Лекция 5

Списки

Списки

Список — упорядоченная коллекция элементов. Т.е. списки - это элементы-контейнеры, которые предназначены для хранения одного или более значений.

В качестве элементов списка могут выступать переменные или литералы любых типов. Доступ к каждому такому элементу осуществляется по его порядковому номеру — индексу.

Список является динамической структурой данных. Это означает, что вы можете в любой момент добавить или удалить элементы списка.

Как создать список в Python

Для создания пустого списка (т. е. списка, в котором нет ни одного элемента) в Python можно использовать такой синтаксис:
имя_списка = []

Такой код создаст вам пустой список. Если же вам нужно создать список, в котором сразу будет несколько значений, то можно использовать синтаксис вида:

имя_списка = [значение_1, значение_2, значение_3,...]

Внимание! Важно понимание того, что в Python эти значения могут быть произвольных и, в общем случае, разных типов.

python

Пример объявления пустого списка и списка с элемен

Создание пустого списка

```
my_list = []
print(my_list)
my_list = [1, 2, "Hello"]
print(my_list)
```

Создание списка, в котором уже есть элементы

Доступ к элементам списка

Для того, чтобы обратиться к элементу списка, используется его индекс. Индекс — это порядковый номер элемента списка. Индексы начинаются с 0 и возрастают с каждым последующим шагом на единицу.

```
my_list = [1, 2, "Hello"]
```

0 1 2 - индексы элементов

Для доступа к элементу списка нужно указать имя списка, а после него индекс элемента в квадратных скобках. Например, чтобы получить доступ к элементу с индексом 2, нужно написать

```
my_list [2]
```

Внимание! При доступе к несуществующему индексу, ваша программа завершится с ошибкой!

python

Как добавить элемент в список

Для того, чтобы добавить элемент в конец списка, нужно написать имя списка, после этого поставить точку и написать **append()**. В скобках указать, что вы хотите добавить в конец списка.

Например, создадим список из трех элементов, а потом добавим в него еще один элемент.

```
my_list = [5, 7, 9]

print(my_list)

my_list.append(13)

print(my_list)
```

Создание списка

Добавление в конец списка элемента (число 13)

Как удалить элемент из списка

Для того, чтобы удалить элемент списка, используется его индекс и оператор **del**.

Синтаксис этого оператора :

```
del (list_name[index])
```

list_name — имя списка, из которого вы хотите удалить элемент

index — индекс элемента (число), который вы хотите удалить

python

Пример удаления элемента из списка

Создание списка

```
my_list = [2,4,7,11,0,-2,8]

del(my_list[1])

print(my_list)
```

Удаление элемента с индексом 1

В этом примере из списка удаляется элемент с индексом 1. После этого вывод на экран будет такой [2, 7, 11, 0, -2, 8]. Т.е. элемент с индексом 1 был удален.

Цикл **for** в Python

Предположим, что вам нужно пройтись по всем элементам списка. Например, сложить их. Если список небольшой (5 - 10) элементов, то не трудно вручную указать все индексы. Однако если список большой, то лучше для этой цели использовать цикл **for**.

```
for имя_итерационной_переменной in имя_списка:
    Операторы для работы с итерационной_переменной
```

На каждой итерации такого цикла итерационная переменная принимает значение текущего элемента списка и будет сдвинута на следующий элемент во время следующей итерации. Таким образом этот цикл по очереди переберет все элементы списка.

python

Пример того, как с помощью цикла **for** пройти по списку

Создание списка

```
my_list = [5, 7, 9, 0, 11]

for element in my_list:
    print(element)
```

Цикл для прохода по всем элементам этого списка

При каждой итерации такого цикла переменная с именем element будет равна по очереди каждому элементу этого списка, и тогда цикл по очереди переберет все элементы списка.

python

Генератор числовой последовательности **range**

Я хочу чтобы переменная цикла принимала элемент числовой последовательности с нужным мне шагом. Например ряд чисел 1, 2, 3, 4 ... Как это реализовать в Python?

Для этого в Python используется оператор **range**, который и возвращает числовую последовательность с нужным шагом. Его синтаксис таков:

range(с какого числа начинать, до какого числа, с каким шагом)

Внимание! Последнее число не включается в результирующую последовательность. Задание шага можно опустить и в таком случае шаг будет равен 1. Задание начального значения также можно опустить в так случае он будет равен 0.

По полученной последовательности можно пройти с помощью цикла **for** как по списку.

Пример использования генератора **range**

Предположим, что вам нужно пройтись по всем элементам числовой последовательности от 1 до 10 с шагом 2. В таком случае вы могли бы использовать код вида:

```
for i in
range(1,10,2):
    print(i)
```

Тут переменная с именем i примет по очереди значения числовой последовательности от 1 до 9 (не забывайте, что последнее число в диапазон не включается, поэтому, хоть и указанно, что последнее число 10, в результате все закончится на 9) с шагом в 2. На экране будет последовательность: 1, 3, 5, 7, 9.

Новая возможность оператора **if** — проверка на вхождение

Я хочу узнать, есть ли определенное значение в списке или нет. Мне нужно перебрать весь список, и каждый его элемент сравнить с тем, что я ищу. Или есть более короткий и удобный способ?

Такой способ есть и называется проверка на вхождение элемента в список. Для этого используется оператор **in**. Его синтаксис таков:

element **in** list

Такое выражение вернет True, если element действительно есть в списке (list) и False, если его там нет.

python

Пример использования оператора **in**

Создадим заполненный список и узнаем, есть ли в нем тот или иной элемент или нет.

```
my_list = [0,4,5,6,9]

if 4 in my_list:
    print("4 in list")

if 10 in my_list:
    print("10 in list")
```

Создание списка

Проверка, есть ли элемент 4 в этом списке (есть, поэтому True)

Проверка, есть ли элемент 10 в этом списке (нет, поэтому False)

Как узнать длину списка

Для того, чтобы узнать длину списка (т. е. сколько в нем сейчас элементов), можно использовать оператор **len**

Его синтаксис таков:

```
len(имя_списка)
```

Сам оператор вернет число, равное количеству элементов в списке.

Пример использования оператора **len**

Создание списка

```
my_list = [0,4,5,6,9]

print(len(my_list))
```

Получение длины списка

python

Практический пример использования полученных зна

Поставим перед собой задачу: заполнить список десятью случайными числами от 1 до 100 и посчитать их сумму.

```
import random

my_list = []

for i in range(10):
    my_list.append(random.randint(1,100))
print(my_list)

summa = 0
for element in my_list:
    summa = summa + element
print(summa)
```

Импорт генератора случайных чисел

Создание пустого списка

Проход по последовательности от 0 до 10 (т.е. 10 раз).

Добавление случайного элемента в список

Проход по списку

Накопление суммы элементов в переменной summa

Какие еще возможности есть у списков?

На самом деле у списков еще много возможностей. Так что, самое время продвинуться дальше в их изучении.

Например, списки можно складывать. Результатом сложения будет список, в котором будут присутствовать элементы первого и, второго списков.

```
first_list = [2,4,7]
second_list = [3, 5, 8]
my_list = first_list + second_list
print(my_list)
```

В результирующем списке будет элементы и из первого и из второго списка. На экран будет выведено: [2, 4, 7, 3, 5, 8].

python

Умножение списка на целое число

Если умножить список на целое число, то в результате получим список, в котором элементы базового списка повторятся столько раз, на значение какого числа вы умножили.

```
first_list = [2,4,7]
my_list = first_list *
3
print(my_list)
```

В результирующем списке будут присутствовать элементы, повторенные 3 раза.

Поэтому на экран будет выведено: [2, 4, 7, 2, 4, 7, 2, 4, 7].

Использование срезов списков

Как уже было сказано, обратиться к элементу списка можно по его индексу. Однако также можно обратится и к нескольким элементам списка одновременно.

Такая возможность называется использованием **срезов**.

Для получения среза списка достаточно указать имя списка, после которого, в квадратных скобках, указать два, числа разделенных двоеточием. Первое число - это начальный индекс среза, второе число - конечный индекс среза. Начальный индекс может отсутствовать и, в таком случае, он равен 0. Также может отсутствовать и конечный индекс — и ,в таком случае, он равен концу списка.

Внимание! В качестве индексов могут использоваться и отрицательные целые числа. Это означает, что отсчет начинается не от начала, а с конца списка.

Пример получения среза списка

```
first_list = [2,4,7,11,0,-2,8]

my_list = first_list [3:6]

print(my_list)
```

Получение среза списка

В результирующем списке будут элементы базового списка с 3-его по 5-ый индексы (Внимание! 6-ой индекс включен не будет, хотя он и указан). Поэтому результатом работы приложения будет: [11, 0, -2].

Присвоение значений срезам

Если срезу списка присвоить значение, то это значение отобразится как содержимое вашего списка. Однако, если размер среза, стоящего слева от оператора = ,больше, чем то, что стоит справа от оператора =, то список будет уменьшен.

Как происходит присвоение срезу?

При присвоении значения срезу списка выполняется два этапа:

1. **Удаление**. Раздел списка, определяемый слева от оператора =, удаляется.
2. **Вставка**. Новые элементы, содержащиеся в объекте, расположенном справа от оператора =, вставляются в список, начиная с левого края, где находился прежний удаленный срез.

Пример присвоения значения срезу списка

```
my_list = [2,4,7,11,0,-2,8]

my_list[0:2] = [1]

print(my_list)
```

В этом примере срезу с 0-го по 2-ой индексы присвоен список из одного элемента. Получилось, что сначала кусок списка с 0-го по 2-ой индексы были удалены. Впоследствии на их место был поставлен только один элемент равный 1.

Вывод на экран этого примера таков: [1, 7, 11, 0, -2, 8].

Дополнительные методы списков

Описание метода	Действие
append(element)	Добавит element в конец списка.
insert(index, element)	Вставит element на место index списка.
extend(list)	Добавит список list в конец списка.
pop(index)	Удаляет и возвращает элемент с индексом index. Если index не указан, то удалится последний элемент.
count(element)	Вернет сколько раз element встречается в списке.
clear()	Очистит список.
sort()	Сортирует список по возрастанию.
reverse()	Разворот списка.
index(element)	Вернет индекс элемента, на котором стоит element.
remove(element)	Удалит element из списка.

Как использовать методы списков

Я прочел таблицу с описанием методов для работы со списками. Но как ими пользоваться?

Для того, чтобы использовать тот или иной метод списка, нужно указать имя списка, потом поставить точку и после него написать имя метода. В общем случае синтаксис примерно таков:

```
list_name.method_name()
```

Например, нужно добавить в строку "Hello" на позицию с индексом 2 в список с именем my_list.

```
my_list.insert(2, "Hello")
```

Вопросы о сортировке списков

Списки сортируются по возрастанию. Могу ли я отсортировать их по убыванию?

Да. Для этого нужно в скобках метода sort() написать reverse=True

Например:

```
my_list = [2,4,7,11,0,-2,8]
my_list.sort(reverse=True)
print(my_list)
```

А могу я сам определять, по какому критерию задавать направление сортировки?

Да, можно. Но, для этого нужно знать понятие функции. Его мы будем изучать позднее.

Список списков

В других языках (Си, Pascal и т. д.) есть двумерные массивы (матрицы). Но в Python нет массивов! Как описать матрицу тут?

Да, действительно, в Python нет массивов. Поэтому существует два пути создания объектов, похожих на матрицы, в Python:

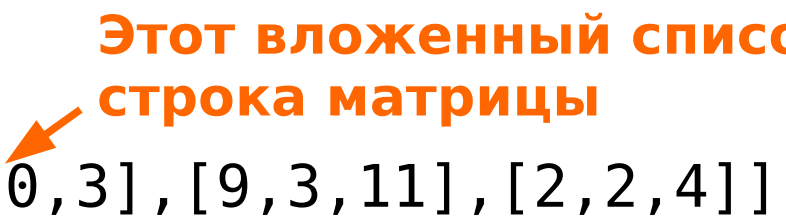
1) Использование дополнительной библиотеки Python — **Numpy**. Эта библиотека использует матрицы для вычислений и она активно применяется в научной работе.
2) Использование списка списков. Т.е. такой список, каждый элемент которого, в свою очередь, является тоже списком.

Оставим пока наукоемкие технологии в покое. И рассмотрим, как создать список списков.

Создание списка списков

На самом деле, создать список списков очень просто. Нужно вместо обычных чисел, как вы это делали в обычном списке, просто поставить список, который будет описывать строку матрицы.

Например, так:




```
my_matrix = [[1,0,3],[9,3,11],[2,2,4]]
```

В этом примере создана матрица размером 3 X 3 (три строчки по три столбца в каждой).

Как обратиться к элементу списка списков

Как обратиться к элементу списка понятно (по индексу), а как обратится к элементу списка, который состоит из списков?

Для обращения используйте такую нотацию:

list_name[row_index][col_index]

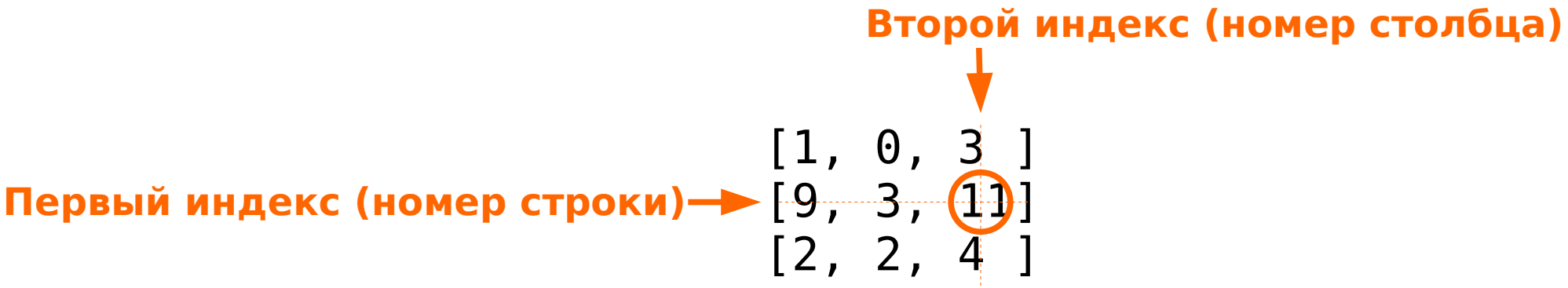
list_name — имя списка
col_index — индекс, на котором расположен вложенный список (можете представить, как индекс столбца матрицы)
row_index — индекс элемента во вложенном списке (можете представлять как номер строки матрицы)

python

Пример обращения к элементу списка списков

```
my_matrix = [[1, 0, 3], [9, 3, 11], [2,
2, 4]]
a = my_matrix[1][2]
print(a)
```

Как можно себе представить список списков



Тут первый индекс указывает на индекс, на котором стоит список, который представляет строку, а второй указывает на элемент в этом списке.

Пример создания и заполнения списка списков

```
import random

matrix = []
for i in range(5):
    col = []
    for j in range(4):

col.append(random.randint(0,100))
    matrix.append(col)

print(matrix)
```

Этот список будем заполнять списками

Это вложенный список

Список использованной литературы

1) Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011 - 133 - 137 с.
2) Саммерфилд М. Программирование на Python 3. Подробное руководство. - Пер. с англ. - СПб.:Символ-Плюс, 2009. - 135 - 142 с.

python

Домашнее задание

1) Дан список [0,5,2,4,7,1,3,19]. Написать программу для подсчета нечетных цифр в нем.
2) Создайте список случайных чисел (размером 4 элемента). Создайте второй список в два раза больше первого, где первые 4 элемента должны быть равны элементам первого списка, а остальные элементы заполните удвоенными значением начальных. Например,
   Было → [1,4,7,2]
   Стало → [1,4,7,2,2,8,14,4]
3) Создайте список из 12 элементов. Каждый элемент этого списка представляет собой зарплату рабочего за месяц (случайное число от 7500 до 15000 грн.). Выведите этот список на экран и вычислите среднемесячную зарплату этого рабочего.
4) Представьте в виде списка списков матрицу

   [1, 2 , 3, 4]
   [5, 6 , 7, 8]
   [9,10, 11, 12]
   [13,14, 15, 16]

   Напишите программу, которая выведет эту матрицу на экран, вычислит и выведет сумму элементов этой матрицы.

python

Дополнительное домашнее задание

1)«Перевернуть матрицу». Т.е. написать программу, которая повернет базовую матрицу на 90 градусов по часовой стрелке.

Было	Стало
[1, 2, 3, 4, 5, 6]	[1, 1, 1, 1, 1, 1]
[1, 2, 3, 4, 5, 6]	[2, 2, 2, 2, 2, 2]
[1, 2, 3, 4, 5, 6]	[3, 3, 3, 3, 3, 3]
[1, 2, 3, 4, 5, 6]	[4, 4, 4, 4, 4, 4]
[1, 2, 3, 4, 5, 6]	[5, 5, 5, 5, 5, 5]
[1, 2, 3, 4, 5, 6]	[6, 6, 6, 6, 6, 6]

2)Написать код для зеркального переворота списка [7,2,9,4] -> [4,9,2,7]. Список может быть произвольной длины. (При выполнении задания использовать дополнительный список нельзя).