Методические указания

Урок 6.2 Коллекции и их методы

Задачи урока:

- Глубже познакомиться со списками
- Познакомиться с кортежами

0. Подготовка к уроку

До начала урока преподавателю необходимо:

1) Просмотреть, как ученики справились с домашним заданием
2) Прочитать методичку

1. Основы работы со списками

Учитель: Напомню, списки представляют собой наборы данных, как правило, но не всегда, однотипных. Сегодня мы пойдем дальше и поговорим о сходстве списков и строк.
Действительно, строки и списки очень похожи, ведь по сути строки — наборы символов, как бы подмножества списков. Но если списки могут содержать абсолютно разные элементы, скажем, целые числа, вещественные числа, строки, да и любой другой тип данных, то строки всегда содержат только символы.

Хорошая новость заключается в том, что, поскольку строки — частный случай списков, то весь функционал для строк доступен и при работе с списками. Это круто, своего рода двукратное использование наших знаний, а точнее, их адаптация под новый материал. Конечно у списков и строк есть и отличия.

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.

2. Методы списков

Учитель: При изучении строк мы говорили о строковых методах. Напомню, методы это специальные функции, применимые к определенным типам данных. В частности, строковые методы вызываются именно для переменных, обозначающих строки. Метод, по сути, есть функция для выполнения часто встречающейся задачи.

Методы есть и у списков. Если вспомните предыдущие уроки, то, возможно, заметите, что мы всегда работали с уже созданным списком, другими словами, со статическим списком. Но ведь нужно уметь и конструировать списки самостоятельно. К примеру, мы можем считать данные от пользователя и добавить их в список. Или удалить какие-то элементы из списка.

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

Конечно же, это не все методы, которые существуют(о них я вам советую почитать отдельно). Давайте рассмотрим как они работают на практике

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]
```

4. Кортежи

Учитель: 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)
```

5. Решение задач

Задача 1

Сформировать возрастающий список из **чётных чисел** (количество элементов **45**);

Решение

```
a = []
i = 0
while len(a) < 45:
    if i % 2 == 0:
        a.append(i)
    i += 1
print(a)
```

Дополнительно

Если на уроке остается время, то ученикам можно предложить начать прорешивать домашнее задание.

Домашняя работа

Задача 1
Дан список:
a = [1, 2, 3, 4, 5, 6]
Требуется удалить из списка все нечетные элементы, а четные разделить нацело на два.
Новый список создавать нельзя

Решение

```
a = [1, 2, 3, 4, 5, 6]
i = 0
n = len(a)
while i < n:
    if a[i] % 2 == 0:
        a[i] //= 2
        i += 1
    else:
        a.pop(i)
        n -= 1
print(a)
```