

# Введение в типы данных



Олег  
Булыгин



**Олег Булыгин**



# План занятия

1. [Простые типы данных](#)
2. [Индексация и срезы строк](#)
3. [Списки](#)
4. [Кортежи](#)



# 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.

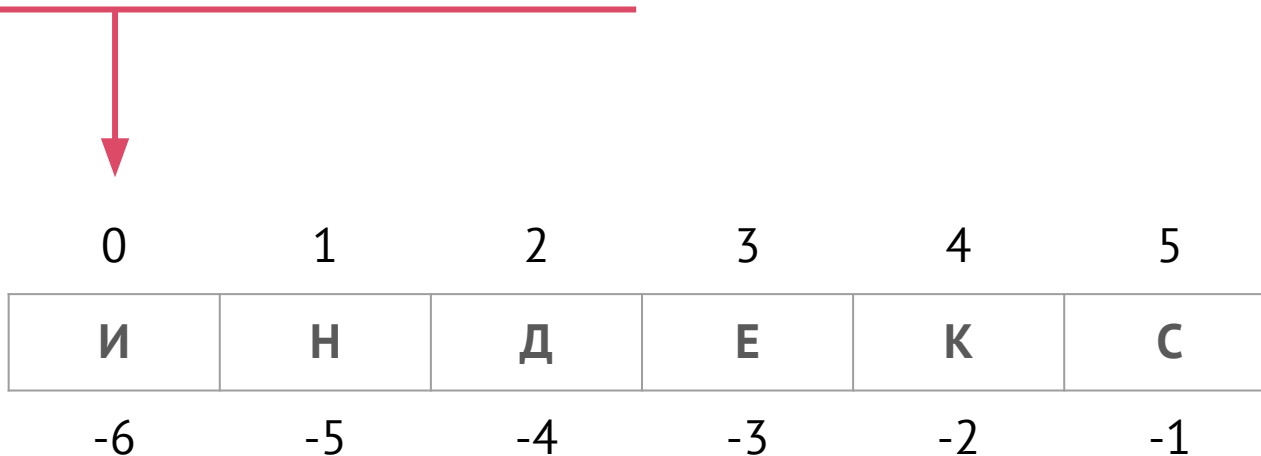


# Индексация и срезы строк

# Индексация и срезы строк

Доступ к элементам объекта по их порядковому номеру в нем.

Индексация элементов **начинается с нуля**.



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

## Поиск символов

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

`my_string[0]`

или

`my_string[-6]`

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



# 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).

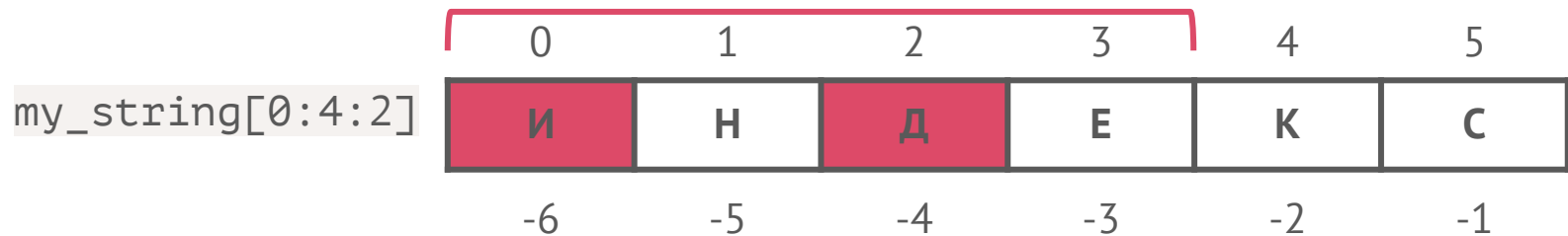
	0	1	2	3	4	5
<code>my_string[1:3]</code>	И	Н	Д	Е	К	С
	-6	-5	-4	-3	-2	-1

# 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:]
```

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

```
my_string[:3]
```

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



# 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.

`table = [ [1, 2, 3], [4, 5, 6] ]`

0 1 строки

0 1 2 0 1 2 столбцы

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).

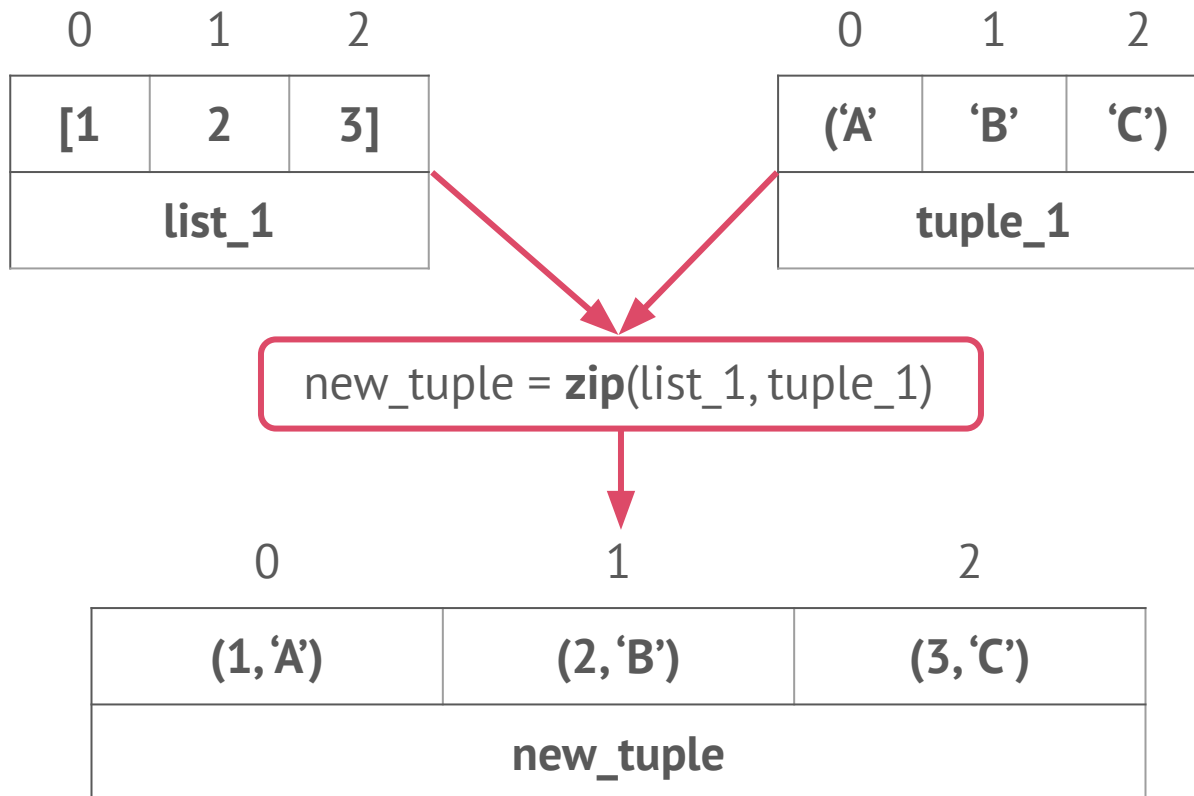
Кортежи инициализируются при помощи ( ).

```
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.

⌘ нетология

**Задавайте вопросы и  
пишите отзыв о лекции!**

Олег Булыгин