Лабораторная работа №6

Списки.

Операции соединения и повторения

Списки – это изменяемые последовательности в отличие от строк.

Представим строку как объект в памяти, в этом случае, когда над строкой выполняются операции конкатенации и повторения, то сама строка не меняется, но в результате выполнения операции в другом месте памяти создается другая строка.

В строку нельзя добавить новый символ или удалить существующий, не создав при этом новой строки.

Пример:

Так, например, в Питоне нельзя переприсваивать значение для отдельных символов строки.

Программа выдаст ошибку!

```
s="aaa";
s[1]="b";
print(s1)
```

Изменять строку можно только, работая с ней, **как с объектом** (метод replace, например):

```
s1="breKeKeKeKs";
s1=s1.replace('Ke','XoXo',2)
s1 # breXoXoXoXoKeKs
```

Что касается списков, то при выполнении операций другие списки могут не создаваться, при этом *изменяется непосредственно оригинал*. Из списков можно **удалять и добавлять новые элементы**.

Операция конкатенации:

```
[33, -12, 'may'] + [21, 48.5, 33] # [33, -12, 'may', 21, 48.5, 33]
```

или так:

```
a=[33, -12, 'may']
b=[21, 48.5, 33]
```

```
print(a+b)# [33, -12, 'may', 21, 48.5, 33]
```

Операция повторения:

```
[[0,0],[0,1],[1,1]] * 2 # [[0, 0], [0, 1], [1, 1], [0, 0], [0, 1], [1, 1]]
```

Пример:

Для списков операция переприсваивания значения отдельного элемента списка **разрешена!**:

```
a=[3, 2, 1]
a[1]=0;
print(a) # [3, 0, 1]
```

Можно!

Задание 6_1: Ввести строку, в которой записана сумма натуральных чисел, например, ‘1+25+3’. Вычислите это выражение. Работать со строкой, как со списком.

Начало программы:

```
s=input('введите строку')
l=list(str(s));
```

Другие операции над списками при помощи функций

```
a=[1,7,3,88,33]
a.sort() #[1,3,7,33,88]  - сортировка
a.reverse() #[88,33,7,3,1]  - обратная сортировка
a.index(7) #2  - индекс элемента
a.clear() #  - очистка списка
len(a) #  - длина списка
sum(a) #  - суммирование элементов
```

Функция join() – соединение элементов через определенный символ:

```
lst=['11','22','33']
lst="-".join(lst)# '11-22-33'
```

Функция split([sep]) – возвращает список подстрок, получающихся разбиением строки a разделителем sep:

```
str="1-2-3-4"
s1=str.split("-") # ['1','2','3','4']
```

Задание 6_2:

Дан список из 5 различных элементов. Используя функции (не использовать цикл), необходимо найти и вывести:

- минимальный и максимальный элементы списка;
- сумму и среднее арифметическое;
- второй минимальный элемент (второй по минимальности).

Начало программы:

```
lst=[4,5,2,3,4]
```

Добавление и удаление элементов списка

- Добавление элемента:

```
>>> a=[]
>>> a.append('444')
>>> a
['444']
```

Пример:

Поиск нечетных элементов в массиве *mas* и копирование их в массив *B*.

Решение:

```
B = []
for x in mas:
    if x % 2 != 0:
        B.append(x)
```

- Удаление элемента:

```
> a.remove('444')
>>> a
[]
```

- Удаление элемента по индексу:

```
>>> del a[0]
>>> a
[]
```

При копировании списков, т.е. присваивании одного списка другому, изменение первого списка влечет за собой изменение второго списка. Так как эти объекты связаны одной областью памяти (ссылка на список).

```
mas1 =[1, 2, 3]
mas2 = mas1 # создается ссылка на список
mas1[0] = 4
print(mas2) #[4, 2, 3]
```

Чтобы создать не ссылку на список, а копию списка можно использовать либо срез либо функцию *copy*.

1.

```
mas2 = mas1[:] # используем срез
```

2.

```
import copy
mas1 =[1,2,3]
mas2 = copy.copy(mas1)
```

Задание 6_3: Проверить, является ли заданное слово палиндромом.

Примечание:

- Пример палиндрома: *казак*, *ABBA*
- Использовать функции.
- Поскольку при присваивании одного списка другому, изменение первого ведет к аналогичному изменению второго списка, то необходимо использовать копию (copy).

Начало программы:

```
import copy
stroka=input('введите слово')
lst=list (stroka) # конвертируем строку в список
```

Генерация случайных чисел

Встроенный модуль Питона **random** позволяет генерировать псевдослучайные числа.

Модуль **random** включает в себя функцию *random*, которая возвращает действительное число в диапазоне от *0.0* до *1.0*. Каждый раз при вызове функции возвращается число из длинного ряда.

Пример:

```
import random
for i in range(10):
  x = random.random()
  print (x) # 0.5185207383774904  0.78283351055836  0.23601341583293534  ...
```

Чтобы получить случайное число между *0.0* и верхней границей *high*, просто умножьте *x* на high.

Например, от 0.0 до 15.0:

```
import random
for i in range(10):
  x = random.random()
  print (x * 15) # 11.075319687990554  7.152253113207329 ...
```

Для того, чтобы получить псевдослучайное целое число:

```
import random
random.randint(<начало>,<конец>)
```

Для того, чтобы получить псевдослучайное вещественное число:

```
import random
random.uniform(<начало>,<конец>)
```

Еще пример:

```
from random import randint
l = [randint(10,80) for x in range(10)]
```

Задание Python 6_4: Найдите в массиве все простые числа и скопируйте их в новый массив.

Задание Python 6_5: Решить задачу поиска среднего значения в списке из N элементов. Использовать метод добавления элементов списка и суммирования элементов

Цикл for при работе со списками

```
mylist=[1,2,3,4,5]
for item in mylist:
  item = 0
# mylist не меняется!
print(mylist) # [1, 2, 3, 4, 5]
n=5
for i in range(n):
  mylist[i] = 0
# mylist меняется
print(mylist) # [0, 0, 0, 0, 0]
```

В списке чисел проверить, все ли элементы являются уникальными, т.е. каждое число встречается только один раз

Решение:

Комментарии к программе: Решать данную задачу на языке Python мы будем «классическим» вариантом – брать по очереди элементы списка и сравнивать каждый элемент со стоящими за ним. При первом же совпадении элементов делается вывод, что в списке есть одинаковые элементы, и работа программы завершается.

Для выхода из цикла будем использовать метод quit()

```
import random
m = 6
mass=[]
k=0
j=0
for i in range(m):
  mass.append(random.randint(-10,10))
  print(mass[i]) # -10 0 -8 0 -10 1
for i in mass:
  k=k+1
```

```
    for j in range(k,m): # j = 0 -8  0 -10 -> quit
            if (i==mass[j]): # -10==0  -10==-8  -10==0  -10==-10 -> quit
                print('yes')
                quit()
```

Задание Python 6_6: Определить индексы элементов массива (списка), значения которых принадлежат заданному диапазону (т.е. не меньше заданного минимума и не больше заданного максимума)

```
 -5  9  0  3 -1 -2  1  4 -2 10  2  0 -9  8 10 -9
min= 5
max= 15
кол-во:  5
индексы:  [1, 9, 13, 14, 19]
```

* в результате получили индексы элементов, значения которых находятся в диапазоне [5,15]

Алгоритм:

1. Заполнить список (массив) случайными числами
2. Запросить для ввода минимум и максимум диапазона
3. Найти индексы элементов, значения которых входят в диапазон. Добавлять найденные индексы к новому списку
4. Вывести общее число найденных индексов (функция len()) и отдельно все индексы

Задание Python 6_7: Дополнить предыдущую программу следующим:

После того, как элемент с подходящим значением добавлен в новый список – удалять его из исходного списка