Методические указания

Урок 6.1 Коллекции и их методы.

Задачи урока:

- Познакомиться со списками

0. Подготовка к уроку

До начала урока преподавателю необходимо:

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

1. Введение в списки

Учитель: Ребята, сегодня мы начинаем новый модуль, который посвящен **спискам**.
Представьте себе ситуацию, когда программе требуется работать с двумя значениями. В таком случае можно присвоить эти два значения двум переменным.
Если значений будет 5, нам не составит труда объявить 5 переменных. Но что делать если нужно работать с большим количеством данных? Скажем со 100 значениями. Не будем же мы создавать 100 никак не связанных переменных, правда? Обычно, когда мы работаем с данными в таких количествах, они представляют собой нечто единое, другими словами, у них есть что-то общее, поэтому разумно с ними работать через одну переменную.
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. Забегая немного вперед, сразу скажу, что списки очень похожи на строки.

Давайте посмотрим, как создать список. Для этого 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.
В списках, как и в строках каждый элемент имеет номер и нумерация начинается с 0. Первый элемент имеет номер или индекс равный 0, второй — 1 и т.д.
Чтобы обратиться к элементам списка, мы так же, как при работе со строками, используем квадратные скобки.

Список **numbers** состоит из 5 элементов, и каждый из них - целое число.

- numbers[0] == 2;
- numbers[1] == 4;
- numbers[2] == 6;
- numbers[3] == 8;
- numbers[4] == 10.

Список **languages** состоит из 4 элементов, каждый — **строка.**

- languages[0] == 'Python';
- languages[1] == 'C#';
- languages[2] == 'C++';
- languages[3] == 'Java'.

Таким образом, вместо создания 5 несвязанных переменных **num1, num2, num3, num4, num5**, мы создали 1 переменную, имеющую тип “список”. В нем содержится 5 элементов, и к ним можно обратиться так: **numbers[0], numbers[1], numbers[2], numbers[3], numbers[4]** .

Эти два примера демонстрируют списки, содержащие один тип данных, однако списки могут состоять из данных абсолютно разных типов . Например:

info = ['Python', 2022, 24.2]

Список **info** содержит строковое значение, целое число и число с плавающей точкой.

- info[0] == 'Python';
- info[1] == 2022;
- info[2] == 24.2

При изучении целых чисел и строк, мы говорили про особые начальные значения. Так, например, численным переменным обычно в самом начале присваивают значение 0, а строковым — значение пустой строки. Аналогично для начальной инициализации списочной переменной используется так называемый пустой список, то есть список без элементов.

Пустой список

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
```

Вывод содержимого списка

Раньше мы пользовались функций **print()** для вывода на экран значения переменной, будь то число или строка. Точно также можно вывести содержимое целого списка:

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

Функция **print()** выводит на экран элементы списка в квадратных скобках, разделенные запятыми:

[2, 4, 6, 8, 10]

['Python', 'C#', 'C++', 'Java']

Обратите внимание, что вывод списка содержит квадратные скобки. Позже мы научимся выводить элементы списка в более удобном виде с помощью циклов.

Функция list()

При работе со списками очень часто приходится использовать встроенную функцию list().

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

Следующий код

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

создает список [0, 1, 2, 3, 4]

Можно создать список только из четных или нечетных чисел:

```
even_numbers = list(range(0, 10, 2))
odd_numbers = list(range(1, 10, 2))
```

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

Для преобразования строки в список мы пишем следующий код:

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

В результате выполнения такого кода в переменной chars будет хранится список символов: `['a', 'b', 'c', 'd', 'e']`

Учитель: А теперь давайте устно подведем небольшой итог по работе со списками

1. **Что такое список? Какую задачу решают списки?**
   Список – структура данных, позволяющая хранить и обрабатывать множество однотипных и/или логически связанных данных.
2. **Когда нужно использовать списки? Можно ли обойтись без списков?**
   Если программа работает с большим количеством данных (набором данных), то обойтись без списков практические невозможно. Можно создать отдельную переменную на каждое значение, однако невозможно одинаково обрабатывать такие данные в цикле.
3. **Как обращаться к элементам списка?**
   К элементам списка обращаются по их индексам, то есть по их номерам. Нумерация начинается с нуля. Возможна отрицательная нумерация с конца списка, и это исключительно фича (функционал) Python.
4. **Могут ли списки в Python содержать значения разных типов данных?**
   Да, могут, например [1, 2.6, 'hello', True]. Однако обычно при написании программ в списках хранятся значения одного типа данных: список чисел, список строк и т.д.

Учитель: Списки также могут быть вложенными. Например список может содержать в качестве элементов другие списки или иные типы данных, позволяющих хранить более одного значение(кортежи, словари, множества)

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

Для доступа к необходимым данным, в данном случае требуется указать уже не 1, а 2 индекса в квадратных скобках. Например выведем число 3

```
numbers = [[1, 2, 3], [4, 5, 6]]
print(numbers[0][2])
```

Учитель: Кстати для перебора списка мы можем воспользоваться также циклами, как и при работе со строками

4. Решение задач

Задача 1

Дополнительно

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

Домашняя работа

Задача 1