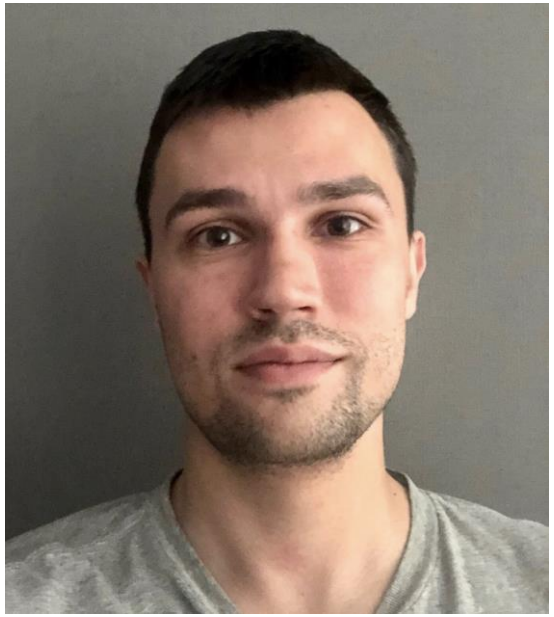
Python Starter

Списки и срезы

Introduction



Бондаренко Кирилл
Data scientist & Python developer

https://www.linkedin.com/in/kirill-bond/

https://medium.com/@bond.kirill.alexandrovich

Тема урока

Списки и срезы

План урока

1. Понятие списков
2. Срезы списков
3. Методы списков
4. Обход списков
5. Примеры работы со списками

Понятие списков

Списки – это упорядоченные коллекции объектов различных типов.

У каждого элемента списка есть его индекс, который отвечает его порядковому номеру в списке.

Индексация начинается с 0.

Для обращения к конкретному элементу списка нужно, после его названия, в квадратных скобках указать индекс элемента.

```
>>> my_list = list('abcd')
>>> print(my_list)
['a', 'b', 'c', 'd']
>>> my_list[0]
'a'
>>>
```

Срезы списков

Для того что бы делать выборку более чем по одному элементу, можно использовать срезы, которые вернут подмножество элементов списка.

Синтаксис: my_list[start:stop:step]

Start – индекс, с которого начинается подмножество.

Stop – индекс, перед которым закончится подмножество.

Step – шаг выбора элементов.

```
>>> my_list = list('abcd')
>>> print(my_list)
['a', 'b', 'c', 'd']
>>> my_list[0]
'a'
>>> my_list[0:1]
['a']
>>> my_list[0:5]
['a', 'b', 'c', 'd']
>>> my_list[0:5:2]
['a', 'c']
>>>
```

Методы списков

append() – добавление нового элемента в конец списка.

```
>>> my_list = [1]
>>> my_list.append('new_element')
>>> print(my_list)
[1, 'new_element']
>>>
```

clear() – удалить все элементы списка.

```
>>> my_list = [1]
>>> my_list.clear()
>>> print(my_list)
[]
>>>
```

Методы списков

extend() – расширяет список переданной последовательностью.

```
>>> first_list = [1,2,3]
>>> second_list = [4,5,6]
>>> first_list.extend(second_list)
>>> print(first_list)
[1, 2, 3, 4, 5, 6]
>>>
```

index() – возвращает index указанного элемента, если таких элементов несколько - вернет первый найденный.

```
>>> my_list = [1,1,2,3]
>>> my_list.index(1)
0
>>>
```

Методы списков

pop() – удаляет элемент указанного индекса и возвращает его.

```
>>> my_list = [1,1,2,3]
>>> removed_element = my_list.pop(0)
>>> print(removed_element)
1
>>> print(my_list)
[1, 2, 3]
>>>
```

reverse() – зеркально отображает список.

```
>>> my_list = [1,2,3]
>>> my_list.reverse()
>>> print(my_list)
[3, 2, 1]
>>>
```

Методы списков

sort() – сортирует список. Если указать в параметрах reverse=True, тогда сортировка будет в обратном порядке.

```
>>> my_list = [2,1,4,3]
>>> my_list.sort()
>>> print(my_list)
[1, 2, 3, 4]
>>>
>>> my_list = [2,1,4,3]
>>> my_list.sort(reverse=True)
>>> print(my_list)
[4, 3, 2, 1]
>>>
```

Обход списков

Списки можно обходить с помощью циклов.

Синтаксис для прохождения с помощью цикла for:

```
for elem in my_list:
    ...
```

```
>>> for elem in my_list:
...     print(elem)
...
2
1
4
3
>>>
```

Примеры работы со списками

1. Найти и удалить все четные элементы списка.
2. Возвести все элементы списка в квадрат.
3. Найти максимальный элемент списка.

Проверка знаний

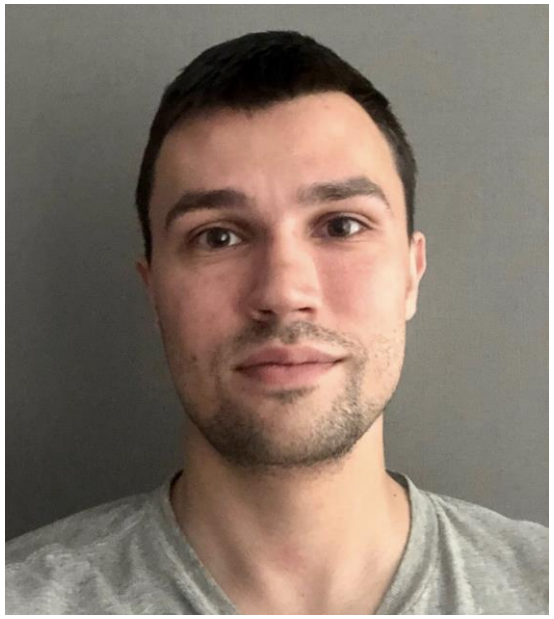
TestProvider.com

TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и для общей оценки знаний IT специалиста.

Проверьте как Вы усвоили данный материал на TestProvider.com

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

Спасибо за внимание! До новых встреч!



Бондаренко Кирилл
Data scientist & Python developer

Информационный видеосервис для разработчиков программного обеспечения

TestProvider Pearson CyberBionic systematics Microsoft Partner Windows Azure EBA