НЕТОЛОГИЯ

ВВЕДЕНИЕ В ТИПЫ ДАННЫХ, ЦИКЛЫ

ОЛЕГ БУЛЫГИН

ОЛЕГ БУЛЫГИН

IT-аудитор в ПАО “Сбербанк"

obulygin91@ya.ru

fb.me/obulygin91

План занятия

1. Простые типы данных
2. Списки
3. Кортежи
4. Циклы

Простые типы данных

Простые типы данных

Выделяют 4 типа данных:

- **int** (integer) – целые числа
- **float** – действительные числа
- **str** (string) – строки
- **bool** (boolean) – логический тип

Примеры:

```
number = 10
q = 9.8
name = 'Коля'
sun = True
```

Тип объекта можно узнать при помощи функции `type()`.

Тип данных можно принудительно изменить функциями `int()`, `float()`, `bool()`, `str()` и т.д.

Операции со строками

1. Конкатенация (объединение) строк возможна при помощи `+` ;
2. Умножение строки на число позволит повторить ее нужное количество раз;
3. `.upper()` приводит строку к верхнему регистру;
4. `.lower()` приводит строку к нижнему регистру;
5. `.capitalize()` приводит первую букву к верхнему регистру;
6. `.replace('что заменить', 'на что заменить')` заменяет элемент в строке на указанный;
7. `len(my_string)` позволяет определить длину строки (количество символов в ней);

Со всеми методами работы со строками можно ознакомиться по ссылке.

Форматирование строк (f-строки)

Добавляя префикс `f` к строке, можно встраивать в нее произвольные выражения при помощи фигурных скобок `{ }`.

```
name = 'Коля'
age = 13
print(f'Меня зовут {name}. Я родился в {2020-age} году.')
```

Индексация и срезы строк

Доступ к элементам объекта по их порядковому номеру в нем.

Индексация элементов **начинается с нуля**.

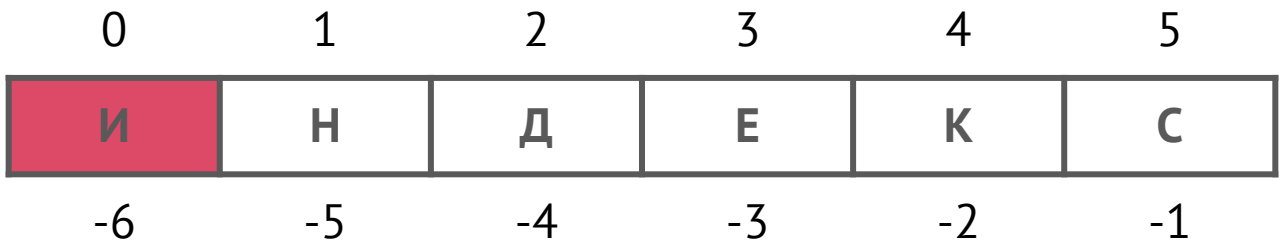
Поиск символов

Получить значение элемента по индексу можно при помощи `[ ]`.

`my_string[0]`

или

`my_string[-6]`



Срез строк

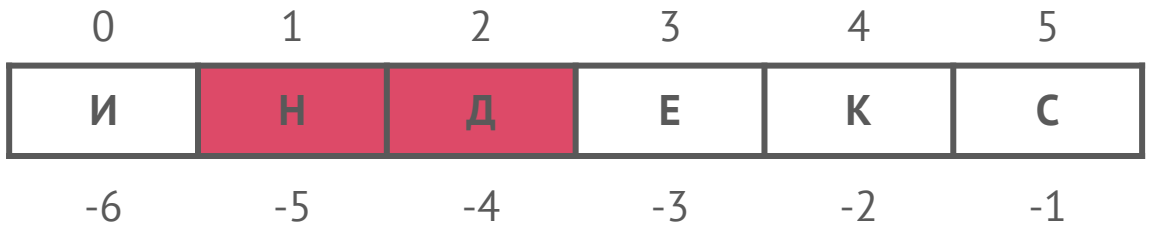
Можно извлечь из строки несколько элементов при помощи “срезов” (slicing). Для указания интервала среза используется `:`.

Синтаксис: `string[start:stop]`, где

`start` – индекс первого элемента в списке,

`stop` – индекс списка, перед которым срез должен закончиться (т.е. сам элемент с индексом stop не будет входить в выборку.

`my_string[1:3]`

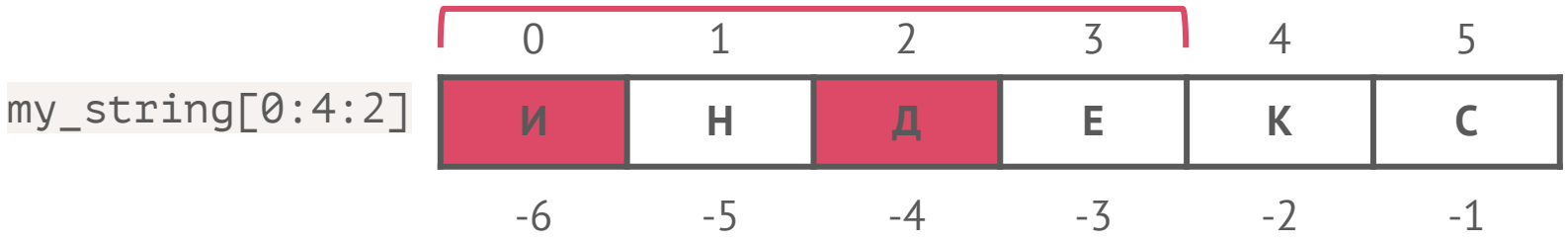


Срез строк

Срез с шагом. Шаг указывает, на сколько символов нужно подвинуться после взятия первого символа.

Синтаксис: `string[start:stop:step]`, где

`step` – шаг прироста выбираемых индексов.



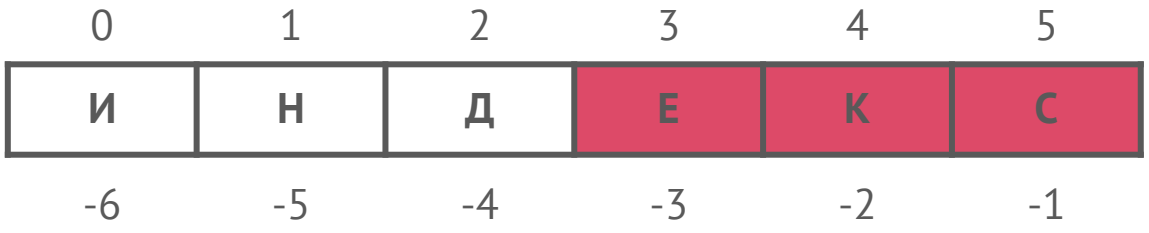
Срез строк

`string[start:stop:step]`

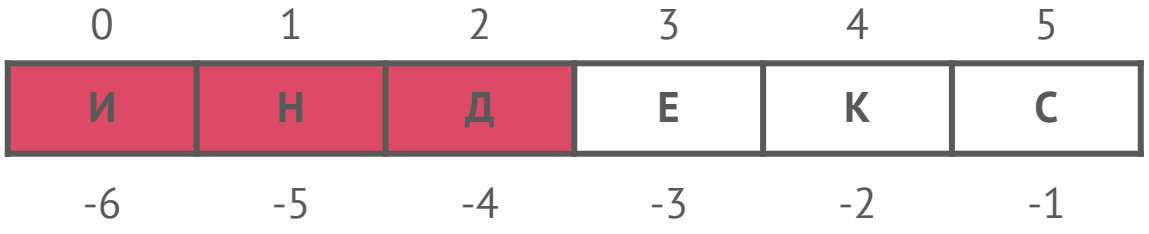
При этом любой из параметров может быть опущен. Тогда вместо соответствующего параметра будет выбрано значение по умолчанию:

- `start` по-умолчанию означает «от начала списка»,
- `stop` по-умолчанию означает «до конца списка» (включительно),
- `step` по-умолчанию означает «брать каждый элемент».

`my_string[3:]`



`my_string[:3]`



Списки

Списки (list)

Это структура данных для **упорядоченного** хранения объектов **различных** типов.

Последовательность элементов в списке начинается с 0, как у символов в строке.

0	1	2	3
‘Петров’	**‘Николай’**	**‘Иванович’**	**25**

Список “Данные пользователя”

Является изменяемым типом данных, в отличие от всех предыдущих.

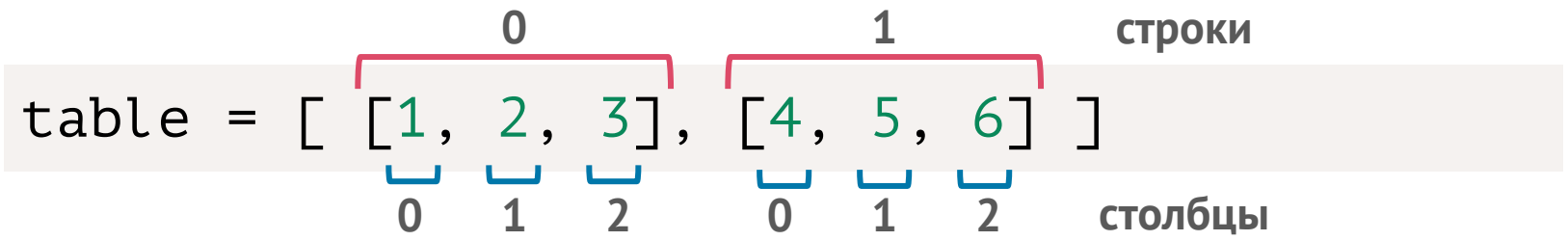
Списки

Список инициализируется при помощи `[ ]`, элементы в списке разделяются запятыми.

```
name_list = [] #пустой список
user_data = ['Петров', 'Николай', 'Иванович', 25]
```

Многомерные списки

Внутри одного списка могут быть другие списки.



Представить многомерный список можно в виде таблицы:

	0	1	2
0	**1**	**2**	**3**
1	**4**	**5**	**6**

Операции со списками

- списки можно складывать;
- `del(list[index])` удаляет элемент из списка по индексу;
- `.remove(el)` удаляет указанный элемент из списка;
- `.append(el)` позволяет добавить элемент в список;
- `.count(el)` считает количество вхождений элемента в список;
- `.index(el)` позволяет узнать индекс элемента в списке;
- `.reverse()` разворачивает список;
- `sorted(list)` сортирует список;
- ...

Мы можем менять элементы списка при помощи индексации и срезов (т.к. **списки изменяемы**).

Со всеми операциями над списками можно ознакомиться по ссылке.

Кортежи

Кортежи (tuples)

Это неизменяемые списки (нельзя добавлять или удалять элементы из уже созданного кортежа).

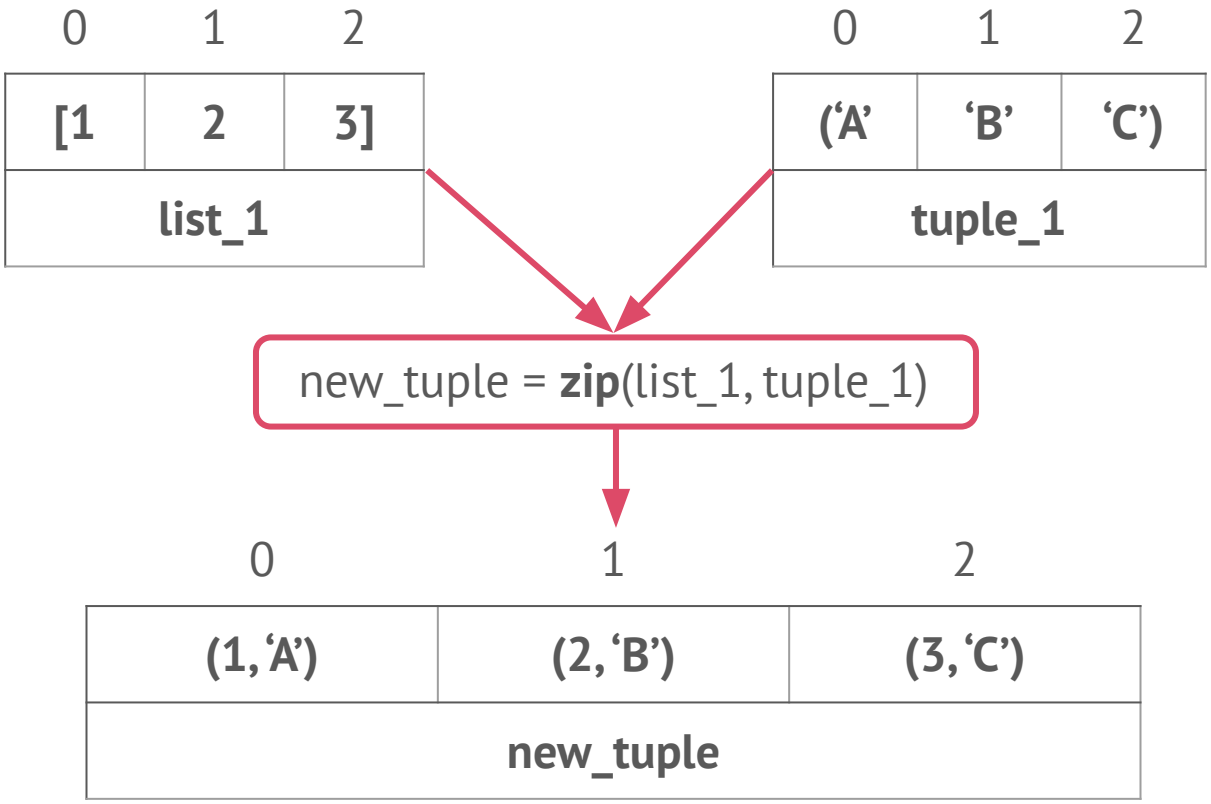
Кортежи инициализируется при помощи `( )`.

```
user_data = ('Петров', 'Николай', 'Иванович', 25)
```

> Занимает меньше памяти при работе с ними по сравнению со списками.

Функция `zip`

Функция `zip(list_1, list_2, ...)` берёт на вход несколько списков/кортежей и создаёт из них специальный zip-объект, состоящий из кортежей, такой, что первый элемент полученного объекта содержит кортеж из первых элементов всех списков-аргументов.



Операторы проверки вхождения

IN – возвращает **True**, если элемент **входит** в объект.

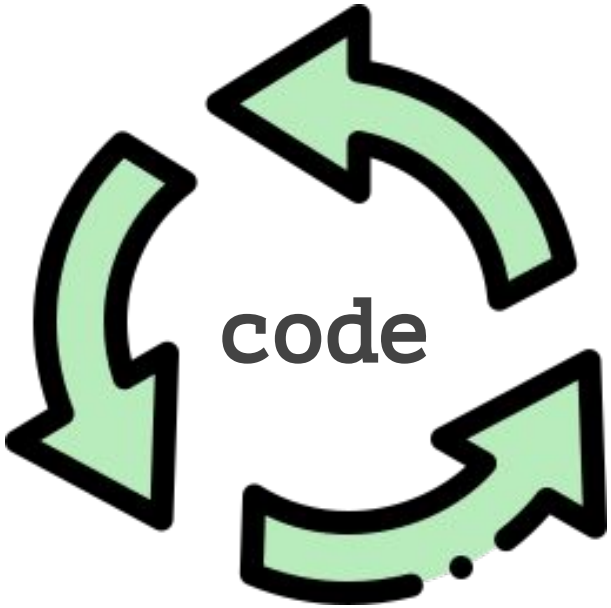
NOT IN – возвращает **True**, если элемент **не входит** в объект.

Циклы

Циклы

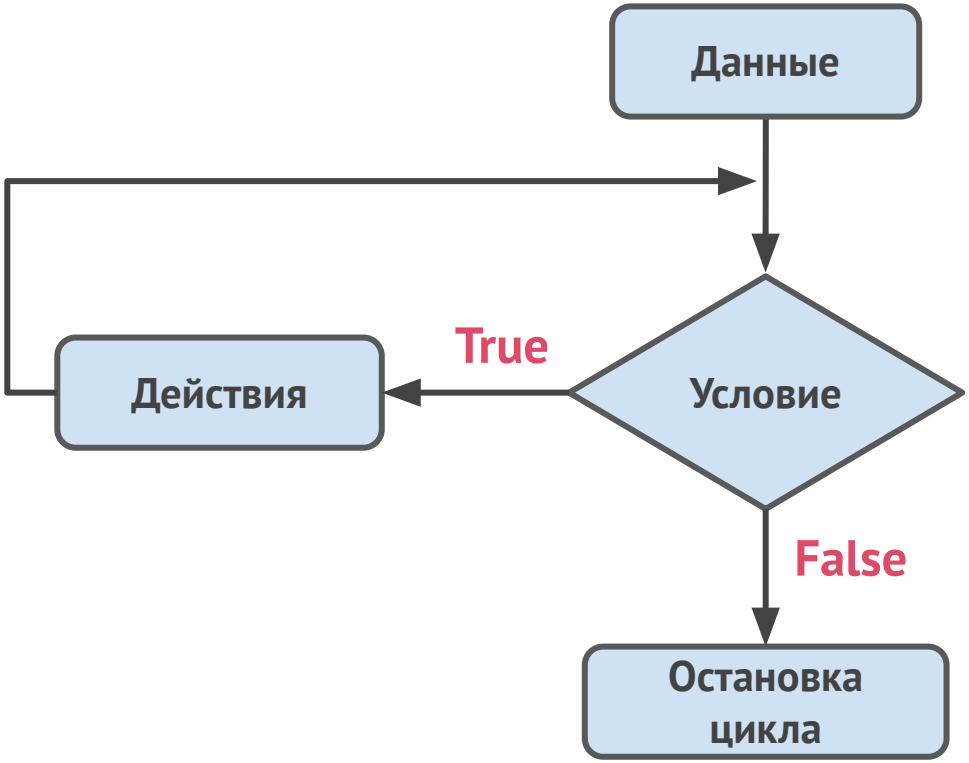
Циклы позволяют организовать повторение выполнения участков кода.

В Python существует два типа циклов: цикл `while` и цикл `for`.



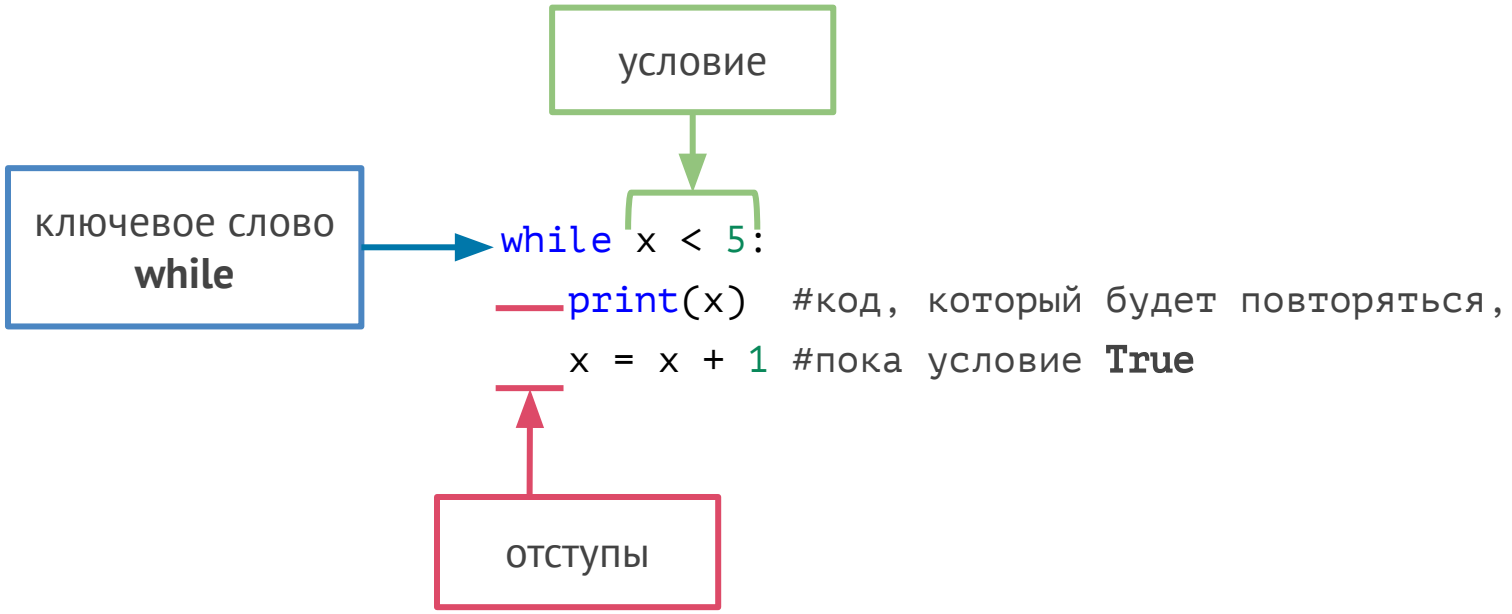
Цикл `while`

Позволяет выполнить одну и ту же последовательность действий, пока проверяемое условие истинно.



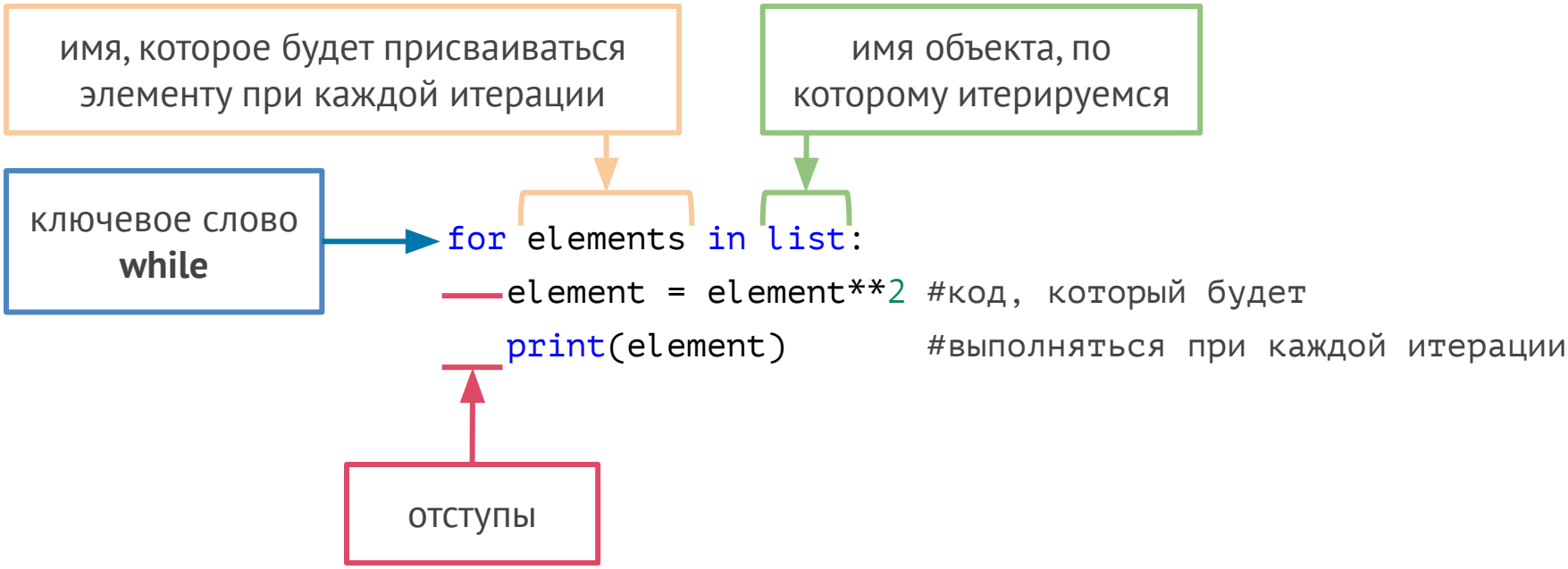
Цикл `while`

Как правило, цикл `while` используется, когда невозможно заранее определить точное значение количества проходов исполнения цикла.

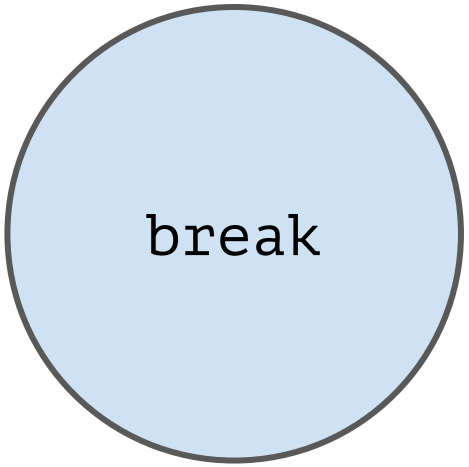


Цикл `for`

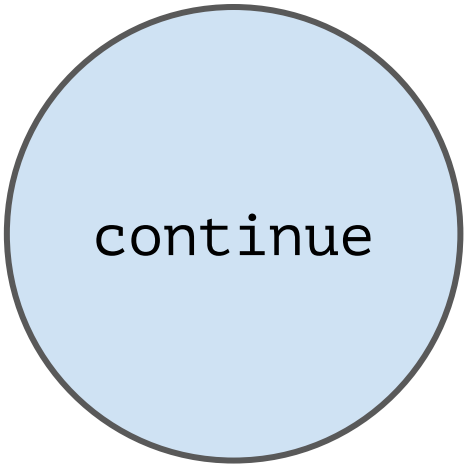
Цикл `for` проходится по элементам любого итерируемого объекта (строки, списка и т.д.) и во время каждого прохода выполняет заданную последовательность действий.



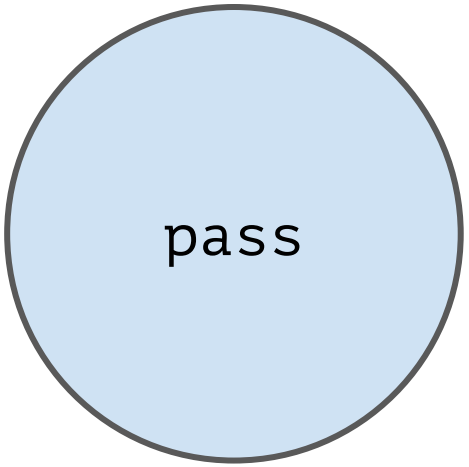
Ключевые слова `break`, `continue` и `pass`



Прерывает исполнение цикла



Завершает исполнение текущей итерации цикла и переходит к следующей итерации



Игнорирует условие и продолжает исполнение цикла

Домашнее задание

Давайте посмотрим ваше домашнее задание.

- Вопросы по домашней работе задаём в чате Slack!
- Задачи можно сдавать по частям.
- Зачёт по домашней работе проставляется после того, как приняты **все задачи**.

НЕТОЛОГИЯ

Задавайте вопросы и напишите отзыв о лекции!

ОЛЕГ БУЛЫГИН

 obulygin91@ya.ru

 fb.me/obulygin91