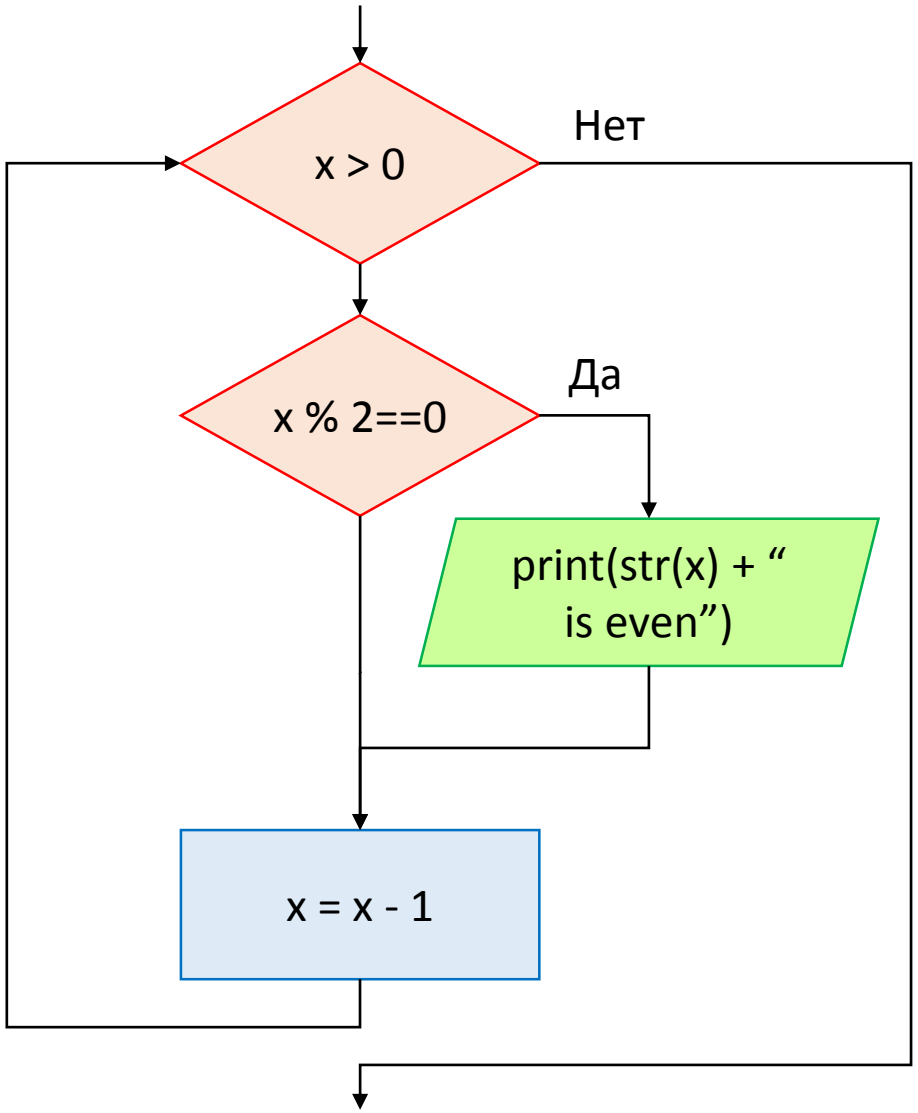
Python:

```
x = 4
while(x > 0):
    print(str(x) + " is positive")
    x = x - 1
```

Результат

“4 is positive”
“3 is positive”
“2 is positive”
“1 is positive”

Пример цикла WHILE



Python:

```
x = 4
while(x > 0):
    if(x%2==0):
        print(str(x) + " is even")
    x = x - 1
```

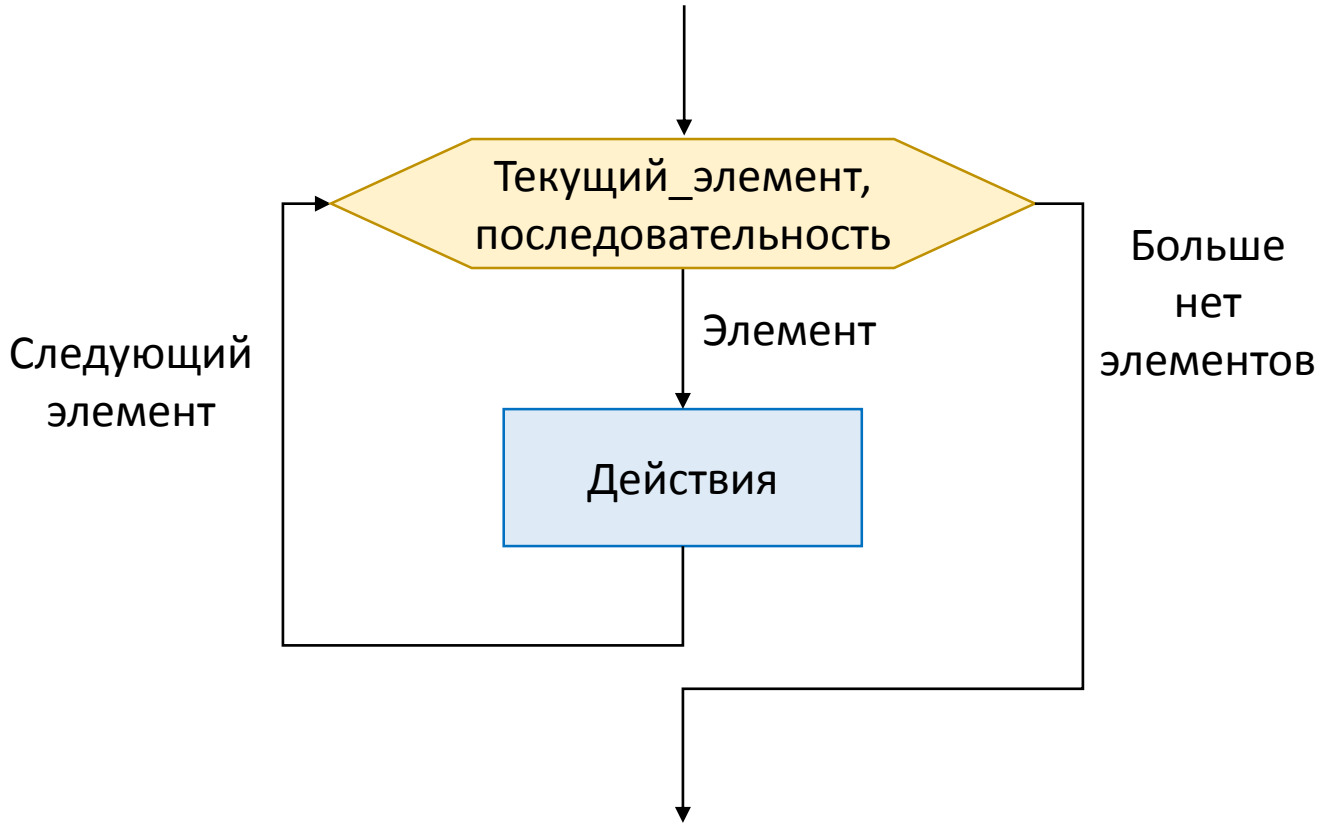
Результат

“4 is even”
“2 is even”

Цикл FOR

Цикл FOR выполняет действия, определенные в теле цикла, последовательно для каждого элемента из указанной последовательности.

for *текущий_элемент* in *последовательность элементов*:
    действия



Последовательность элементов (поддержка итераций)

Строка (string). Строка состоит из множества символов в определенной последовательности.

```
s = "hello"   print(s[0])
```
 “h”

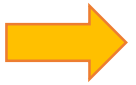
Список (list). Множество элементов, каждый из который имеет значение и позицию в списке.

```
l = [1,2,3,4] print(l[0])
```
 “1”

Кортеж (tuple). Множество неизменяемых элементов, каждый из который имеет значение и позицию в кортеже.

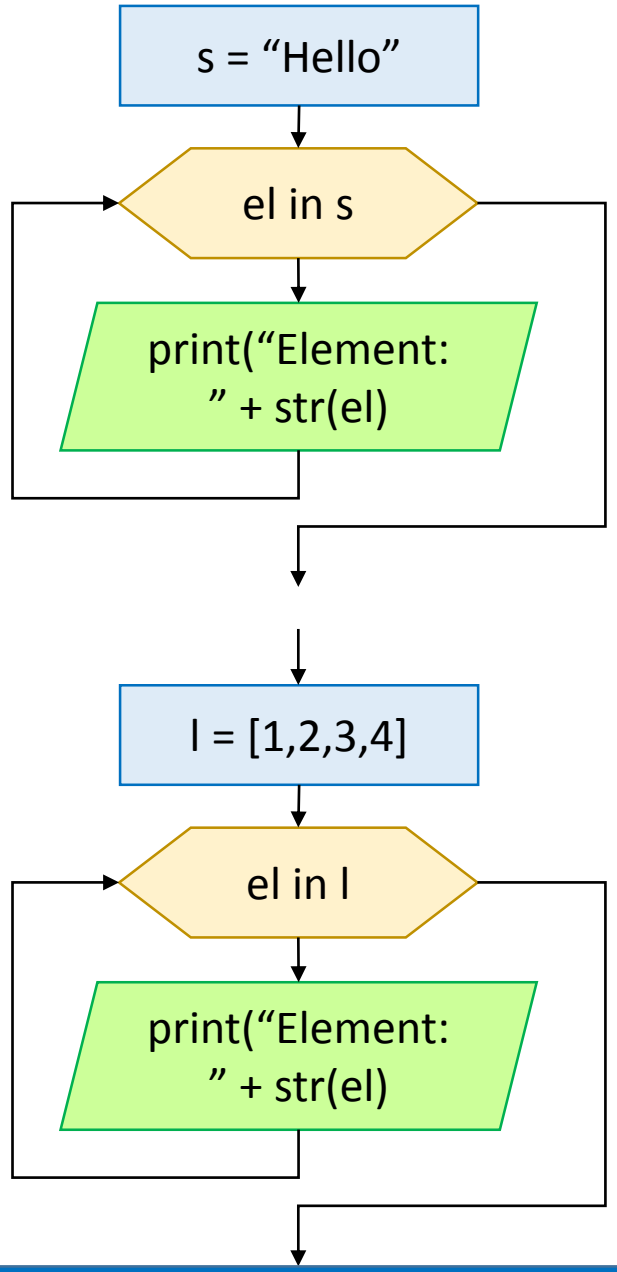
```
t = (1,2,3,4) print(t[0])
```
 “1”

Словарь (dictionary). Множество пар вида ключ-значение.

```
d = {1:"a",2:"b",3:"c",4:"d"}  print(d[1])
```
 “a”

Тип представления последовательности RANGE

Пример цикла FOR



Строка (string)

```
s = "hello"
for el in s:
    print("Element: " + str(el))
```

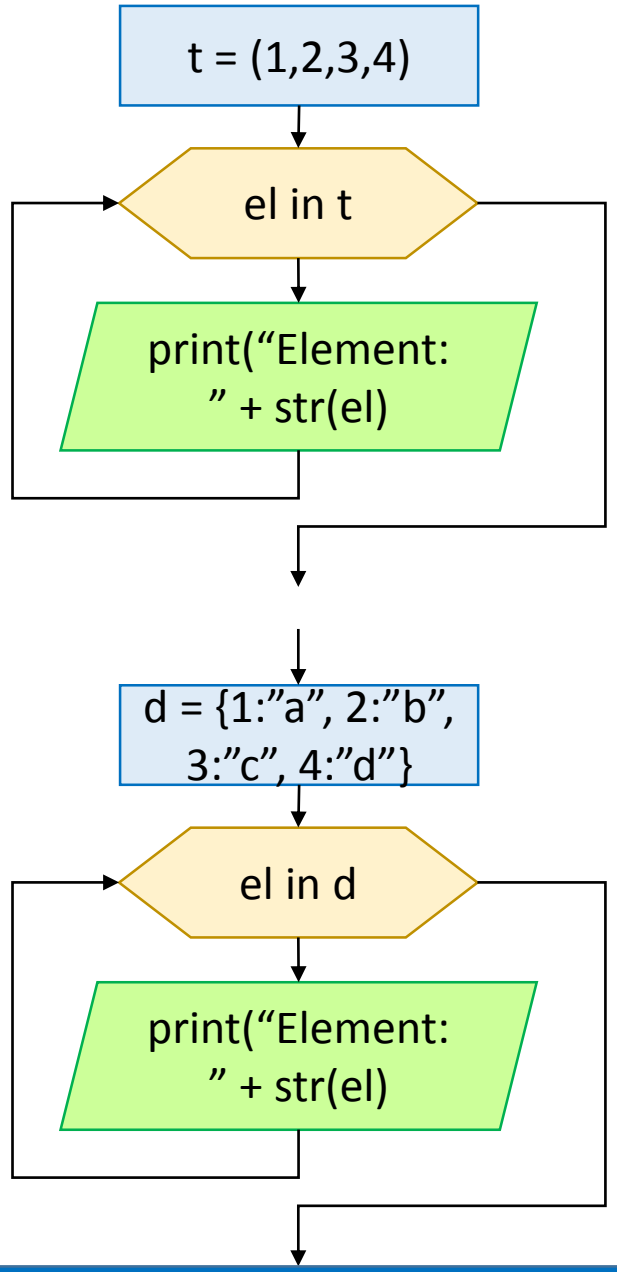
Element: h
Element: e
Element: l
Element: l
Element: o

Список (list)

```
l = [1,2,3,4]
for el in l:
    print("Element: " + str(el))
```

Element: 1
Element: 2
Element: 3
Element: 4

Пример цикла FOR



Кортеж (tuple)

```
t = (1,2,3,4)
for el in t:
    print("Element: " + str(el))
```

```
Element: 1
Element: 2
Element: 3
Element: 4
```

Словарь (dictionary)

```
d = {1:"a",2:"b",3:"c",4:"d"}
for el in d.items():
    print("Element: " + str(el))
```

```
Element: (1, 'a')
Element: (2, 'b')
Element: (3, 'c')
Element: (4, 'd')
```

RANGE

1. **range**(*stop*) используется для формирования целых чисел от 0 до *stop* с шагом 1:

```
list(range(10))
```

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

2. **range**(*start*, *stop*) используется для формирования целых чисел от *start* до *stop* с шагом 1:

```
list(range(3,10))
```

[3, 4, 5, 6, 7, 8, 9]

3. **range**(*start*, *stop*, *step*) используется для формирования целых чисел от *start* до *stop* с шагом *step*:

```
list(range(0, 20, 4))
```

[0, 4, 8, 12, 16]

```
list(range(10, -4, -1))
```

[10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, -1, -2, -3]

Пример цикла FOR + range()

```
x = 0
for y in range(1, 10):
    x=x+y
    print("Sum of numbers from 1 to %i is %i" % (y,x))
```

Sum of numbers from 1 to 1 is 1
Sum of numbers from 1 to 2 is 3
Sum of numbers from 1 to 3 is 6
Sum of numbers from 1 to 4 is 10
Sum of numbers from 1 to 5 is 15
Sum of numbers from 1 to 6 is 21
Sum of numbers from 1 to 7 is 28
Sum of numbers from 1 to 8 is 36
Sum of numbers from 1 to 9 is 45

Область видимости переменных

```
for y in range(1, 10):
    x = x + y
    print("Sum of numbers from 1 to %i is %i" % (y, x))
```

NameError: name 'x' is not defined

Область видимости переменных

```
for y in range(1, 10):
    x = 0
    x = x + y
    print("Sum of numbers from 1 to %i is %i" % (y,x))
```

```
Sum of numbers from 1 to 1 is 1
Sum of numbers from 1 to 2 is 2
Sum of numbers from 1 to 3 is 3
Sum of numbers from 1 to 4 is 4
Sum of numbers from 1 to 5 is 5
Sum of numbers from 1 to 6 is 6
Sum of numbers from 1 to 7 is 7
Sum of numbers from 1 to 8 is 8
Sum of numbers from 1 to 9 is 9
```

Область видимости переменных

```
for y in range(1, 10):
    x = 0
    x = x + y
    print("Sum of numbers from 1 to %i is %i" % (y,x))
print("Variable x within the loop is equal to " + str(x))
```

```
Sum of numbers from 1 to 1 is 1
Sum of numbers from 1 to 2 is 2
Sum of numbers from 1 to 3 is 3
Sum of numbers from 1 to 4 is 4
Sum of numbers from 1 to 5 is 5
Sum of numbers from 1 to 6 is 6
Sum of numbers from 1 to 7 is 7
Sum of numbers from 1 to 8 is 8
Sum of numbers from 1 to 9 is 9
Variable x within the loop is equal to 9
```

Область видимости переменных

```
x = 0
for y in range(1, 10):
    print("y = " + str(y))
    y = 0
    x = x + y
    print("Sum of numbers from 1 to %i is %i" % (y,x))
```

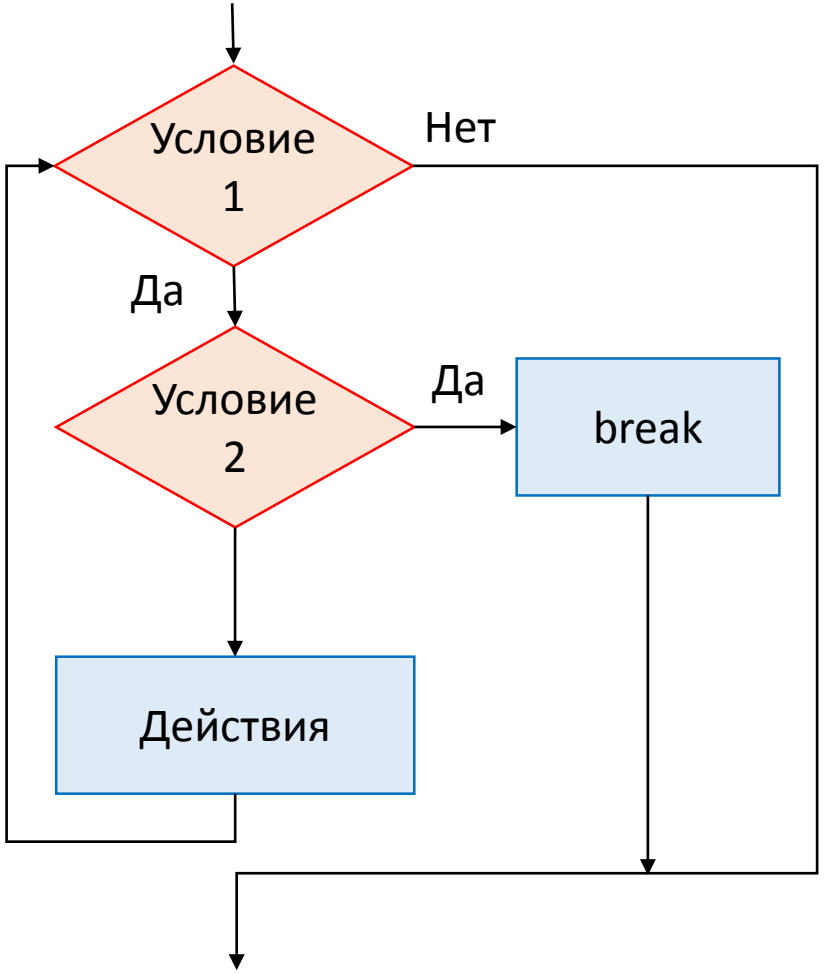
y = 1 | Sum of numbers from 1 to 0 is 0
y = 2 | Sum of numbers from 1 to 0 is 0
y = 3 | Sum of numbers from 1 to 0 is 0
y = 4 | Sum of numbers from 1 to 0 is 0
y = 5 | Sum of numbers from 1 to 0 is 0
y = 6 | Sum of numbers from 1 to 0 is 0
y = 7 | Sum of numbers from 1 to 0 is 0
y = 8 | Sum of numbers from 1 to 0 is 0
y = 9 | Sum of numbers from 1 to 0 is 0

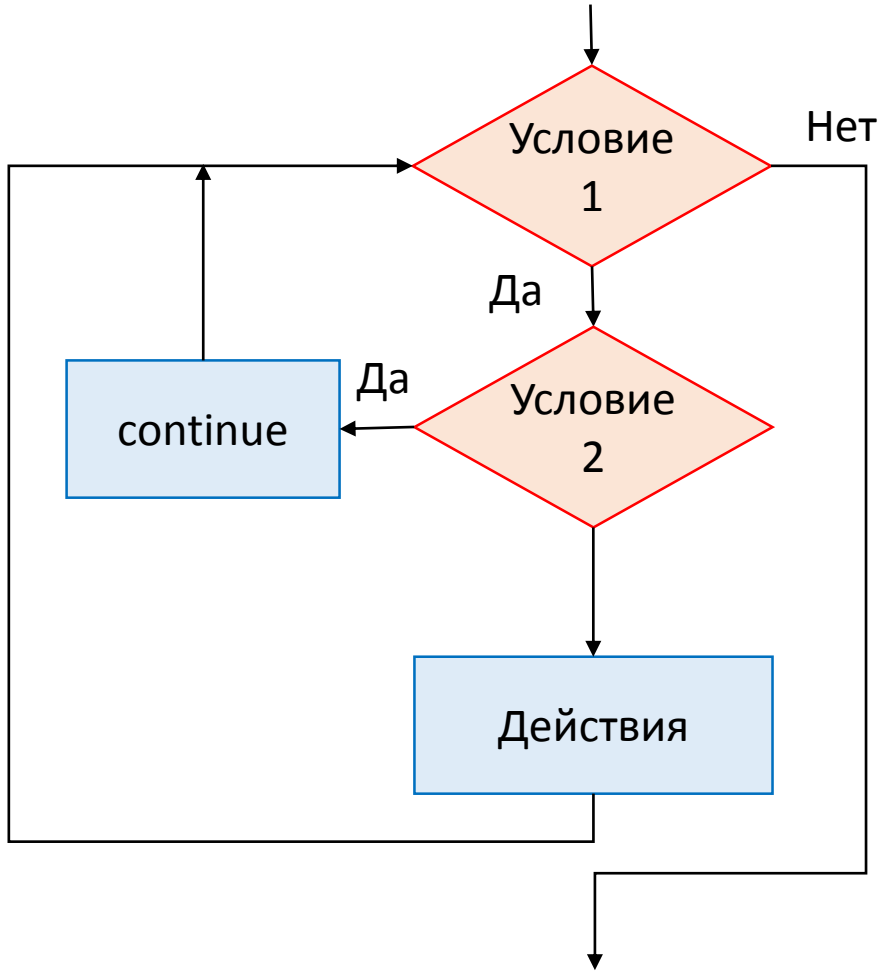
Прерывание (break) и продолжение (continue) цикла

Если необходимо преждевременной покинуть цикл, используется **BREAK**.

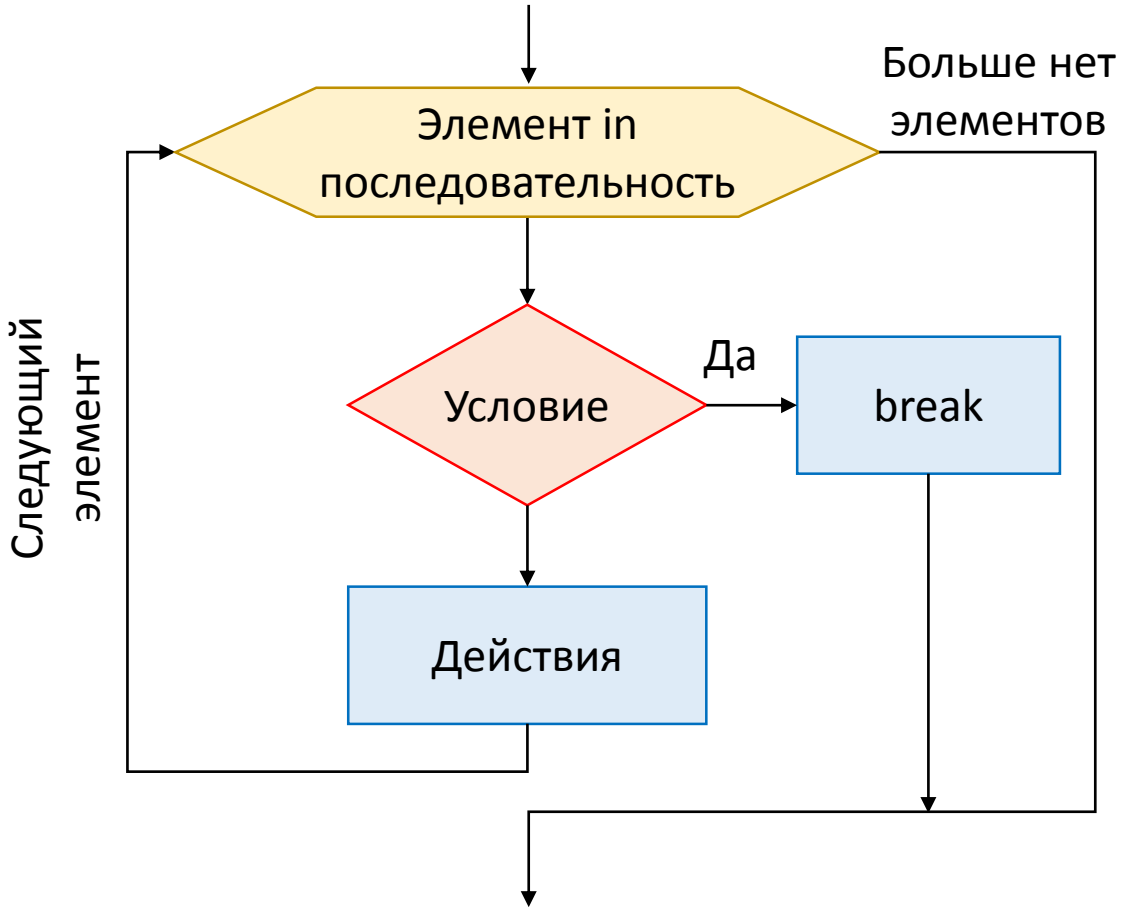
Если необходимо перейти к следующей итерации цикла до завершения всех операций в теле цикла, используется **CONTINUE**.

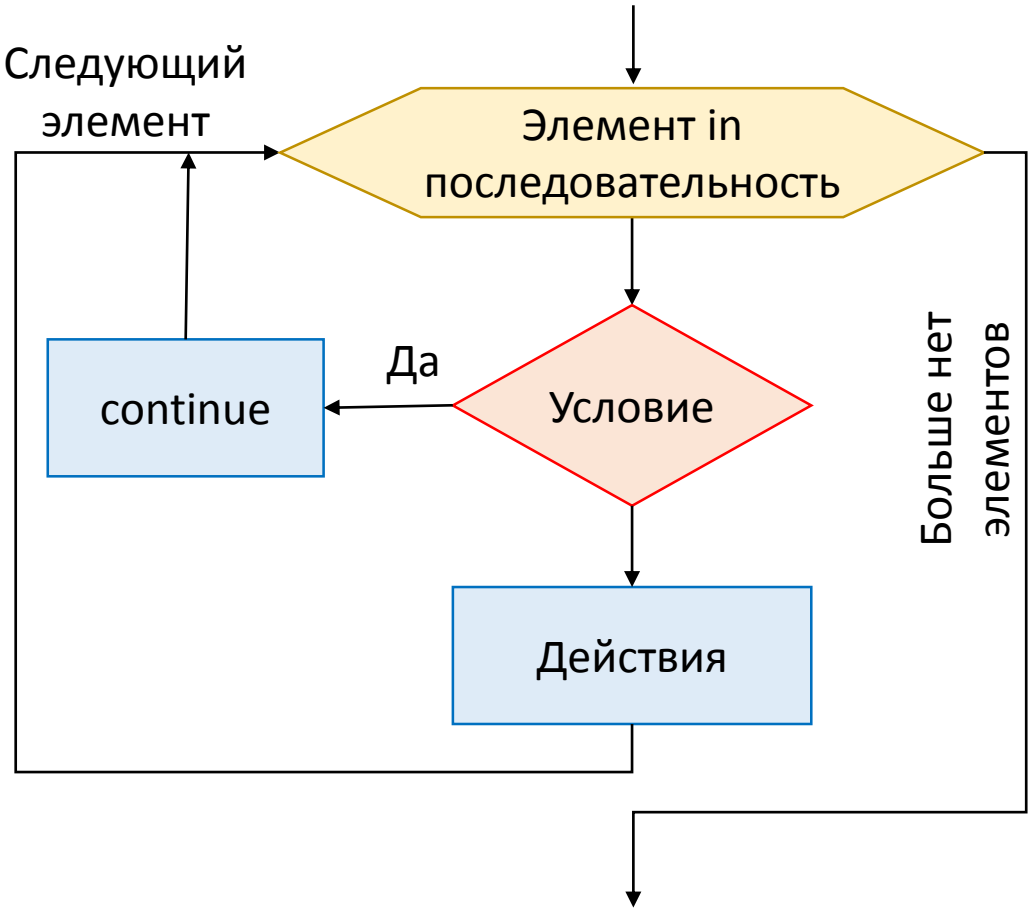
Break и continue для цикла WHILE



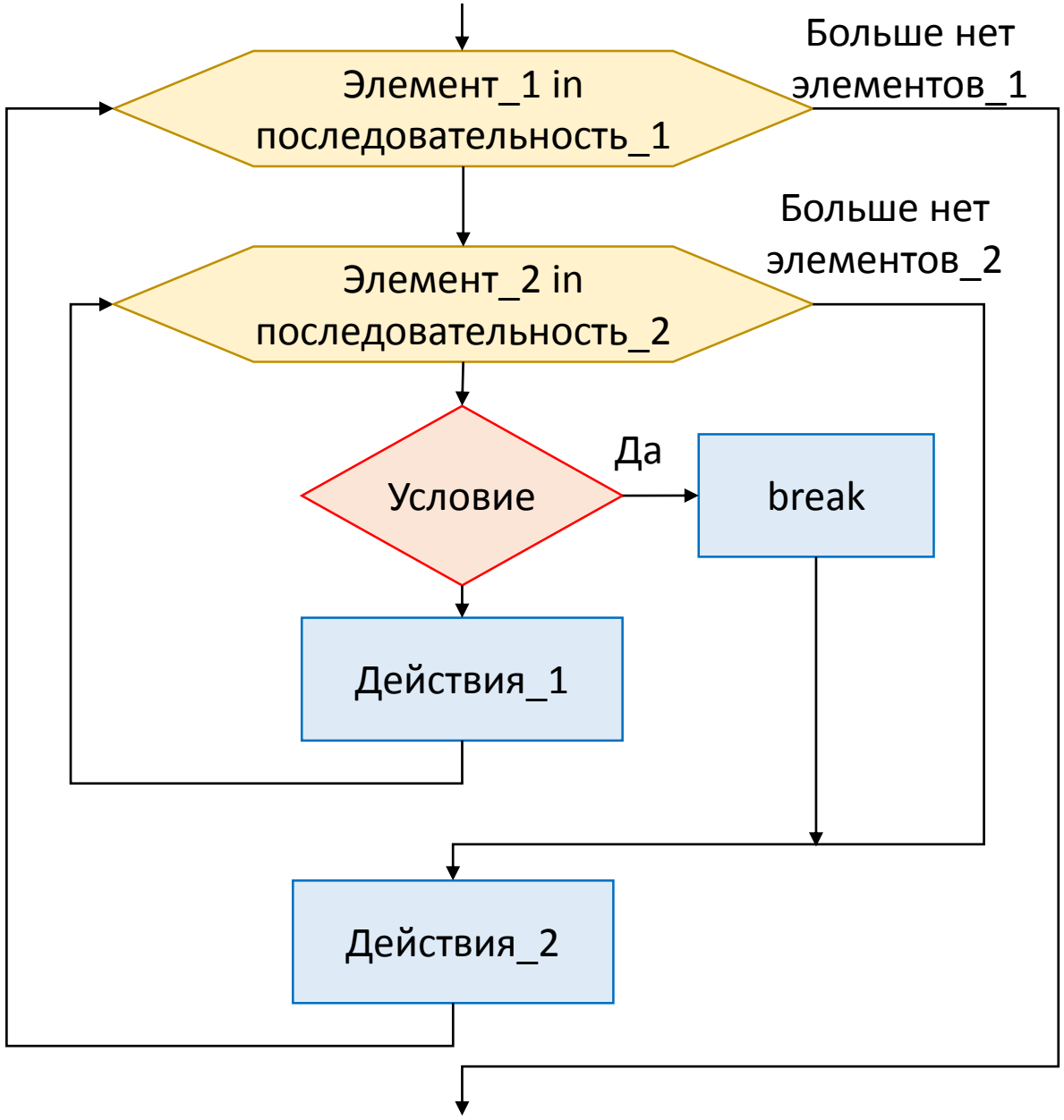


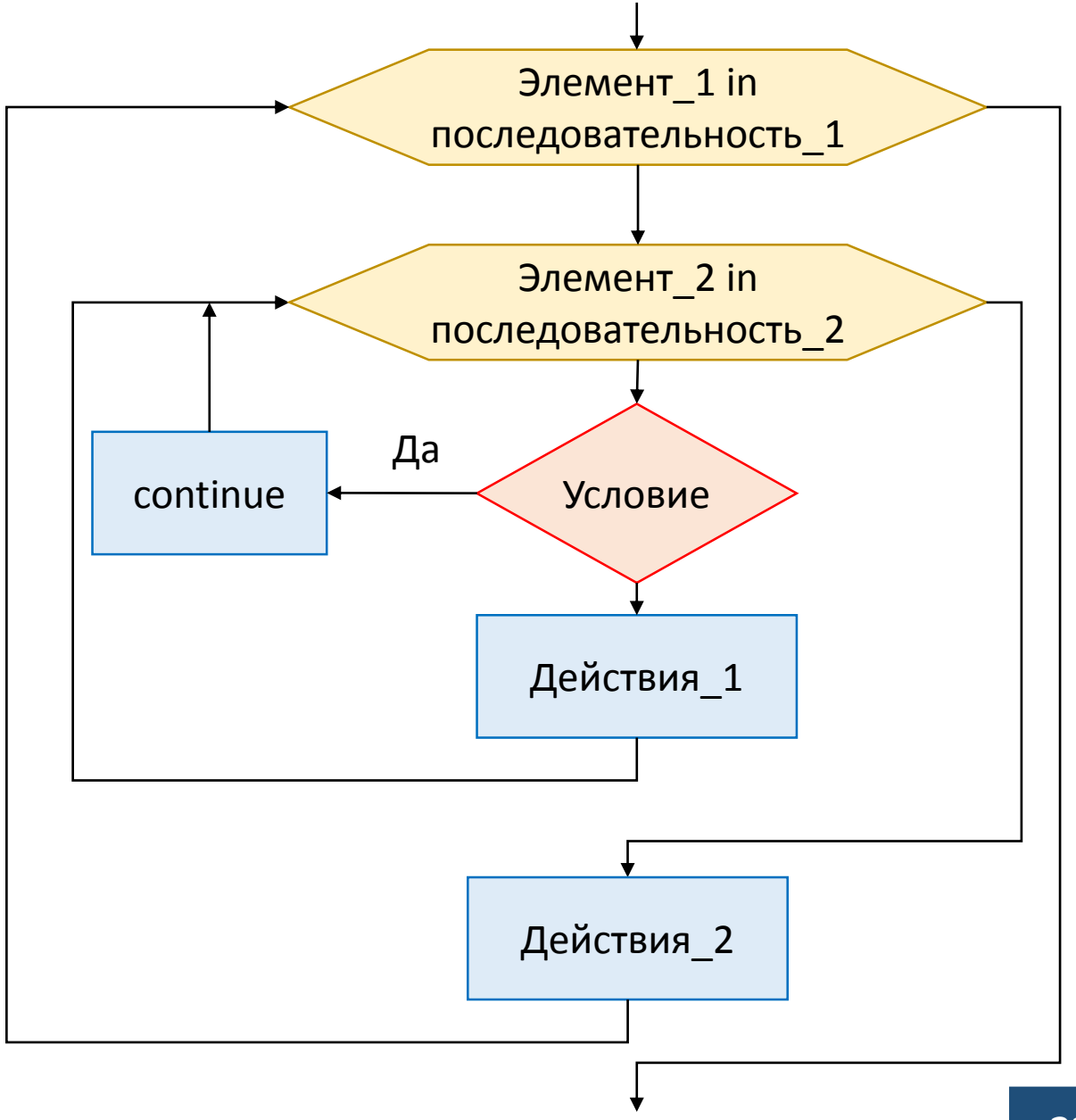
Break и continue для цикла FOR





Break и continue для цикла FOR





Пример break и continue

1

```
for y in range(1, 10):
    if(y%5==0):
        print("Remainder of %i/5 is 0" % y)
        break
    print("y = " + str(y))
```

```
y = 1
y = 2
y = 3
y = 4
Remainder of 5/5 is 0
```

2

```
x = 0
while(True):
    print("x = " + str(x))
    x = x + 1
    if(x == 7):
        break
```

```
x = 0
x = 1
x = 2
x = 3
x = 4
x = 5
x = 6
```

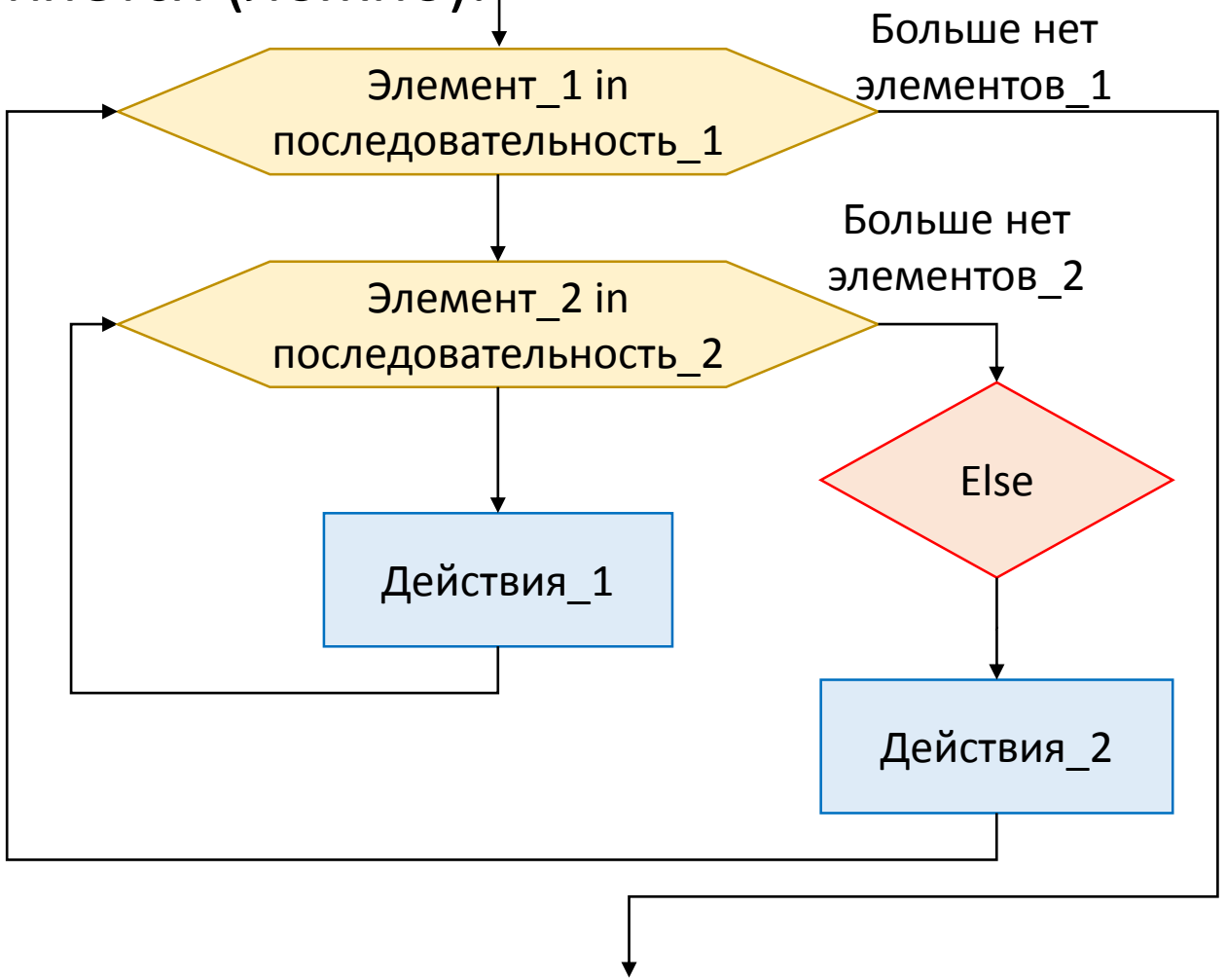
Пример break и continue

```
x = 0
for y in range(1, 10):
    if(y%2==0):
        print(str(y) + " is the even number.")
        continue
    x = x + y
    print("Sum of odd numbers within [1, %i] is %i" % (y, x))
```

```
Sum of odd numbers within [1, 1] is 1
2 is the even number.
Sum of odd numbers within [1, 3] is 4
4 is the even number.
Sum of odd numbers within [1, 5] is 9
6 is the even number.
Sum of odd numbers within [1, 7] is 16
8 is the even number.
Sum of odd numbers within [1, 9] is 25
```

Else для циклов

В цикле с использованием else контроль передается в блок else, если перебраны все элементы последовательности при for-цикле или условие while-цикла не выполняется (ложно).



Пример цикла с else

```
z = 3
for x in range(1, 5):
    for y in range(1, 5):
        if(x*y==z**2):
            print(x*y)
            break
    else:
        print("NONE")
```

```
NONE
NONE
9
NONE
```

Полезные возможности

Чтобы поменять значения двух переменных в Python можно использовать следующую форму записи:

```
(a, b) = (b, a)
```

Пример

```
a = 5
b = 10
print("BEFORE: a = %i and b = %i" % (a, b))
(a, b) = (b, a)
print("AFTER: a = %i and b = %i" % (a, b))
```

BEFORE: a = 5 and b = 10

AFTER: a = 10 and b = 5

Полезные возможности

В Python возможно использовать условия с множеством переменных.
Например, в if можно записать выражение: `a<b<c<d`

Пример

```
a = 2; b = 4; c = 5
if(a < b < c):
    print("TRUE")
else:
    print("FALSE")
```

TRUE

Источники

1. Built-in Functions. Range

2. Built-in Types. Range

3. Charles R Severance. Python for Informatics: Exploring Information

4. Compound statements

5. Conditionals and loops

6. More Control Flow Tools