Теоретический материал к занятию Коллекции и их методы. Урок 1

В Python для того, чтобы работать с большим количеством данных, используется структура данных под названием **список**.

В информатике структура данных (data structure) — программная единица, позволяющая хранить и обрабатывать множество однотипных и/или логически связанных данных.

Так вот, список представляет собой последовательность (набор) элементов, пронумерованных от 0, как символы в строке.

Чтобы создать список нужно всего лишь в квадратных скобках перечислить его элементы через запятую:

```
numbers = [2, 4, 6, 8, 10]
languages = ['Python', 'C#', 'C++', 'Java']
```

Мы создали два списка. Первый называется **numbers** и содержит 5 элементов. Второй называется **languages** и содержит 4 элемента.

Списки могут состоять из данных абсолютно разных типов .

```
info = ['Python', 2022, 24.2]
```

Создать пустой список можно двумя способами:

1. Использовать пустые квадратные скобки [];
2. Использовать встроенную функцию list().

Следующие две строки кода создают пустой список:

```
mylist = []      # пустой список
mylist = list()  # пустой список
```

Вывести список можно с помощью команды print()

```
numbers = [2, 4, 6, 8, 10]
print(numbers)
```

Функция list() с пустыми скобками создает пустой список. Она также может создать список из последовательности чисел, созданной другой встроенной функцией — range().

```
numbers = list(range(5))
```

Функция list() позволяет создать список и из символов строки.

```
s = 'abcde'
chars = list(s)
```

Сходства и отличия списков и строк

Сходства:

- функция len() для подсчета длины списка, как у строк;
- оператор принадлежности in, как и у строк;
- индексация, то есть, обращение по индексу. Также начинается с нуля, и возможна отрицательная индексация с конца списка, как и у строк;
- срезы позволяют получать подсписки исходных списков, как у строк;
- операция конкатенации и умножения на число, работает для списков также, как и для строк;
- функции min() и max() со списками работают отлично, и позволяют не писать самостоятельно код поиска наибольшего и наименьшего элемента списка;

Различия:

- списки **изменяемы**, иногда их называют мутирующими коллекциями;
- функция sum(), находящая сумму элементов списков, отсутствует у строк.

Основные методы списков:

- append - добавить в конец списка
- remove - удалить первый совпадающий элемент списка
- pop - удалить элемент по индексу с возможностью сохранить значение в переменной
- clear - очистить список
- sort - отсортировать список
- extend - объединить списки
- index - найти индекс элемента
- insert - вставить элемент по индексу

1. **append**

```
a = [1, 2, 3]
a.append(4)
print(a)
```

Результат

```
[1, 2, 3, 4]
```

2. **remove**

```
a = [1, 2, 3]
a.remove(2)
print(a)
```

Результат

```
[1, 3]
```

3. **pop**

```
a = [1, 2, 3]
a.pop(2)
print(a)
```

Результат

```
[1, 2]
```

```
a = [1, 2, 3]
b = a.pop(2)
print(a)
print(b)
```

Результат

```
[1, 2]
3
```

4. **clear**

```
a = [1, 2, 3]
a.clear()
print(a)
```

Результат

```
[]
```

5. **sort**

```
a = [2, 1, 3]
a.sort()
print(a)
```

Результат

```
[1, 2, 3]
```

6. **extend**

```
a = [1, 2, 3]
```

```
b = [3, 5, 6]
a.extend(b)
print(a)
```

Результат

```
[1, 2, 3, 3, 5, 6]
```

7. **index**

```
a = [2, 1, 3]
print(a.index(3))
```

Результат

```
[1, 2, 3]
```

8. **insert**

```
a = [2, 1, 3]
a.insert(0, 4)
print(a)
```

Результат

```
[4, 2, 1, 3]
```

Кортежи очень похожи со списками, единственное кортежи неизменяемы. Т.е мы не можем совершать операции изменяющие элементы кортежа. В остальном работа такая же как и со списками(нахождения индекса элемента, длины кортежа, обращение по индексу и т.д).

Создается кортеж с помощью круглых скобок ()

```
a = ()  # для создания пустого кортежа
a = (2,)  # для создания кортежа из одного элемента
```

Если же мы хотим изменить кортеж, мы можем преобразовать его в список, изменить, а потом преобразовать обратно в кортеж

```
a = (1, 2)
a = list(a)
a.append(3)
a = tuple(a)
print(a)
```