ЯЗЫК PYTHON

Лекция 5. Цикл for

Цикл for

- Специальный цикл
- Но очень популярный
- Часто в цикле хочется пройтись по строчке
- Или по списку
- Перебрать все элементы

Цикл for

- Есть отдельная конструкция
- Сначала ключевое слово for
- Потом имя переменной, в которой будет хранится очередное значение
- Ключевое слово in
- Список или строка
- Двоеточие и тело цикла

Пример со списком

```
1 data = [12, 43, 11, 0, 34]
2 for value in data:
3     print(value)
```

Проанализируем

- Перебираем элементы списка
- По итерации на элемент
- В каждой итерации value принимает очередное значение
- 12, потом 43 и так далее

Можно по строке

- Если после in идет строка
- То получаем цикл по символам
- В каждой итерации
  - переменной цикла присваивается односимвольная строка

Пример со строкой

```
1 for c in 'hello':
2     print(c)
```

Проанализируем

- Перебираем элементы строки
- По итерации на элемент
- В каждой итерации с принимает очередное значение
- 'h', потом 'e' и так далее

Повторение действий несколько раз

- Это можно сделать тоже циклом for
- Справа от in надо поставить вызов встроенной функции range
- И передать ей число
- Какой число передадим - столько раз цикл и исполнится

Пример с повторением

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

Если переменная цикла не нужна

- Хотим 10 раз напечатать 'hello'
- Цикл нужен
- Переменная цикла - нет
- В таких случаях используется псевдопеременная _

Пример с псевдопеременной

```
1 for _ in range(10):
2     print('hello')
```

Можем перебрать диапазон

- Используем range с двумя параметрами
- Смысл параметров - как в отрезках
- Да, третий параметр тоже можно использовать
- С таким же смыслом, как в отрезках

Пример с двумя параметрами

```
1 for i in range(123, 140):
2     print(i, i * i)
```

Пример с шагом

```
1 for i in range(1, 20, 3):
2     print(2 ** i)
```

Пример с отрицательным шагом

```
1 for i in range(10, 3, -1):
2     print(i ** i)
```

Нумерация элементов списка

- В списке лежат данные о замерах температуры за месяц
- Сначала за 1 число, потом второе и т.п.
- Хотим узнать дни с минусовой температурой
- Можно сделать цикл по range
- И использовать переменную цикла как индекс

Нумерация через range

```
1 for i in range(len(data)):
2     if data[i] < 0:
3         print(i + 1)  # потому что нумерация с нуля
```

enumerate

- Есть функция enumerate
- Превращает список в последовательность пар
- Второй элемент пары - очередной элемент списка
- Первый - номер элемента
- Начиная с 0

Групповое присваивание и пара

- Пара - специальный тип данных
- К паре можно применить групповое присваивание
- Слева от знака присваивания - две переменные
- Через запятую
- Первой присваивается первый элемент, второй - второй

Цикл и присваивание

- В начале цикла очередное значение присваивается переменной цикла
- Если цикл идет по парам
- То переменная цикла будет принимать значение-пару
- К которой можно применить групповое присваивание

Использование enumerate

```
1 for pair in enumerate(data):
2     index, value = pair
3     if value < 0:
4         print(index + 1, value)  # потому что нумерация с ну
```

Цикл и присваивание

- Но in - тоже в каком-то смысле присваивание
- И если цикл идет по парам
- Python позваляет слева от in указать пару переменных
- И получает общепринятый вариант перебора с индексами

Общепринятый вариант

```
1 for index, value in enumerate(data):
2     if data[index] < 0:
3         print(index + 1, value)  # потому что нумерация с ну
```

Цикл по двум спискам

- Студенты сдают экзамен
- Их вызывают по списку
- И оценки ставят тоже в список
- В другой
- Имея два списка, хочется напечатать оценку рядом с фамилией студента

Цикл по двум спискам

- Могли бы опереться на enumerate
- По одному из списков
- И использовать индекс
- Чтобы прочитать элемент из второго

Вариант через enumerate

```
1 for index, value in enumerate(names):
2     print(value, scores[index])
```

zip

- Но есть функция zip
- Принимает два параметра-списка
- И возвращает последовательность пар
- Первый элемент - очередной из первого списка
- Второй - очередной из второго
- С ней красивее

Вариант через zip

```
1 for name, score in zip(names, scores):
2     print(name, score)
```

Перебор строк текстового файла

- Можно открыть текстовый файл
- И справа от in использовать его
- Получится цикл по строкам файла
- Надо учесть, что перевод строки будет частью строкового значения

Перебор строк файла

```
1 with open('data.txt', 'rt') as f:
2     for line in f:
3         print(line, end='')
```

zip/enumerate

- Все про zip/enumerate остается в силе для файлов
- Можно перенумеровать их
- Или сделать цикл по строкам двух файлов
- Например, сравнивая их

Нумерация строк файла

```
1 with f = open('data.txt', 'rt'):
2     for line in enumerate(f):
3         print(index, line, end='')
```

Склейка строк файла

```
1 with open('data1.txt', 'rt') as f1:
2     with open('data2.txt', 'rt') as f2:
3         for line1, line2 in zip(f1, f2):
4             if line1 != line2:
5                 print(line1, end='')
6                 print(line2, end='')
7                 print()
```

enumerate не с нуля

- Не всегда хочется нумеровать с нуля
- Например, для строк файла часто ожидается нумерация с 1
- И в списке могут быть дни месяца
- А корректировать прибавкой 1 не очень удобно
- А еще итерироваться по отрезку
- И хочется, чтобы нумерация соответствовала
- У enumetrate есть для этого параметр

Нумерация с заданного индекса

```
1 for index, value in enumerate(data[5:], 5):
2     if value < 0:
3         print('negative at index', index)
```

break/continue/else

- break позволяет выйти из цикла
- До завершения перебора
- continue позволяет перейти на новую итерацию
- else исполняется, завершили перебор
- И не вышли по break

break и else

```
1 for value in data:
2     if value % 2 == 1:
3         print('odd found', v )
4         break
5 else:
6     print('No add found')
```

Пример вложенного for

```
1 data = ['apple', 'pineapple', 'mellon']
2 count = 0
3 for s in data:
4     for c in s:
5         if c in 'aeiouy':
6             count += 1
7 print('found ', count, 'vowels')
```

Изменение списка в цикле

- Хотим пройти по списку
- И строки, начинающиеся с 'a', добавить в конец списка
- Вроде несложно
- Добавлять в конец мы умеем
- Приваиванием отрезку

Проблемный код

```
1 data = ['apple', 'pineapple', 'mellon']
2 for s in data:
3     if s[0] == 'a':
4         data[len(data):] = [s]
```

Изменение списка в цикле

- Проблема в том, что мы постоянно наращиваем список
- Любое изменение списка, по которому идет цикл, чревато неприятностями
- Есть простое решение
- Скопировать список

Исправленный код

```
1 data = ['apple', 'pineapple', 'mellon']
2 for s in data[:]:
3     if s[0] == 'a':
4         data[len(data):] = [s]
```

Методы strip/rstrip

- Иногда строка приходит из внешнего источника
- С клавиатуры или из файла
- И в ней могут быть пробелы спереди или сзади
- И их хочется обрезать
- Чтобы не мешали

Методы strip/lstrip/rstrip

- Для этого есть два метода
- strip, lstrip и rstrip
- Первый - обрезаем пробельные символы с обоих сторон
- Второй - только слева
- Третий - только справа

Методы strip/lstrip/rstrip

- Перевод строки тоже считается пробельным символом
- Благодаря этому strip и rstrip часто используются в циклах по строкам файла
- Чтобы обрезать перевод строки
- Но они отрезают и пробелы тоже
- Надо это учитывать

Печатаем строки длиннее 10 символов

```
1 with open('data.txt', 'rt') as f:
2     for index, line in enumerate(f):
3         line = line.strip()
4         if len(line) > 10:
5             print(index, line)
```