АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

и.о. доцент кафедры «Информационные системы»
Муханова Аягоз Асанбековна

Лекция №6 **Типовые задачи обработки списков**

План лекции:

Поиск и отбор нужных
элементов
Запись всех элементов списка с заданными свойствами в другой список
Поиск индекса первого элемента списка с заданными свойствами
Определение индекса максимального элемента списка
Определение количества максимальных/минимальных элементов списка
Основные операции со списками

Поиск и отбор нужных элементов

Вывод на экран элементов с заданными свойствами

Ранее в лекции приводились примеры вывода на экран всех или группы элементов списка. В данной задаче на экран должны быть выведены только элементы с заданными свойствами (например, положительные, четные и т. п.). Если условие отбора определяется значением элемента списка a[i] или/и его индексом i, то задача решается следующим образом:

```
for i in range(n):
    if <условие>:
        print(a[i])
```

Если же оно зависит только от значения элемента, то можно оформить фрагмент так:

```
for el in a:
    if <условие>:
        print(el)
```

```
>>> a = [1,2,3,4,5]
>>> for i in range(4):
        if a[i]%2==0:
            print (a[i])


2
4
>>>
```

Запись всех элементов списка с заданными свойствами в другой список

Алгоритм решения задачи: сначала создаем пустой список, затем
перебираем все элементы исходного списка и, если очередной элемент нам нужен, добавляем его в новый список с помощью метода
append():

```
b = [] #Новый список
for i in range(n):
    if <условие>:
        b.append(a[i])
```

```
>>> a = [1,2,3,4,5]
>>> b=[]
>>> for i in range(5):
                if a[i]%2==0:

                b.append(a[i])

                print (b)
[2, 4]
>>>
```

Или

```
b = []
for el in a:
    if <условие>:
        b.append(el)
```

Можно также использовать генератор списка с условием:

```
b = [a[i] for i in range(n) if <условие>]
```
или
```
b = [el for el in a if <условие>]
```

В двух последних фрагментах перебираются все элементы списка a, и те из них, которые удовлетворяют заданному условию, включаются в новый список b.

Поиск индекса первого элемента списка с заданными свойствами

Используем в программе величину vperv типа bool, принимающую значение True, если элемент с заданными свойствами встретился в списке впервые, и False – в противном случае:

```
isk_index = -1 #Условное значение
#Переменная isk_index – искомый индекс
vperv = True #Если встретится нужный элемент – он будет первым
for i in range(n):
    if <условие>: #Встретится элемент с заданными свойствами
        #Проверяем, является ли он первым
        #Если является
         if vperv == True: #Или if vperv:
            #Запоминаем его индекс
             isk_index = i
              #Другие элементы с заданными свойствами
              #уже будут не первыми
               vperv = False
#Вывод результата
if isk_index > -1:
        print('Индекс этого элемента:', isk_index)
else:
        print('Таких значений в списке нет')
```

Можно прекратить обработку списка после нахождения (возможного) первого нужного элемента, используя инструкцию break:

```
isk_index = -1
vperv = True
for i in range(n):
    if <условие>:
        if vperv == True: #Или if vperv:
        isk_index = i
        #Прекращаем проверку
        break
#Вывод результата
...
```

- Если стоит задача поиска индекса первого элемента списка, имеющего заданное значение N, то удобно использовать стандартную функцию index(), которая возвращает указанный индекс:

```
ind = a.index(N) #Функция вызывается как метод
```

- Однако надо учитывать, что если нужного элемента в списке нет, при выполнении программы эта строчка вызовет ошибку.
- Заметим также, что функция index() возвращает индекс элемента, равного заданному значению, а описанный до этого метод – универсальный и применим к нахождению индекса первого элемента списка с любыми заданными свойствами (четного, положительного и т. п.)

Определение индекса максимального элемента списка

- Определение индекса максимального элемента списка

1) определить максимальный элемент M:

M = max(a)

2) найти его индекс ind_max с помощью функции index()

ind_max = a.index(M)

При решении задачи методом, аналогичным описанному ранее, список просматривается только один раз:

```
#Начальное присваивание значений искомым величинам
M = a[0] #Максимальное значение
ind_max = 0 #Его индекс
```

- Можно также использовать не индексы, а значения элементов: M = a[0] for el in a: if el > M: M = el
- В таком варианте элемент a[0] рассматривается дважды (второй раз – в цикле, где выполняется полный перебор всех элементов).
- Для определения минимального элемента списка может быть использована функция min().

```
for i in range(1, n): #1 – номер второго элемента списка
                      #в памяти компьютера
                    if a[i] > M:
                        M = a[i]
                        ind_max = i ...
```

- Если значение M находить не требуется, то без величины M можно вообще обойтись. В самом деле, если нам известно значение индекса максимального среди рассмотренных элементов, то мы знаем и значе ние соответствующего элемента (оно равно a[ind_max]):

```
ind_max = 0
for i in range(1, n):
    if a[i] > a[ind_max]:
        ind_max = i

#Максимальный элемент списка равен a[ind_max] ...
```

Определение количества максимальных/минимальных элементов списка

- Задача может быть решена двумя способами:

1) за два прохода по списку;

2) за один проход по списку.

В первом случае на первом проходе следует найти максимальный (минимальный) элемент списка, а на втором – подсчитать количество элементов, равных максимальному (минимальному) значению, эту задачу мы рассматривали ранее

- Методика решения задачи за один проход по списку

```
#Начальное присваивание значений искомым величинам
M = a[0] #Максимальное значение
kol_max = 1 #Количество максимальных значений #Рассматриваем остальные элементы списка
for i in range (1, n):
    if a [i] > M:
        #Встретилось новое максимальное значение
        #Принимаем его в качестве значения M
        M = a[i]
        #Пока он – единственный
        kol_max = 1
    else:
        #Проверяем, не равен ли очередной элемент списка
        #'старому' максимальному значению
        if a[i] == M:
            #Встретилось еще одно максимальное значение
            #Учитываем это
            kol_max = kol_max + 1
```

Возможен также характерный только для языка Python вариант решения (очень краткий). Найдите его

Перестановки элементов

- Обмен местами двух элементов списка

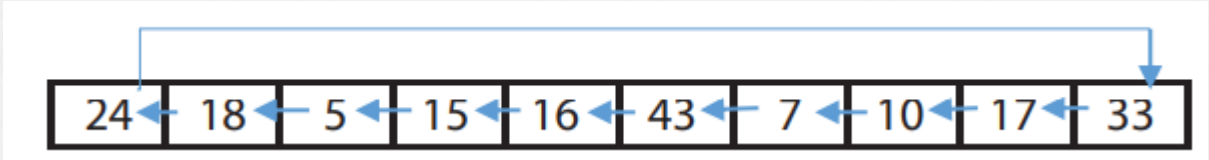
Пример задачи: «Поменять местами 2-й и 5-й элементы списка. Принять, что указаны “реальные” номера элементов». Задача решается аналогично задаче обмена значениями двух «простых» переменных.

```
vsp = a[1] #Запоминаем значение a[1]
a[1] = a[4] #Элементу a[1] присваиваем значение другого
            #элемента
a[4] = vsp #Элементу a[4] присваиваем
            #'старое' значение элемента a[1]
```

Обратите внимание на индексы элементов в программе. Можно также применить вариант обмена, основанный на множественном присваивании

Циклический сдвиг элементов списка влево

- При циклическом сдвиге элементов влево начальный элемент записывается на последнее место в списке, а остальные элементы сдвигаются влево на 1 позицию:



Как решить задачу? Если сдвинуть все элементы, кроме начального, влево, то мы «потеряем» начальное значение (24). Если же сначала записать его на последнее место в списке, то «потеряется» последний элемент (33). Как же быть? Ответ такой – надо запомнить значение начального элемента в какой-то вспомогательной переменной :

```
vsp = a[0]
```

Теперь сдвиг влево провести можем:

```
for i in range(n - 1):
    a[i] = a[i + 1]
```

Осталось записать на последнее место в списке исходное значение начального элемента:

```
a[n – 1] = vsp
```

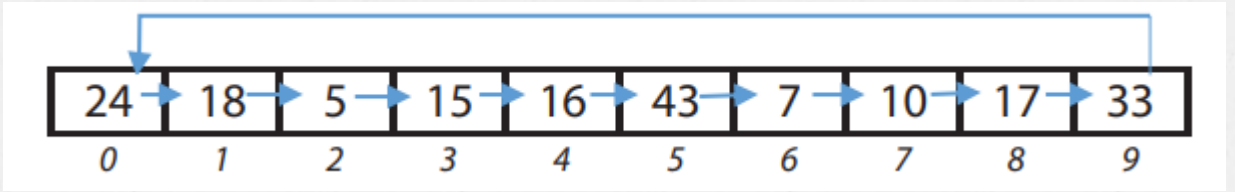
Примечание. Циклический сдвиг влево можно выполнить и с использованием среза:

```
a = a[1 : len(a)] + [a[0]]
```

Обратите внимание на двойные квадратные скобки (к списку можно «прибавить» только список).

Циклический сдвиг элементов списка вправо

- Схема такого сдвига следующая:



Программа решения задачи:

```
#Запоминаем значение последнего элемента
vsp = a[n - 1]
#Сдвигаем элементы вправо
for i in range(n - 1, 0, -1):
    a[i] = a[i – 1]
#Записываем 'старое' значение последнего элемента в начало списка
a[0] = vsp
```

Основные операции со списками

Функция или операция	Описание и результат
lst1 + lst2	Объединение списков. Получается новый список, в котором после элементов списка lst1 находятся элементы списка lst2. Пример: lst1 =[1,2,3] lst2 =['raz', 'dva'] lst3 =lst1+lst2 lst3 → [1, 2, 3, 'raz', 'dva']
lst *n (или n*lst)	n-кратное повторение списка lst. Результат — новый список. Пример: lst2 =['raz', 'dva'] lst2 *3 → ['raz','dva', 'raz', 'dva', 'raz', 'dva']
lst [i]	Выбор из lst элемента с номером i, нумерация начинается с 0 (первый элемент имеет номер 0) Если $i < 0$, отсчёт идёт с конца (последний элемент списка имеет номер -1). Пример: lst3 = [1, 2, 3, 'raz', 'dva'] lst3 [2] → 3 lst3 [−2] → 'raz'
lst [i:j:k]	Срез — список, содержащий элементы списка lst с номерами от i до j с шагом k (элемент с номером i входит в итоговый список, а элемент с номером j уже не входит). Если k не указан (использован вариант lst [i:j]), то символы идут подряд (равносильно lst [i:j:1]). Пример: lst3 = [1, 2, 3, 'raz', 'dva'] lst3 [1:4] → [2, 3, 'raz']
min(lst)	Определяется элемент с наименьшим значением в соответствии с алфавитным («словарным») порядком. Пример: lst3 = [1, 2, 3, 'raz', 'dva'] min(lst3) → 1

Функция или операция	Описание и результат
max(lst)	Определяется элемент с наибольшим значением в соответствии с алфавитным («словарным») порядком. Пример: lst3 = [1, 2, 3, 'raz', 'dva'] max(lst3) → 'raz'
lst [i]=x	Замена элемента списка с номером i на значение x. Если x является списком, то на место элемента списка будет вставлен список. При этом новый список не создаётся. Примеры: lst3 =[1, 2, 3, 'raz','dva'] lst3 [2]=' tri ' lst3 → [1, 2, ' tri ', 'raz', 'dva'] lst3 [2]=[7,8] lst3 → [1, 2, [7, 8], 'raz', 'dva']
del lst [i]	Удаление из списка элемента с номером i. Новый список не создаётся. Пример: lst3 =[1, 2, [7, 8], 'raz', 'dva'] **del** lst3 [2] lst3 → [1, 2, 'raz', 'dva']
lst [i:j]=x	Замена среза списка lst на элемент или список x (несколько элементов заменяются на x). Примеры: lst3 =[1, 2, 3, 'raz', 'dva'] lst3 [2:4]=' tri ' lst3 → [1, 2, 't', 'r', 'i', 'dva'] lst3 [2:4]='a' lst3 → [1, 2, 'a', 'i', 'dva'] Обратите внимание, что строка интерпретируется как список!
del lst [i:j]	Удаление элементов, входящих в указанный срез («вырезание среза»). Пример: lst3 =[1, 2, 'a', 'i', 'dva'] **del** lst3 [2:4] lst3 → [1, 2, 'dva']

Основные методы списков

Метод	Описание и результат
lst .append(x)	Добавление элемента x в конец списка lst. x не может быть списком. Создания нового списка не происходит. Пример: lst=['raz','dva','tri',1,2] lst.append(3) lst → ['raz','dva','tri',1,2,3]
lst .extend(t)	Добавление кортежа или списка t в конец списка lst (похоже на объединение списков, но создания нового списка не происходит). Пример: lst1=[1,2,3] lst2=['raz','dva'] lst1.extend(lst2) lst1 → [1,2,3,'raz','dva']
lst .count(x)	Определение количества элементов, равных x, в списке lst. Результат является числом. Пример: lst=[1,2,3,'raz','dva','raz','dva'] lst .count('raz') → 2
lst .index(x)	Определение первой слева позиции элемента x в списке lst. Если такого элемента нет, возникает сообщение об ошибке. Пример: lst=[1,2,3,'raz','dva','raz','dva'] lst .index('dva') → 4

Метод	Описание и результат
lst . remove(x)	Удаление элемента x в списке lst в первой слева позиции. Если такого элемента нет, возникает сообщение об ошибке. Пример: lst=[1,2,3,'raz','dva','raz','dva'] lst.remove('dva') lst → [1,2,3,'raz','raz','dva']
lst . pop(i)	Удаление элемента с номером i из списка lst. При этом выдаётся значение этого элемента («извлечение» элемента из списка). Если номер не указан, удаляется последний элемент. Новый список не создаётся. Примеры: lst=[1,2,3,'raz','raz','dva'] \ lstinline lst.pop(3)| → 'raz'| lst → [1,2,3,'raz','dva'] \ lstinline lst.pop()| → 'dva'| lst → [1,2,3,'raz']
lst . insert (i ,x)	Вставка элемента или списка x в позицию i списка lst. Если i>=0, вставка идёт в начало списка. Если i>len(lst), вставка идёт в конец списка. Новый список не создаётся. Пример: lst= [1,2,3,'raz'] lst.insert(3,'tri') lst → [1,2,3,'tri','raz']
lst . sort ()	Сортировка списка по возрастанию (в алфавитном порядке). Новый список не создаётся. Пример: lst=[1,2,3,'tri','raz'] lst.sort() lst → [1,2,3,'raz','tri']
lst . reverse ()	Замена порядка следования элементов на обратный. Новый список не создаётся. Пример: lst=[1,2,3,'raz','tri'] lst.reverse() lst → ['tri','raz',3,2,1]

Примеры вызовов функций:

```
>>> e=[56.8060, 57.1578, 57.4093, 56.1843, 57.2207]
>>> e
[56.806, 57.1578, 57.4093, 56.1843, 57.2207]
>>> len(e)
5
>>> max(e)
57.4093
>>> min(e)
56.1843
>>> sum(e)
284.7781
>>> sorted(e)
[56.1843, 56.806, 57.1578, 57.2207, 57.4093]
>>> e
[56.806, 57.1578, 57.4093, 56.1843, 57.2207]
>>>
```

оператор in :

```
>>> h=['bonjour',7,'hola',-1.0,'привіт']
>>> if 7 in h:
print ('Значение есть в списке')
Значение есть в списке
>>>
```

операция взятия среза:

```
>>> h=['bonjour',7,'hola',-1.0,'привіт']
>>> h
['bonjour', 7, 'hola', -1.0, 'привіт']
>>> g=h[1:2]
>>> g
[7]
>>>
```

- Вернемся к инструкции del и удалим с помощью среза подсписок:

```
>>> a = [-1, 1, 66.25, 333, 333, 1234.5]
>>> del a[0]
>>> a
[1, 66.25, 333, 333, 1234.5]
>>> del a[2:4] # удаление подсписка
>>> a
[1, 66.25, 1234.5]
>>> del a[:]
>>> a
[]
>>>
```

Примеры популярных методов списка:

- ```
  >>> colors=['red', 'orange', 'green']
  >>> colors.extend(['black','blue']) # расширяет список списком
  >>> colors
  ['red', 'orange', 'green', 'black', 'blue']
  ```
- ```
  >>> colors.append('purple') # добавляет элемент в список
  >>> colors
  ['red', 'orange', 'green', 'black', 'blue', 'purple']
  ```
- ```
  >>> colors.insert(2,'yellow') # добавляет элемент в указанную позицию
  >>> colors
  ['red', 'orange', 'yellow', 'green', 'black', 'blue', 'purple']
  ```
- ```
  >>> colors.remove('black') # удаляет элемент из списка
  >>> colors
  ['red', 'orange', 'yellow', 'green', 'blue', 'purple']
  ```
- ```
  >>> colors.count('red') # считает количество повторений аргумента метода
  1
  >>> colors.index('green') # возвращает позицию в списке аргумента метода
  3
  ```
  ```

Еще несколько полезных методов для списка:

- >>> colors
  ['red', 'orange', 'yellow', 'green', 'blue', 'purple']
  >>> colors.pop() # удаляет и возвращает последний элемент списка
  'purple'
  >>> colors
- ['red', 'orange', 'yellow', 'green', 'blue']
  >>> colors.reverse() # список в обратном порядке
  >>> colors
  ['blue', 'green', 'yellow', 'orange', 'red']
- >>> colors.sort() # сортирует список (вспомните о сравнении строк)
  >>> colors
- ['blue', 'green', 'orange', 'red', 'yellow']
  >>> colors.clear() # очищает список. Метод появился в версии 3.3. Аналог del color[:]
  >>> colors
  []
  >>>

Преобразование типов

- Очень часто появляется потребность в изменении строк, но напрямую мы этого сделать не можем. Тогда нам на помощь приходят списки.
- Преобразуем строку в список, изменим список, затем вернем его в строку:

```
>>> s='Строка для изменения'
>>> list(s) # функция list() пытается преобразовать аргумент в список
['С', 'т', 'р', 'о', 'к', 'а', ' ', 'д', 'л', 'я', ' ', 'и',
'з', 'м', 'е', 'н', 'е', 'н', 'и', 'я']
>>> lst = list(s)
>>> lst[0]='М' # изменяем список, полученный из строки
>>> lst
['М', 'т', 'р', 'о', 'к', 'а', ' ', 'д', 'л', 'я', ' ', 'и',
'з', 'м', 'е', 'н', 'е', 'н', 'и', 'я']
>>> s=''.join(lst) # преобразуем список в строку с помощью строкового метода join()
>>> s
'Мтрока для изменения'
>>>
```

- Отдельно рассмотрим несколько примеров строкового метода join():
  >>> A = ['red', 'green', 'blue']
  >>> ' '.join(A)
  'red green blue'
  >>> ''.join(A)
  'redgreenblue'
  >>> '***'.join(A)
  'red***green***blue'
  >>> Метод join() принимает на вход список, который необходимо преобразовать в строку, а в качестве строкового объекта указывается соединитель элементов списка.

- Аналогично можно преобразовать число к списку (через строку) и затем изменить полученный список:

```
>>> n=73485384753846538465
>>> list(str(n)) # число преобразуем в строку, затем строку в список
['7', '3', '4', '8', '5', '3', '8', '4', '7', '5', '3', '8',
'4', '6', '5', '3', '8', '4', '6', '5']
>>> Если строка содержит разделитель, то ее можно
преобразовать к списку с помощью строкового метода split(),
который по умолчанию в качестве разделителя использует пробел:
>>> s='d a dd dd gg rr tt yy rr ee'.split()
>>> s
['d', 'a', 'dd', 'dd', 'gg', 'rr', 'tt', 'yy', 'rr', 'ee']
>>>
Возьмем другой разделитель:
>>> s='d:a:dd:dd:gg:rr:tt:yy:rr:ee'.split(":")
>>> s
['d', 'a', 'dd', 'dd', 'gg', 'rr', 'tt', 'yy', 'rr', 'ee']
>>>
```

- Мы уже упоминали, что в качестве элементов списка могут быть объекты любого типа, например, списки:
  >>> lst=[['A', 1], ['B',2], ['C',3]]
  >>> lst
  [['A', 1], ['B', 2], ['C', 3]]
  >>> lst[0]
  ['A', 1]
  >>> Подобные структуры используются для хранения матриц. Обращение (изменение) к вложенному списку происходит через указание двух индексов:
  >>> lst[0][1]
  1
  >>>

Практика

- L = [3, 6, 7, 4, -5, 4, 3, -1] 1. Определите сумму элементов списка L. ЕСЛИ сумма превышает значение 2, ТО вывести на экран число элементов списка. 2. Определить разность между минимальным и максимальным элементами списка. ЕСЛИ абсолютное значение разности больше 10, ТО вывести на экран отсортированный по возрастанию список, ИНАЧЕ вывести на экран фразу «Разность меньше 10».
- L = [3, 'hello', 7, 4, 'привет', 4, 3, -1] Определите наличие строки «привет» в списке. ЕСЛИ такая строка в списке присутствует, ТО вывести ее на экран, повторив 10 раз.
- L = [3, 'hello', 7, 4, 'привет', 4, 3, -1] Определите наличие строки «привет» в списке. ЕСЛИ такая строка в списке присутствует, ТО удалить ее из списка, ИНАЧЕ добавить строку в список. Подсчитать, сколько раз в списке встречается число 4, ЕСЛИ больше одного раза, ТО очистить список.
- Напишите программу, которая запрашивает у пользователя две строки и формирует из этих строк список. Если строки состоят только из чисел, то программа добавляет в середину списка сумму введенных чисел, иначе добавляется строка, образованная из слияния двух введенных ранее строк. Итоговая строка выводится на экран.
- Задан список слов. Необходимо выбрать из него случайное слово. Из выбранного случайного слова случайно выбрать букву и попросить пользователя ее угадать.
  Задан список слов: ['самовар', 'весна', 'лето']
  Выбираем случайное слово: 'весна'
  Выбираем случайную букву: 'с'
  Выводим на экран: ве?на
  Пользователь пытается угадать букву.
  Подсказка: используйте метод choice() модуля random.
- Найдите все значения функции y (x) = x2 + 3 на интервале от 10 до 30 с шагом 2.
- L = [-8, 8, 6.0, 5, 'строка', -3.1] Определить сумму чисел, входящих в список L. *Подсказка*: для определения типа объекта можно воспользоваться сравнением вида type(-8) == int.
- Дан список числовых значений, насчитывающий N элементов. Поменяйте местами первую и вторую половины списка.