

Введение в типы данных и циклы

Олег Булыгин

Нетология



Булыгин Олег

- Преподаватель на курсах “Основы языка программирования Python”, “Продвинутый Python”, “Python для анализа данных” в Нетологии
- Начальник бюро планирования и управления в АО “НПО автоматики”



o.bulygin@netology.ru



<https://www.facebook.com/obulygin91>



О чем мы поговорим сегодня

1. Простые типы данных
2. Списки
3. Кортежи
4. Множества
5. Словари
6. Цикл *while*
7. Цикл *for*



Простые типы данных

- *integer* (целые числа);
- *float* (числа с плавающей точкой);
- *string* (строка/текст);
- *boolean* (булевый/логический тип).

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

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



Операции со строками

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



Индексация и срезы строк

Индексация – доступ к элементам объекта по их порядковому номеру в нем. Индексация элементов начинается с нуля.

Получить значение элемента по индексу можно при помощи `[]`, например: `my_string[0]` и `my_string[-6]` вернет **И**.

Можно “доставать” из строки несколько элементов при помощи “срезов” (slicing). Для указания интервала среза используется `:`, например `my_string[0:4:2]`

0	1	2	3	4	5
И	Н	Д	Е	К	С
-6	-5	-4	-3	-2	-1



Форматирование строк (f-строки)

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



Списки

Список (list) – это структура данных для упорядоченного хранения объектов различных типов. Является изменяемым типом данных, в отличие от все предыдущих.

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

В одном списке могут быть одновременно элементы разных типов (даже другие списки).



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).



Кортежи

Кортежи (tuples) – неизменяемые списки (нельзя добавлять или удалять элементы из уже созданного кортежа).

Кортежи инициализируется при помощи ().

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



Функция `zip`

Над кортежами доступны все операции над списками, не изменяющие список.

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



Множества

Множества (sets) – “контейнер”, содержащий не повторяющиеся элементы в случайном порядке

Множества инициализируется при помощи **set()**, как правило создаются из списков.

Реализуют теорию множеств в Python.



Операции над множествами

- *.add(el)* добавляет элемент в множество;
- *.update(set)* соединяет множество с другим множеством/списком;
- *.discard(el)* удаляет элемент из множества по его значению;
- *.union(set)* объединяет множества (логическое “ИЛИ”);
- *.intersection(set)* – пересечение множеств (логическое “И”);
- *.difference(set)* – возвращает элементы одного множества, которые не принадлежат другому множеству (разность множеств);
- *.symmetric_difference(set)* – возвращает элементы, которые встречаются в одном множестве, но не встречаются в обоих



Словари

Словари (dictionaries) – неупорядоченные коллекции произвольных объектов с доступом по ключу.

Словарь инициализируется при помощи `{ }`, элементы них хранятся в формате *key:value*.

Ключами могут быть *strings*, *booleans*, *integers* и *floats*.

Любое значение из словаря можно получить следующим образом: **`my_dict[key]`**.

Все ключи в словаре должны быть уникальными.



Операции со словарями

- *del(dict[key])* удаляет элемент из списка по ключу;
- *.keys()* позволяет получить все ключи словаря;
- *.values()* позволяет получить все значения словаря;
- *.items()* позволяет получить ключи и значения словаря;
- *.get(key)* “безопасно” возвращает значение по ключу (при отсутствии ключа ошибка не возникает);



Циклы

Циклы позволяют организовать повторение выполнения участков кода.

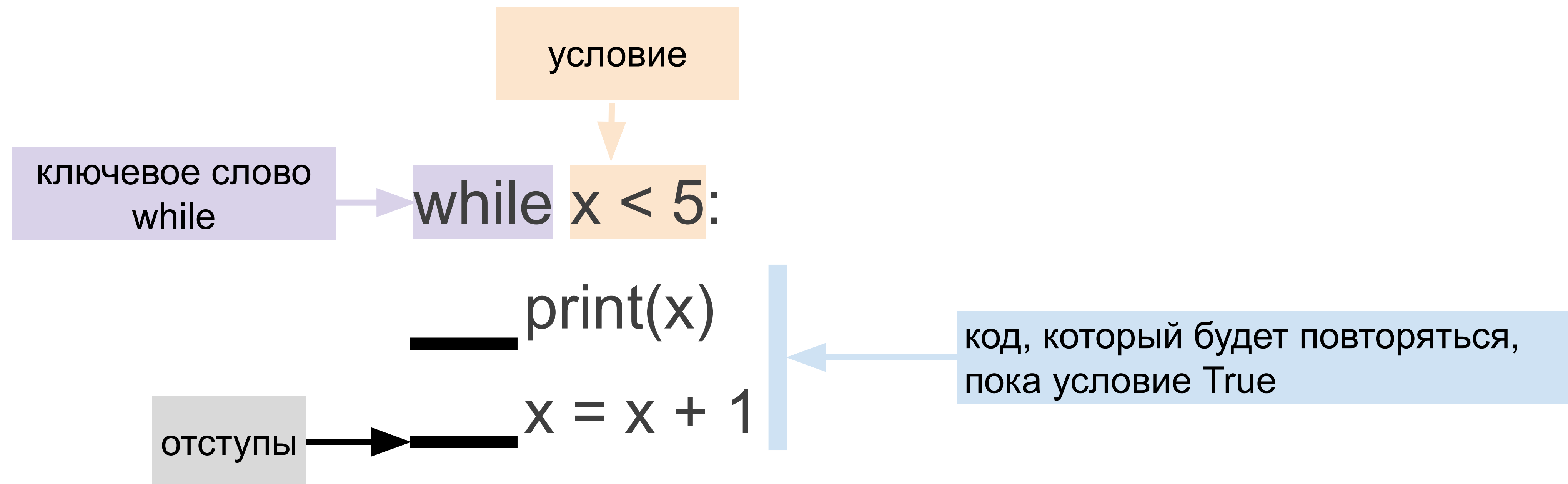
В Python существует два типа циклов:

- цикл **while**
- цикл **for**



Цикл while

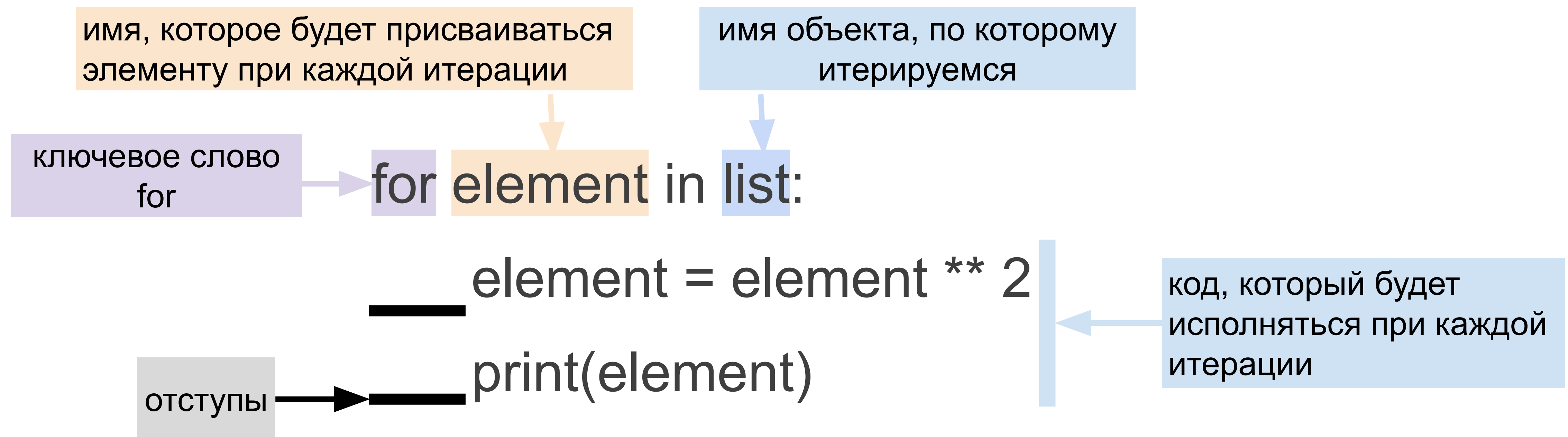
Позволяет выполнить одну и ту же последовательность действий, пока проверяемое условие истинно. Как правило, цикл **while** используется, когда невозможно заранее определить точное значение количества проходов исполнения цикла.





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*

- ***break*** прерывает исполнение цикла;
- ***continue*** завершает исполнение текущей итерации цикла и переходит к следующей итерации;
- ***pass*** игнорирует условие и продолжает исполнение цикла (ничего не делает; чаще всего используется в качестве заглушки при написании кода).



НЕТОЛОГИЯ
групп

Спасибо за внимание!

ОЛЕГ БУЛЫГИН



<https://www.facebook.com/obulygin91>



o.bulygin@netology.ru



<https://vk.com/obulygin91>



<https://www.linkedin.com/in/obulygin>