Циклы в Python

как в Python выполнять повторяющиеся действия, либо делать проход по коллекциям

Циклы

Цикл – конструкция, которая предназначена для многократного исполнения инструкций, либо для прохода по элементам итерируемого объекта

Виды циклов

В Python есть 2 вида циклов:

- Цикл for для итерации по некоторой последовательности
- Цикл while для выполнения действий, пока истинно условие

Цикл while

Выполняется пока условие **истинно**, либо пока не достигнет **break**

```
>>> counter = 0
>>> while counter < 5:
...     print(f"Счетчик равен: {counter}")
...     counter += 1
...
Счетчик равен: 0
Счетчик равен: 1
Счетчик равен: 2
Счетчик равен: 3
Счетчик равен: 4
>>> counter
5
```

Цикл for

Используется для прохода по некоторому итерируемому объекту, например, списку, словарю, строке, кортежу, множеству или другому итератору

```
>>> some_list = [1, 2, 3]
>>> for item in enumerate(some_list):
...     print(f"Элемент с индексом {item[0]} равен {item[1]}")
...
Элемент с индексом 0 равен 1
Элемент с индексом 1 равен 2
Элемент с индексом 2 равен 3
>>> some_dict = {1: 100, 2: 200}
>>> for key, value in some_dict.items():
...     print(f"Элемент с ключем {key} равен {value}")
...
Элемент с ключем 1 равен 100
Элемент с ключем 2 равен 200
```

Вопрос-ответ

? Как мне сделать цикл, который выполняется бесконечно?

✔ Нужно выполнить цикл while, у которого в качестве условия выполнения должно быть True

```
>>> while True:
...     print('Бесконечный цикл')
...     break
...
Бесконечный цикл
```

Прерываем выполнение цикла

Если нам необходимо прервать выполнение цикла (например мы нашли нужное значение), то необходимо воспользоваться ключевым словом **break**

```
>>> counter = 0
>>> while counter < 100:
...     if counter == 5:
...         break
...     print(f"счетчик равен {counter}")
...     counter += 1
...
счетчик равен 0
счетчик равен 1
счетчик равен 2
счетчик равен 3
счетчик равен 4
```

Пропускаем текущую итерацию

Если нам надо перейти к следующей итерации в цикле, то необходимо воспользоваться ключевым словом **continue**

```
>>> for i in range(10):
...     if i % 2 == 0:
...         continue
...     print(f"i равно {i}")
...
i равно 1
i равно 3
i равно 5
i равно 7
i равно 9
```

Проверяем, завершился ли цикл заранее

Для циклов есть опциональный блок **else**, который выполняется только в том случае, если ключевое слово **break не было вызвано**.

Это бывает полезно, если мы хотим выполнить некоторые действия в том случае, если предыдущий цикл выполнился полностью

```
>>> for i in range(5):
...     print(f"i равно {i}")
...     break
... else:
...     print("цикл выполнился полностью")
...
i равно 0
>>> for i in range(5):
...     print(f"i равно {i}")
... else:
...     print("цикл выполнился полностью")
...
i равно 0
i равно 1
i равно 2
i равно 3
i равно 4
цикл выполнился полностью
```

Вложенные циклы

Циклы могут быть вложены друг в друга и будут выполняться последовательно

```
>>> some_data = ['раз', 'два']
>>> for string in some_data:
...     for ch in string:
...         print(ch)
...     print('конец строки')
...
р
а
з
конец строки
д
в
а
конец строки
```

Функция range

Данная функция возвращает генератор для некоторого диапазона значений, который удобно использовать с циклами.

- range(stop)
- range(start, stop[, step])

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

Функция enumerate

Данная функция возвращает генератор, отдающий пары счетчик-элемент для элементов указанной последовательности.

- enumerate(sequence[], start=0)

```
1  sequence = ['я', 'простая', 'последовательность']
2
3  for index, item in enumerate(sequence):
4      print(f"На {index} месте {item}")
```