Python: циклы | Я.Шпора

Цикл — это операция, позволяющая многократно выполнять один и тот же фрагмент кода.

Цикл `for`

Применяется в тех случаях, когда определённый набор операций надо выполнить известное количество раз.

Цикл `for` всегда применяется в сочетании с коллекцией или с иным итерируемым объектом.

Как объявить и применить цикл `for`

```
vegetables = ['Помидоры', 'Огурцы', 'Баклажаны', 'Перец', 'Капуста']

# Объявление цикла можно прочитать так:
# "Для (for) каждого элемента в (in) коллекции vegetables..."
for vegetable in vegetables:
    # "...выполнить код в теле цикла."
    # Напечатать содержимое переменной vegetable.
    print(vegetable)

# Код вне цикла.
print('Цикл завершился, и теперь выполняется код, следующий за циклом.')

# Вывод в терминал:
# Помидоры
# Огурцы
# Баклажаны
# Перец
# Капуста
# Цикл завершился, и теперь выполняется код, следующий за циклом.
```

Цикл while

Используется в ситуациях, когда количество итераций неизвестно. Цикл будет выполнять вложенный в него код до тех пор, пока будет соблюдаться условие,

заданное при его объявлении.

Как объявить цикл `while`

```
from random import randint

is_door_open = True

# До тех пор, пока дверь открыта...
while is_door_open == True:
    # ...кошка тщательно выполняет свою программу:
    print('Кошка носится из комнаты в комнату.')

    # Случайным образом получить число в диапазоне от 0 до 10 включительно
    # и преобразовать его в bool: 0 преобразуется в False,
    # а любое другое число - в True.
    # Получившееся булево значение присваивоить переменной.
    is_door_open = bool(randint(0, 10))

print('Дверь закрыта, кошка заперта!')

# Вывод в терминал:
# Кошка носится из комнаты в комнату.
# ...
# Дверь закрыта, кошка заперта!
```

Вложенные циклы

Циклы можно вкладывать друг в друга, по аналогии с условиями. Синтаксис вложенного цикла `for`:

```
# Внешний цикл, он перебирает коллекцию elements.
for element in elements:
    ...             # Выполнить какие-то действия в цикле,
    print(element)  # например распечатать каждый элемент.
    for item in element:  # И прямо в теле цикла объявить новый цикл.
        ...               # И в этом цикле тоже выполнить какие-то действи
```

Синтаксис вложенного цикла `while`:

```
while условие_1:
    while условие_2:
        тело вложенного цикла
```

Управление циклами

Инструкция `continue`

Если не вмешиваться в работу цикла, он на каждой итерации выполнит весь код в теле цикла. Но можно вмешаться и настроить цикл так, чтобы он не выполнял итерацию полностью. Для этого используется оператор `continue`:

```
numbers = [1, 2, 3, 4, 5]
for number in numbers:
    if number % 2 != 0:
        continue
    print(number)

# Вывод в терминал:
# 2
# 4
```

Инструкция `break`

Прерывает текущий цикл, итерация по последовательности прерывается:.

```
numbers = [1, 2, 3, 4, 5]
for number in numbers:
    if number % 2 == 0:
        break
    print(number)

# Вывод в терминал:
# 1
```

Когда применять `continue` и `break`

- `continue` полезен, когда нужно пропустить определённые итерации цикла

или определённые действия внутри итерации, но при этом нужно продолжить выполнение оставшихся итераций.

- `break` полезен, когда нужно немедленно завершить выполнение цикла: например, когда найден нужный элемент или достигнуто некоторое условие.

> Эти операторы воздействуют только на текущий цикл. Если использовать их во вложенных циклах, они не повлияют на внешние циклы.

Списковые и словарные включения

Списковое включение

Чтобы создать список с помощью цикла, используется такой синтаксис:

```
new_sequence = [<значение_элемента> for <переменная_цикла> in <исходный_сі
```

Пример:

```
new_sequence = []

for value in range(6):
    list.append(new_sequence, value * 3)

print(new_sequence)

# А теперь то же самое запишем в сокращённой форме:
new_sequence_better = [value * 3 for value in range(6)] # Всё!
print(new_sequence_better)

# Вывод в терминал:
# [0, 3, 6, 9, 12, 15]
# [0, 3, 6, 9, 12, 15]
```

Словарные включения

Словари можно создавать при помощи конструкции *dict comprehension*. Эта конструкция создаёт словарь на основе значений исходной последовательности — перебирает последовательность в цикле, получает её элементы и создаёт ключи и значения элементов словаря.

```
<имя_словаря> = {
    <ключ>: <значение> for <переменная_цикла> in <имя_исходной_последов
ательности>
}
```

Пример:

```
# В цикле for перебрать последовательность range(3).
new_collection = {f'Ключ {value}': f'Значение {value}' for value in range(

print(new_collection)

# Вывод в терминал:
# {'Ключ 0': 'Значение 0', 'Ключ 1': 'Значение 1', 'Ключ 2': 'Значение 2'}
```

Различать, когда полагаться на шпаргалку, а когда на собственную память — вот знание мудрого.

Тест Эмпирик, древнеримский экспериментатор