

Основы и методология программирования

Семинар 8

Списки: поиск и сортировка

Списки (повторение)

Элементы списка не обязательно должны быть одного типа.

```
lst = ['spam', 'drums', 100, 1234]
```

Для списка можно получить срез, объединить несколько списков и т.д.

```
lst[1:3] # ['drums', 100]
```

Можно менять как отдельные элементы списка, так и диапазон:

```
lst[3] = 'piano'
```

```
lst[0:2] = [1, 2]
```

```
print(lst) # [1, 2, 100, 'piano']
```

Вставка:

```
lst[1:1] = ['guitar', 'microphone']
```

```
print(lst) # [1, 'guitar', 'microphone', 2, 100, 'piano']
```

Можно сделать выборку из списка с определенной частотой:

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
print(numbers[::4]) # [1, 5, 9]
```

Задача 1

В списке случайных чисел найти самую длинную последовательность, которая упорядочена по возрастанию.
Если таких последовательностей с одинаковой максимальной длиной несколько, то найти их все.
Вывести на экран сам список, длину самой длинной упорядоченной по возрастанию последовательности и эту последовательность (и).

```
from random import random
```

```
# Задаем список случайных чисел  
n = 20  
lst = [0] * n  
for i in range(n):  
    lst[i] = int(random() * 50)  
print(lst)
```

Задача 1

Находим максимальную последовательность неубывающих элементов

```
lenMax = 1 = 1
```

```
for i in range(1, n):
```

```
    if lst[i] > lst[i - 1]:
```

```
        l += 1
```

```
    else:
```

```
        if l > lenMax:
```

```
            lenMax = l
```

```
        l = 1
```

```
print("Максимальная неубывающая последовательность:", lenMax)
```

Проверяем, сколько таких последовательностей, если найдено, печатаем срез списка

```
l = 1
```

```
for i in range(1, n):
```

```
    if lst[i] > lst[i - 1]:
```

```
        l += 1
```

```
    else:
```

```
        if l == lenMax:
```

```
            print(lst[i - l:i])
```

```
        l = 1
```

Задача 2

Требуется отсортировать список случайных чисел от 10 до 100 любым известным способом.

```
myLst1 = []  
# Заполнение  
for i in range(100):  
    myLst1.append(10 + int(random() * 91))  
# Проверка  
for j in myLst1: # для каждого элемента списка  
    if (j < 10 or j > 100):  
        print("Error in List: ", j)  
        break  
print("List was created")  
# Сортировка (самостоятельно!)  
# ...  
  
# Вывод отсортированного списка в виде таблицы 10*10  
row = 0  
for i in myLst1:  
    print("%3d" % i, end = ' ')  
    row += 1  
    if not(row % 10):  
        print()
```

Задача 3

Список заполнен случайным образом нулями и единицами.

Найти самую длинную непрерывную последовательность единиц и определить индексы первого и последнего элементов в ней.

```
import random
```

```
list = []
```

```
for i in range(50):
```

```
    n = int(random.random() * 2)
```

```
    list.append(n)
```

```
    print(n, end='')
```

```
    if not ((i + 1) % 10):
```

```
        print()
```

Задача 3

```
count = 0
maxCount = 0
index = 0
i = 0

while i < len(list):
    if list[i] == 1:
        count += 1
    else:
        if count > maxCount:
            maxCount = count
            index = i - 1 # последовательность закончилась на предыдущем
элементе
        count = 0
    i += 1

print("\nКоличество элементов: ", maxCount)
print("id первого элемента: ", index - maxCount + 1)
print("id последнего элемента: ", index)
```

Задача 4

Определить количество уникальных элементов списка, содержащего 20 случайным образом заданных значений от 10 до 15 с шагом 0.5.

```
import random
```

```
myLst = [0] * 20 # пустой список из 20
```

```
for i in range(len(myLst)): # как записать короче?
```

```
    myLst[i] = (20 + round(random.random() * 11)) / 2
```

```
print(myLst)
```

```
countUnique = 0
```

```
for i in range(len(myLst)):
```

```
    if not myLst[i] in myLst[i + 1:]:
```

```
        countUnique += 1
```

```
print("Количество уникальных элементов:", countUnique)
```

```
print("Второй способ: ", len(set(myLst)))
```


Сравнение строк, кортежей и списков

```
# строки сравниваются лексикографически (как в словаре)
# для правильного сравнения строк надо их приводить к одному регистру!
str1 = "Мама мыла раму"
str2 = "мама мыла раму"
print(str1 > str2) #true? false?
print(str1 == str2) #true? false?
print(str1.lower() == str2.lower()) #true? false?

myTuple1 = (2, 6, 3)
myTuple2 = (3, 1, 3)
print(myTuple1 < myTuple2) myTuple3 = (2, )
print(myTuple1 < myTuple3)
```

Сравнивать можно только однотипные элементы! Если нет уверенности, что данные будут однотипные - надо привести их к одному типу, например, воспользовавшись функцией `map`.

```
myTuple1 = (2, 6, 3)
myTuple2 = ("3", 1, 3)
# print(myTuple1 < myTuple2) #ошибка
print(tuple(map(str, myTuple1)) < tuple(map(str, myTuple2))) # true
```

sort и sorted

```
# sorted - встроенная функция,  
# не изменяет объекта,  
# возвращая list - его копию  
# можно применять к любым перечислимым  
типам
```

```
myTuple = (26, 3, 5)  
myTuple2 = sorted(myTuple)  
print(type(myTuple2), myTuple2)  
print(type(myTuple), myTuple)  
myStr = "Мой дядя самых честных правил"  
myStr2 = sorted(myStr)  
print(type(myStr2), sorted(myStr), myStr)  
# однако  
myList3 = [26, 3, 5, "в", "aa"]  
# ошибка  
# sorted(myList3)  
myList1 = myStr.split()  
print(sorted(myList1, reverse=True))
```

```
# .sort() - метод у  
объекта, изменяет его
```

```
myTuple = (26, 3, 5)  
# нельзя кортеж -  
неизменяемый объект  
# myTuple.sort()  
myList = list(myTuple)  
myList.sort()  
print(myList)
```

Параметры при сортировке

`list.sort([key], [reversed])`
`sorted([key], [reversed])`

Имеет два необязательных параметра:

key — функция, которая принимает элемент списка и возвращает объект, который будет использоваться при сравнении во время сортировки вместо оригинального элемента списка;

reversed — если `True`, то список будет отсортирован в обратном порядке.

```
print(sorted("This is a test string in Python".split(), key=str.lower))  
# ['a', 'in', 'is', 'Python', 'string', 'test', 'This']  
print(sorted("This is a test string in Python".split()))  
# ['Python', 'This', 'a', 'in', 'is', 'string', 'test']
```

#Что будет выведено в результате выполнения сортировки?

```
mylist = [3, 6, 3, 2, 4, 8, 9]  
print(sorted(mylist, key=lambda x: max(mylist) + x if not x % 2 else x))
```

Задача 5

Задан список спортсменов, принимавших участие в забеге: фамилия, имя, результат. Вывести таблицу итоговых результатов.

При равных результатах сортировать спортсменов по фамилии, однофамильцев по имени.

```
def sportKey(sport):
```

```
    return(sport[2], sport[0], sport[1])
```

```
sportList = [['Иванов', 'Иван', 10], ['Иванов', 'Петр', 10], ['Петров', 'Петр', 10.3], ['Петров', 'Иван', 10.3],  
['Акимов', 'Иван', 10]]
```

```
sportList.sort(key=sportKey)
```

```
for sportsman in sportList:
```

```
    resName = ''.join(sportsman[0:2])
```

```
    res = ': '.join(map(str, [resName, sportsman[2]]))
```

```
    print(res)
```

Задача 6

```
# Прочитать из файла список скачиваний файлов за неделю (Формат: <имя файла> <номер недели>  
<количество>.
```

```
# Вывести в output.txt все файлы в алфавитном порядке без повторов с суммарным количеством  
скачиваний.
```

```
# Рассчитать самую загруженную неделю
```

```
names = [] # имена файлов
```

```
downloads = [] # количество скачиваний
```

```
weeks = [] # номера недель
```

```
downldsByWeeks = [] # количество скачиваний
```

```
file = open('input.txt', 'r', encoding='utf-8')
```

```
# Обработка файла. См. след. слайд
```

```
file.close()
```

Задача 6

```
for line in file.readlines():
    if len(line.strip()): #если не пустая строка
        values = line.split()
        if values[0] in names:
            downloads[names.index(values[0])] += int(values[2])
        else:
            names.append(values[0])
            downloads.append(int(values[2]))

    if values[1] in weeks:
        downldsByWeeks[weeks.index(values[1])] += int(values[2])
    else:
        weeks.append(values[1])
        downldsByWeeks.append(int(values[2]))
```

Задача 6

```
# Вывод
```

```
print ("Самая нагруженная неделя: %s, количество скачиваний %d" %  
      (weeks[downldsByWeeks.index(max(downldsByWeeks))], max(downldsByWeeks)))  
namesSorted = sorted(names)
```

```
file2 = open('output.txt', 'w', encoding='utf-8')  
for name in namesSorted:  
    file2.write("%s %d\n" % (name, downloads[names.index(name)]))  
file2.close()
```