Циклы

Так как в анализе данных мы часто работаем со списками, нам потребуется выполнять одни и те же действия множество раз. Для этого существуют *циклы*. Цикл - это действие, которое выполняется либо опреденное количество раз, либо пока выполняется определенное *условие*.

В языке Python существует два цикла: `for` и `while`.

Цикл `while`

Этот цикл очень похож на простое условие, за исключением того, что условие выполняется только один раз, а цикл `while` - *до тех пор, пока условие соблюдается*.

Синтаксис цикла `while` таков:

```
while условие:
    действие
```

Пример:

```
i = 1                # объявляем переменную - целое число i со значением 1

while i < 5:         # цикл: пока i меньше чем 5
    print("*" * i)   # печатаем i звездочек
    i += 1           # увеличиваем число i на единицу каждую итерацию цикла
```

> Примечание: итерация - это текущее действие, выполняемое внутри цикла N раз.

Вывод программы:

```
*
**
***
****
*****
```

Для цикла `while` важно правильно составить условие. При некорректных условиях цикл может быть бесконечным:

```
a = 0

while a > 0:
    print(a)
    a += 1
```

Вывод:

```
Ошибка времени выполнения: бесконечный цикл
```

Бесконечный цикл иногда применим в коде. Для этого достаточно использовать такой синтаксис:

```
while True:
    действие
```

Операторы управления циклами

Иногда нам может потребоваться завершить цикл досрочно, или же пропустить какую-нибудь операцию (например, если цикл - бесконечный). Для этого существуют слова `break` и `continue`:

- `break` полностью останавливает текущий цикл.
- `continue` пропускает одну итерацию в цикле и переходит к следующей, не выполняя код после себя.

Пример:

```
drinks = ["cola", "coffee", "tea", "water", "sprite", "vodka", "matcha", "juice"]

i = 0                                   # объявляем переменную-счетчик
while True:                             # бесконечный цикл
    drink = drinks[i]                   # получаем текущий напиток из списка по инде
    if drink == "water" or "tea":       # если напиток - вода или чай, пропускаем ег
        continue
    if drink == "vodka":                # если напиток - водка, прерываем цикл
        break
    print(drink)                        # печатаем напиток
    i += 1                              # увеличиваем счетчик
```

Вывод:

```
cola
coffee
sprite
```

Цикл `for`

Для работы со списками гораздо удобнее использовать цикл `for`. Этот цикл выполняется столько раз, сколько значений содержится в заданном ему диапазоне. Проще всего будет рассказать об этом цикле, переписав пример выше с напитками с использованием этого цикла:

```
drinks = ["cola", "coffee", "tea", "water", "sprite", "vodka", "matcha", "juice"]

for drink in drinks:                    # для каждого элемента в drinks:
    if drink == "water" or "tea":
        continue
    if drink == "vodka":
        break
    print(drink)
```

Как видно, цикл `for` также поддерживает операторы `break` и `continue`, но является более удобным в применении.

Помимо итерации по элементам списка, можно итерироваться по диапазонам целых чисел с использованием функции `range(from, to)`:

> Примечание: функция `range(a, b)` возвращает диапазон целых чисел от a до b:
>
> $$[a, b)$$
>
> Если a не задано, то функция вернет диапазон:
>
> $$[0, b)$$

Пример: выведем все числа, делящиеся на 67 без остатка от нуля до тысячи:

```
for number in range(0, 1000):
    if number % 67 == 0:
        print(number)
```

Вывод:

```
0
67
134
201
268
335
402
469
536
603
670
737
804
871
938
```

С помощью цикла `for` удобно выводить списки:

```
animals = ['cat', 'dog', 'fox', 'elephant', 'horse', 'cow']

for animal in animals:
    print(animal)

print()                             # пропуск строки
for animal in animals:
    print(animal, end=" ")          # вывод в строчку через пробел
```

Вывод:

```
cat
dog
fox
elephant
horse
cow

cat dog fox elephant horse cow
```

Вложенные циклы

Внутри циклов можно писать другие циклы. Например, мы хотим нарисовать квадрат из решеток размером $N \times M$:

```
N = 8
M = 20

for n in range(N):
    for m in range(M):
        print('#', end="")
    print()
```

Вывод:

```
####################
####################
####################
####################
####################
####################
####################
####################
```

В циклах и вложенных циклах можно писать условия и выполнять любой код, который можно выполнить вне цикла. Пример с рисованием аналогичного квадрата из решеток, но полого внутри с толщиной стенок S:

```
N = 8
M = 20
S = 2

for n in range(N):
    for m in range(M):
        if m < S or m >= M - S or n < S / 2 or n >= N - S / 2:
            print('#', end="")
        else:
            print(" ", end="")
    print()
```

Вывод:

```
####################
##                ##
```

```
##                  ##
##                  ##
##                  ##
##                  ##
##                  ##
####################
```