Алексей Васильев

Язык программирования Python

Киев 2019

Лекция 3. Списки и кортежи

- Создание списка
- Знакомство с кортежами
- Доступ к элементам
- Выполнение среза
- Операции со списками и кортежами

Создание списка

Список (тип list) - упорядоченный набор значений (элементов)

Создание списка:

- **Перечисление элементов в квадратных скобках:**
  nums=[1,3,5]
- **С помощью функции list():**
  nums=list(1,3,5)
- **Размер списка определяется с помощью функции len()**

- Обращение к элементу списка - по индексу
- Индексация начинается с нуля
- Отрицательный индекс - отсчет с конца списка

Например:

len(nums) - размер списка

nums[0] - первый элемент

nums[-1] - последний элемент

Создание среза

Два индекса:

- список[i:j+1] - список из элементов с индексами от i до j включительно
- список[i:] - список из элементов с индексами от i и до конца исходного списка
- список[:j+1] - список из элементов, начиная с первого и до элемента с индексом j
- список[:] - копия списка

Три индекса:

- список[i:j+1:k] - список из элементов с индексами от i и меньше j+1 с интервалом k
- если k<0 - выборка в обратном порядке
- список[::-1] - копия списка в обратном порядке

Создание списка

Программа (UsingList.py)

```
# Список из чисел:
nums=[5,10,1,60,25,3]
# Отображение содержимого списка:
print("Список из чисел:",nums)
# Длина списка:
print("Длина списка:",len(nums))
# Первый элемент:
print("Первый элемент:",nums[0])
# Последний элемент:
print("Последний элемент:",nums[-1])
# Наибольшее значение:
print("Наибольшее значение:",max(nums))
# Наименьшее значение:
print("Наименьшее значение:",min(nums))
# Сумма:
print("Сумма:",sum(nums))
# Список в обратном порядке:
print("Список в обратном порядке:",list(reversed(nums)))
# Сортировка по возрастанию значений:
print("По возрастанию значений:",sorted(nums))
# Сортировка по убыванию значений:
print("По убыванию значений:",sorted(nums,reverse=True))
# Исходный список:
print("Исходный список:",nums)
# Изменение значения элемента списка:
nums[1]="текст"
# Отображение содержимого списка:
print("После внесения изменений:",nums)
```

Создание списка

Программа (UsingList.py) - продолжение

```
# Получение среза:
print("Получение среза:",nums[1:len(nums)-1])
# Замена части элементов списка:
nums[1:-1]=["A","B"]
# Список после замены элементов:
print("После замены элементов:",nums)
# Список чисел от 5 до 10:
nums=list(range(5,11))
print("Список чисел от 5 до 10:",nums)
# Удаление элементов из списка:
nums[2:4]=[]
print("После удаления двух элементов:",nums)
# Удаление последнего элемента:
del nums[len(nums)-1]
print("Удален последний элемент:",nums)
# Нечетные числа:
nums=[2*k+1 for k in range(5)]
print("Нечетные числа:",nums)
# Список из символов создается на основе текста:
symbs=list("Python")
# Отображение содержимого списка:
print("Список из символов:",symbs)
# Два первых символа:
print("Два первых символа:",symbs[:2])
# Прочие символы:
print("Остальные символы:",symbs[2:])
```

Создание списка

Результат выполнения:

```
Список из чисел: [5, 10, 1, 60, 25, 3]
Длина списка: 6
Первый элемент: 5
Последний элемент: 3
Наибольшее значение: 60
Наименьшее значение: 1
Сумма: 104
Список в обратном порядке: [3, 25, 60, 1, 10, 5]
По возрастанию значений: [1, 3, 5, 10, 25, 60]
По убыванию значений: [60, 25, 10, 5, 3, 1]
Исходный список: [5, 10, 1, 60, 25, 3]
После внесения изменений: [5, 'текст', 1, 60, 25, 3]
Получение среза: ['текст', 1, 60, 25]
После замены элементов: [5, 'A', 'B', 3]
Список чисел от 5 до 10: [5, 6, 7, 8, 9, 10]
После удаления двух элементов: [5, 6, 9, 10]
Удален последний элемент: [5, 6, 9]
Нечетные числа: [1, 3, 5, 7, 9]
Список из символов: ['P', 'y', 't', 'h', 'o', 'n']
Два первых символа: ['P', 'y']
Остальные символы: ['t', 'h', 'o', 'n']
```

Задание - 1

Напишите программу, в которой создается и отображается список, содержащий степени двойки (числа 1, 2, 4, 8, и так далее)

Задание - 2

Напишите программу, в которой создается список из чисел, которые при делении на 5 дают в остатке 3 (такие числа вычисляются по формуле 5k + 3, где k = 0,1,2,...). Отобразить этот список в прямом и обратном порядке

Создание кортежа

Кортеж (тип tuple) - неизменяемый упорядоченный набор значений (элементов)

Создание кортежа:

- Перечисление элементов в круглых скобках:
  A=(1,3,5) или A=() или A=(100,)
- Круглые скобки можно не использовать:
  A=1,3,5 или A=100,
- С помощью функции tuple():
  A=tuple([1,3,5]) или A=tuple("Python")
- Содержимое кортежа изменить нельзя
- Размер кортежа определяется с помощью функции len()
- Содержимое кортежа изменить нельзя
- Доступ к элементам кортежа - с помощью индекса
- Для кортежа можно выполнять срез

Создание кортежа

Программа (UsingTuples.py)

```
# Разные способы создания кортежей:
Alpha=(5,10,15,"двадцать")
Bravo=100,["один","два","три"],200
Charlie=tuple([1,2,3,(4,5,6,7,8,9)])
Delta=tuple("ABCDEF")
Echo=tuple(2**k for k in range(11))
# Считывание значений элементов и получение среза:
print("Alpha:",Alpha)
print("Элементов:",len(Alpha))
print("Первый:",Alpha[0])
print("Последний:",Alpha[-1])
print("Bravo:",Bravo)
print("Элементов:",len(Bravo))
print("Bravo[1]:",Bravo[1])
print("Bravo[1][2]:",Bravo[1][2])
print("Charlie:",Charlie)
print("Элементов:",len(Charlie))
print("Charlie[3]:",Charlie[3])
print("Charlie[3][1:4]:",Charlie[3][1:4])
print("Delta:",Delta)
print("Элементов:",len(Delta))
print("Delta[-3:]:",Delta[-3:])
print("Echo:",Echo)
Foxtrot=tuple(Echo[k] for k in range(len(Echo)) if k%2==0)
print("Foxtrot:",Foxtrot)
Golf=Echo[2:5]
print("Golf:",Golf)
```

Создание кортежа

Результат выполнения:

```
Alpha: (5, 10, 15, 'двадцать')
Элементов: 4
Первый: 5
Последний: двадцать
Bravo: (100, ['один', 'два', 'три'], 200)
Элементов: 3
Bravo[1]: ['один', 'два', 'три']
Bravo[1][2]: три
Charlie: (1, 2, 3, (4, 5, 6, 7, 8, 9))
Элементов: 4
Charlie[3]: (4, 5, 6, 7, 8, 9)
Charlie[3][1:4]: (5, 6, 7)
Delta: ('A', 'B', 'C', 'D', 'E', 'F')
Элементов: 6
Delta[-3:]: ('D', 'E', 'F')
Echo: (1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024)
Foxtrot: (1, 4, 16, 64, 256, 1024)
Golf: (4, 8, 16)
```

Операции со списками и кортежами

Основные операции:

- считывание значения элемента (или группы элементов);
- изменение значения элемента;
- вставка элемента (или группы элементов) в последовательность;
- удаление элемента (или группы элементов) из последовательности;
- изменение порядка элементов в последовательности.

Операции со списками

Программа (UsingLists_2.py)

```
A: [10, 20, 30]
B: ['Python', [1, 2]]
C: [10, 20, 30, 'Python', [1, 2]]
C: [10, 20, 30, 'Python', [1, 2], 100]
C: [10, 30, 'Python', [1, 2], 100]
C: [200, 10, 30, 'Python', [1, 2], 100]
C: ['A', 'B', 'Python', [1, 2], 100]
C: ['A', 'B', 8, 9, 'Python', [1, 2], 100]
C: ['A', 'B', 7, 9, 'Python', [1, 2], 100]
```

```
# Создание списков:
A=[10,20,30]
print("A:",A)
B=["Python",[1,2]]
print("B:",B)
# Вычисление суммы списков:
C=A+B
print("C:",C)
# Добавление элемента в конец списка:
C+=[100]
print("C:",C)
# Удаление элемента списка:
C[1:2]=[]
print("C:",C)
# Добавление элемента в начало списка:
C=[200]+C
print("C:",C)
# Замена нескольких элементов в списке:
C[:3]=["A","B"]
print("C:",C)
# Вставка элементов в список:
C[2:2]=[8,9]
print("C:",C)
# Присваивание значения элементу списка:
C[2:3]=[7]
print("C:",C)
```

Операции с кортежами

Программа (UsingTuples_2.py)

```
A: (10, 20, 30)
B: ('Python', (1, 2))
C: (10, 20, 30, 'Python', (1, 2))
C: (10, 20, 30, 'Python', (1, 2), 100)
C: (10, 30, 'Python', (1, 2), 100)
C: (200, 10, 30, 'Python', (1, 2), 100)
C: ('A', 'B', 'Python', (1, 2), 100)
C: ('A', 'B', 8, 9, 'Python', (1, 2), 100)
C: ('A', 'B', 7, 9, 'Python', (1, 2), 100)
```

```
# Создание кортежей:
A=(10,20,30)
print("A:",A)
B=("Python",(1,2))
print("B:",B)
# Вычисление суммы кортежей:
C=A+B
print("C:",C)
# "Добавление" элемента в конец кортежа:
C+=(100,)
print("C:",C)
# "Удаление" элемента кортежа:
C=C[:1]+C[2:]
print("C:",C)
# "Добавление" элемента в начало:
C=(200,)+C
print("C:",C)
# "Замена" нескольких элементов:
C=("A","B")+C[3:]
print("C:",C)
# "Вставка" элементов в кортеж:
C=C[:2]+(8,9)+C[2:]
print("C:",C)
# "Присваивание" значения элементу:
C=C[:2]+(7,)+C[3:]
print("C:",C)
```

Создание выборки

Генераторы последовательностей:

выражение for переменная in диапазон if усл

знач if усл else знач for перем in диапазон

знач if усл else знач for перем in диапазон if усл

Также можно использовать срезы

При умножении списка/кортежа на число создается новый список/кортеж кратным повторением содержимого исходного списка/кортежа

Создание выборки

Программа (ListsAndTuples.py)

```
# Кортеж чисел:
A=tuple(k for k in range(1,21) if k%3!=0)
print(A)
# Список чисел:
B=[2**(k//2) if k%2==0 else 3**(k//2) for k in range(15)]
print(B)
# Список чисел:
C=[0 if k==0 or k==1 else k**2 for k in range(13) if not k in [2,5,7]]
print(C)
# Кортеж в обратном порядке:
Alpha=A[::-1]
print(Alpha)
# Элементы выбираются "через один", начиная с первого:
Bravo=B[::2]
print(Bravo)
# Элементы выбираются "через один", начиная со второго:
Charlie=B[1::2]
print(Charlie)
```

```
(1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19, 20)
[1, 1, 2, 3, 4, 9, 8, 27, 16, 81, 32, 243, 64, 729, 128]
[0, 0, 9, 16, 36, 64, 81, 100, 121, 144]
(20, 19, 17, 16, 14, 13, 11, 10, 8, 7, 5, 4, 2, 1)
[1, 2, 4, 8, 16, 32, 64, 128]
[1, 3, 9, 27, 81, 243, 729]
```

Умножение на число

```
Alpha=5*[0]
print(Alpha)
Bravo=(1,)*3
print(Bravo)
Charlie=[1,2]*3
print(Charlie)
Delta=[[1,2]]*3
print(Delta)
Echo=4*(1,[2,3])
print(Echo)
Foxtrot=([1]*2)*3
print(Foxtrot)
Golf=([1]*2,)*3
print(Golf)
```

Программа (ListsAndTuples_2.py)

```
[0, 0, 0, 0, 0]
(1, 1, 1)
[1, 2, 1, 2, 1, 2]
[[1, 2], [1, 2], [1, 2]]
(1, [2, 3], 1, [2, 3], 1, [2, 3], 1, [2, 3])
[1, 1, 1, 1, 1, 1]
([1, 1], [1, 1], [1, 1])
```

Вложенные списки

Программа (ListsAndTuples_3.py)

```
# Импорт функций для генерирования случайных чисел:
from random import seed
from random import randint
# Создание вложенного списка:
A=[[(j+1)*10+i+1 for i in range(2)] for j in range(3)]
print("Список A:",A)
# Инициализация генератора случайных чисел:
seed(2019)
# Список случайных чисел:
B=[[randint(0,9) for i in range(3)] for j in range(2)]
print("Список B:",B)
# Список с буквами:
val='A'
m=2
n=3
C=[['' for i in range(m)] for j in range(n)]
for i in range(n):
    for j in range(m):
        C[i][j]=val
        val=chr(ord(val)+1)
print("Список C:",C)
# Список определяет количество строк во вложенном списке:
size=[1,3,2,4]
# Создание вложенного списка:
D=[['*' for k in range(s)] for s in size]
print("Список D:")
for s in D:
    print(s)
```

```
Список A: [[11, 12], [21, 22], [31, 32]]
Список B: [[2, 3, 7], [2, 3, 3]]
Список C: [['A', 'B'], ['C', 'D'], ['E', 'F']]
Список D:
['*']
['*', '*', '*']
['*', '*']
['*', '*', '*', '*']
```

Копирование списков

```
# Исходный список:
A=[1,3,5]
# Присваивание списков:
B=A
# Изменение значений элементов:
B[1]="Python"
A[2]=(10,20)
# Проверка результата:
print(A)
print(B)
```

Программа (ListsAndTuples_4.py)

```
[1, 'Python', (10, 20)]
[1, 'Python', (10, 20)]
```

```
# Исходный список:
A=[1,3,[10,20],"Python",[40,50]]
# Создание поверхностной копии списка:
B=A[:]
C=A.copy()
print("Исходные значения:")
print("A:",A)
print("B:",B)
print("C:",C)
# Внесение изменений в исходный список:
A[0]=[100,200]
A[2][1]=300
A[3]="Java"
A[4]=90
C[4][1]="C++"
print("После внесения изменений:")
print("A:",A)
print("B:",B)
print("C:",C)
```

Программа (ListsAndTuples_5.py)

```
Исходные значения:
A: [1, 3, [10, 20], 'Python', [40, 50]]
B: [1, 3, [10, 20], 'Python', [40, 50]]
C: [1, 3, [10, 20], 'Python', [40, 50]]
После внесения изменений:
A: [[100, 200], 3, [10, 300], 'Java', 90]
B: [1, 3, [10, 300], 'Python', [40, 'C++']]
C: [1, 3, [10, 300], 'Python', [40, 'C++']]
```

Полная копия

```
# Импорт функции для создания полной копии:
from copy import deepcopy
# Исходный список:
A=[1,3,[10,20],"Python",[40,50]]
# Создание полной копии списка:
B=deepcopy(A)
print("Исходные значения:")
print("A:",A)
print("B:",B)
# Внесение изменений в исходный список:
A[0]=[100,200]
A[2][1]=300
A[3]="Java"
A[4]=90
print("После внесения изменений:")
print("A:",A)
print("B:",B)
```

Программа (ListsAndTuples_6.py)

```
Исходные значения:
A: [1, 3, [10, 20], 'Python', [40, 50]]
B: [1, 3, [10, 20], 'Python', [40, 50]]
После внесения изменений:
A: [[100, 200], 3, [10, 300], 'Java', 90]
B: [1, 3, [10, 20], 'Python', [40, 50]]
```

Функции и методы

- Метод **insert()** - вставки в список нового элемента
- Добавление элемента в конец списка - метод **append()**
- Метод **extend()** добавляет в конец списка элементы списка, указанного аргументом метода
- Метод **pop()** позволяет удалить элемент из списка. Результат метода — значение удаляемого элемента
- Для удаления элемента используют метод **remove()**
- Чтобы изменить (на обратный) порядок элементов используют метод **reverse()** или функцию **reversed()**
- Метод **index()** позволяет определить индекс элемента в списке
- Метод **count()** - количество элементов с данным значением
- Сортировка - метод **sort()** (опция **reverse** со значением **True** или **False**) или функция **sorted()** (опция **reversed** со значением **True** или **False**)

Вставка/удаление

```
# Размер списка:
n=10
# Начальное значение для списка:
A=[1,1]
# Заполнение списка:
for k in range(n-2):
    A.append(A[-1]+A[-2])
# Проверка содержимого списка:
print("A:",A)
# Изменение порядка элементов:
for k in range(len(A)-1):
    A.append(A.pop(-k-2))
# Проверка содержимого списка:
print("A:",A)
# Удаление наибольшего элемента:
A.remove(max(A))
# Проверка содержимого списка:
print("A:",A)
# Удаление наименьшего элемента:
A.remove(min(A))
# Проверка содержимого списка:
print("A:",A)
# Добавление элемента в начало:
A.insert(0,A[0]+A[1])
```

```
# Проверка содержимого списка:
print("A:",A)
# Пустой список:
B=[]
# Элементы списка переносится в другой:
for k in range(len(A)//2):
    B.insert(0,A.pop(-1))
# Проверка содержимого списков:
print("A:",A)
print("B:",B)
# Добавление элемента в конец списка:
A.append(B)
# Проверка содержимого списка:
print("A:",A)
# Удаление последнего элемента в списке:
A.pop(-1)
# Проверка содержимого списка:
print("A:",A)
# Добавление элементов в список:
A.extend(B)
# Проверка содержимого списка:
print("A:",A)
```

Программа (ListsAndTuples_7.py)

Вставка/удаление

Программа (ListsAndTuples_7.py) - результат

```
A: [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
A: [55, 34, 21, 13, 8, 5, 3, 2, 1, 1]
A: [34, 21, 13, 8, 5, 3, 2, 1, 1]
A: [34, 21, 13, 8, 5, 3, 2, 1]
A: [55, 34, 21, 13, 8, 5, 3, 2, 1]
A: [55, 34, 21, 13, 8]
B: [5, 3, 2, 1]
A: [55, 34, 21, 13, 8, [5, 3, 2, 1]]
A: [55, 34, 21, 13, 8]
A: [55, 34, 21, 13, 8, 5, 3, 2, 1]
```

Поиск/подсчет

Программа (ListsAndTuples_8.py)

```
# Подключение функций для генерирования случайных чисел:
from random import *
# Инициализация генератора случайных чисел:
seed(2019)
# Создание списка из случайных чисел:
A=[randint(10,20) for k in range(15)]
# Содержимое списка:
print("A:",A)
# Подсчет элементов с разными значениями:
for a in range(min(A),max(A)+1):
    print(a,"-",A.count(a))
# Наибольшее, наименьшее и среднее значения:
print("Наименьший:")
print("A[",A.index(min(A)),"]=",min(A),sep="")
print("Наибольший:")
print("A[",A.index(max(A)),"]=",max(A),sep="")
print("Среднее:",sum(A)/len(A))
# Сортировка списка:
B=sorted(A)
A.sort(reverse=True)
# Проверка содержимого списков:
print("A:",A)
print("B:",B)
```

Поиск/подсчет

```
A: [12, 13, 17, 12, 13, 20, 13, 20, 15, 20, 19, 20, 14, 16,
19]
12 - 2
13 - 3
14 - 1
15 - 1
16 - 1
17 - 1
18 - 0
19 - 2
20 - 4
Наименьший:
A[0]=12
Наибольший:
A[5]=20
Среднее: 16.2
A: [20, 20, 20, 20, 19, 19, 17, 16, 15, 14, 13, 13, 13, 12, 12]
B: [12, 12, 13, 13, 13, 14, 15, 16, 17, 19, 19, 20, 20, 20, 20]
```

Программа (ListsAndTuples_8.py) - результат

Задание - 3

Напишите программу, в которой создается числовой список. Список заполняется случайными числами. Затем элементы с четными индексами сортируются в порядке возрастания, а элементы с нечетными индексами сортируются в порядке убывания.

Задание - 4

Напишите программу, в которой создается числовой список. Список заполняется случайными числами. Затем между каждой парой элементов этого списка вставляется новый элемент, равный сумме значений соседних элементов.

Домашнее задание

[1] Напишите программу, в которой создается вложенный список из случайных чисел. В матрице, которая реализуется данным вложенным списком, удаляется строка и столбец. Номер строки и номер столбца, которые нужно удалить, вводятся пользователем.

[2] Напишите программу для сортировки списка методом пузырька: последовательно сравниваются значения соседних элементов, и если значение элемента слева больше значения элемента справа, они меняются местами. За один полный перебор элементов в списке элемент с самым большим значением оказывается последим в списке. За второй перебор предпоследним оказывается элемент со вторым по величине значением, и так далее.