Списки

Date: 07/05/2024

Tochilkina Ksenya Stanislavovna

Списки

Что такое список?

Списки - упорядоченная изменяемая коллекция объектов разных типов. То есть, в список можно “положить” все что угодно: числа,строки, словари, кортежи, другие списки, значения которых могут повторяться. Для того, чтобы создать список, используем следующую запись:

```
my_list_1 = []
print(type(my_list_1))
```

Во многих языках программирования используется термин “массивы”, так вот в Python - это списки, с поправкой на то, что хранить в них можно любые типы данных. Если хотите, можете воспринимать список как корзину в магазине: в ней могут быть и продукты, и пылесос и другая корзина с прочими товарами. Однако чаще всего, списки содержат однотипные объекты:

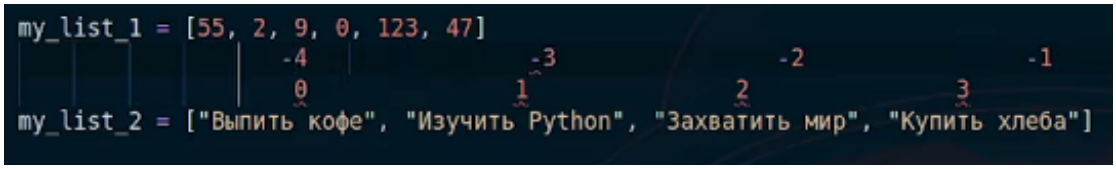
```
my_list_1 = [55, 2, 9, 0, 123, 47]

my_list_2 = ["Выпить кофе", "Изучить Python", "Захватить мир", "Купить хлеба"]
```

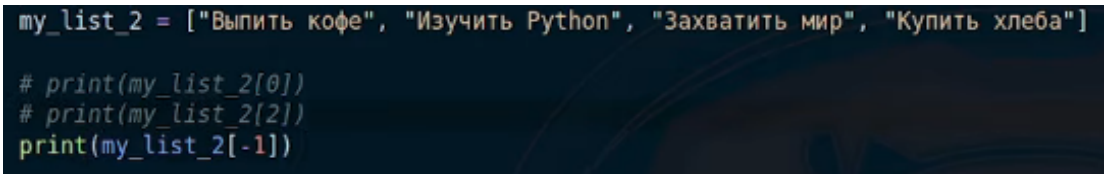
Индексация в списках

Списки в Python упорядочены, и у каждого элемента есть свой индекс - номер, используя который можно обращаться к элементу списка и выполнять над ним

необходимые операции. Нумерация списка начинается с нуля:



Для того, чтобы обратиться к значению элемента списка по индексу, необходимо указать имя списка, из которого хотим выбрать элемент, а затем номер элемента в этом списке:



Срезы в списках

Срез в Python - это извлечение одного или нескольких объектов из списка. Синтаксис для срезов в Python выглядит следующим образом: список[начало:конец:шаг].
Вот некоторые ключевые моменты о срезах в списках в Python:

- Начальный индекс включается в срез, а конечный индекс не включается. Например, список[0:3] вернет элементы с индексами 0, 1 и 2.
- Если начальный индекс не указан, срез начинается с первого элемента списка. Если конечный индекс не указан, срез идет до конца списка.
- Шаг определяет интервал между индексами элементов, которые будут включены в срез. Например, список[::2] вернет элементы с четными индексами.
- Отрицательные индексы также могут использоваться в срезах. Например, список[::-1] вернет список в обратном порядке.
- Срезы также могут применяться к строкам и кортежам в Python.

Вот примеры использования срезов в списках в Python:

```
my_list = [1, 2, 3, 4, 5]
sub_list = my_list[1:4]  # [2, 3, 4]
sub_list2 = my_list[2:]  # [3, 4, 5]
sub_list3 = my_list[:3]  # [1, 2, 3]
sub_list4 = my_list[::2]  # [1, 3, 5]
```

В этом примере мы имеем список my_list, содержащий числа от 1 до 5. Затем мы используем срезы, чтобы получить подмножества элементов списка. sub_list содержит элементы с индексами 1, 2 и 3. sub_list2 содержит элементы с индексами 2, 3 и 4, так как мы не указали конечный индекс. sub_list3 содержит элементы с

индексами 0, 1 и 2. sub_list4 содержит элементы с индексами 0, 2 и 4, так как мы указали шаг 2.

Основные метода для работы со списками

Основные методы для работы со списками в языке программирования Python включают:

- **append(element):** Добавляет элемент в конец списка.

- **extend(iterable):** Расширяет список, добавляя элементы из итерируемого объекта.

- **insert(index, element):** Вставляет элемент на указанную позицию в списке.

- **remove(element):** Удаляет первое вхождение указанного элемента из списка.

- **pop(index):** Удаляет и возвращает элемент на указанной позиции в списке.

- **index(element, start, end):** Возвращает индекс первого вхождения указанного элемента в списке в указанном диапазоне.

- **count(element):** Возвращает количество вхождений указанного элемента в списке.

- **sort():** Сортирует список в порядке возрастания.

- **reverse():** Изменяет порядок элементов в списке на обратный.

- **copy():** Создает копию списка.

Вот пример использования некоторых из этих методов:

```python
my_list = [1, 2, 3, 4, 5]

my_list.append(6)  # [1, 2, 3, 4, 5, 6]
my_list.extend([7, 8, 9])  # [1, 2, 3, 4, 5, 6, 7, 8, 9]
my_list.insert(0, 0)  # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
my_list.remove(3)  # [0, 1, 2, 4, 5, 6, 7, 8, 9]
my_list.pop(2)  # [0, 1, 4, 5, 6, 7, 8, 9]
index = my_list.index(5)  # index = 3
count = my_list.count(4)  # count = 1
my_list.sort()  # [0, 1, 4, 5, 6, 7, 8, 9]
my_list.reverse()  # [9, 8, 7, 6, 5, 4, 1, 0]
my_list_copy = my_list.copy()  # создание копии списка
```