

# Циклы в Python

# План лекции

1. Что такое циклы и зачем они нужны
2. Цикл *for*
3. Операторы *break* и *continue*
4. Использование *else* в циклах
5. Функция *range*
6. Цикл *while*
7. Вложенные циклы
8. Полезные конструкции в циклах

# Что такое циклы и зачем они нужны

Циклы используются в тех случаях, когда необходимо выполнить один и тот же код много раз.

Каждый язык программирования, содержит какую-либо конструкцию цикла. В большинстве языков таких конструкций несколько.

В Python есть два типа циклов:

- Цикл *for*
- Цикл *while*

# Цикл for

Этот цикл проходится по любому итерируемому объекту (например строке или списку), и во время каждого прохода выполняет тело цикла.

```
>>> for [iterating variable] in [sequence]:  
...     [do something]
```

```
>>> # Measure some strings:  
... words = ['cat', 'window', 'defenestrate']  
>>> for w in words:  
...     print(w, len(w))  
...  
cat 3  
window 6  
defenestrate 12
```

```
>>> for i in 'hello world':  
...     print(i * 2, end='')  
...  
hheellllloo  wwoorrrlidd
```

# Операторы break и continue

Оператор *continue* начинает следующий проход цикла, минуя оставшееся тело цикла

```
>>> for i in 'hello world':  
...     if i == 'o':  
...         continue  
...     print(i * 2, end='')  
...  
hheellll  wwrrlldd
```

Оператор *break* досрочно прерывает цикл.

```
>>> for i in 'hello world':  
...     if i == 'o':  
...         break  
...     print(i * 2, end='')  
...  
hheellll
```

# Использование *else* в циклах

Слово *else*, примененное в цикле *for* или *while*, проверяет, был ли произведен выход из цикла инструкцией *break*, или же "естественным" образом. Блок инструкций внутри *else* выполнится только в том случае, если выход из цикла произошёл без помощи *break*.

```
>>> for i in 'hello world':  
...     if i == 'a':  
...         break  
...     else:  
...         print('Буквы а в строке нет')  
...  
Буквы а в строке нет
```

В данном примере *else* относится именно к циклу *for* а не *if*.

# Функция *range*

`range()` позволяет генерировать ряд чисел в рамках заданного диапазона. В зависимости от того, как много аргументов вы передаете функции, вы можете указать, где этот ряд чисел начнется и закончится, а также насколько велика

```
>>> for i in range(5):  
...     print(i)  
...  
0  
1  
2  
3  
4
```

```
range(5, 10)  
5, 6, 7, 8, 9
```

```
range(0, 10, 3)  
0, 3, 6, 9
```

```
range(-10, -100, -30)  
-10, -40, -70
```

```
>>> print(range(10))  
range(0, 10)
```

# 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.

```
>>> while [a condition is True]:  
...     [do something]
```

```
>>> i = 5  
>>> while i < 15:  
...     print(i)  
...     i = i + 2  
...  
5  
7  
9  
11  
13
```



## Цикл *while True*

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.

```
>>> while True:
...     n = input("Please enter your name:")
...     if n.strip().isalpha():
...         break
...     else:
...         print("Err: Name should only include letters")
...         print("Please try again...")
```

```
>>> while True:
...     pass # Busy-wait for keyboard interrupt (Ctrl+C)
... 
```

&gt;&gt;&gt;

# Вложенные циклы

Можно использовать циклы внутри других циклов

```
>>> for [iterating variable] in [sequence]:  
...     [do something]  
...     for [iterating variable] in [sequence]:  
...         [do something]
```

```
>>> numbers = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]  
>>> multiplier = 1  
>>> print(numbers)  
>>> for i in range(0, 3):  
...     multiplier *= 10  
...     for j in range(0, 3):  
...         numbers[i][j] *= multiplier  
>>> print(numbers)  
  
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]  
[[10, 20, 30], [400, 500, 600], [7000, 8000, 9000]]
```

# Полезные конструкции в циклах

```
>>> for i, v in enumerate(['tic', 'tac', 'toe']):  
...     print(i, v)  
...  
0 tic  
1 tac  
2 toe
```

```
>>> questions = ['name', 'quest', 'favorite color']  
>>> answers = ['lancelot', 'the holy grail', 'blue']  
>>> for q, a in zip(questions, answers):  
...     print('What is your {0}? It is {1}.'.format(q, a))  
...  
What is your name? It is lancelot.  
What is your quest? It is the holy grail.  
What is your favorite color? It is blue.
```

```
>>> for i in reversed(range(1, 10, 2)):  
...     print(i)  
...  
9  
7  
5  
3  
1
```

```
>>> basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']  
>>> for f in sorted(set(basket)):  
...     print(f)  
...  
apple  
banana  
orange  
pear
```


# Полезные конструкции в циклах

Безопаснее не изменять объект, по которому проходит итерация, внутри самого цикла.

```
>>> a = [1, 2, 3]
>>> for x in a:
...     a.append(4)
...     print(a)
```

```
>>> a = [1, 2, 3]
>>> for x in a[:]:
...     a.append(4)
...     print(a)
[1, 2, 3, 4]
[1, 2, 3, 4, 4]
[1, 2, 3, 4, 4, 4]
```

Thanks for your  
attention

 [artezio\\_software](#)

 [info@artezio.com](mailto:info@artezio.com)

**[www.artezio.com](http://www.artezio.com)**