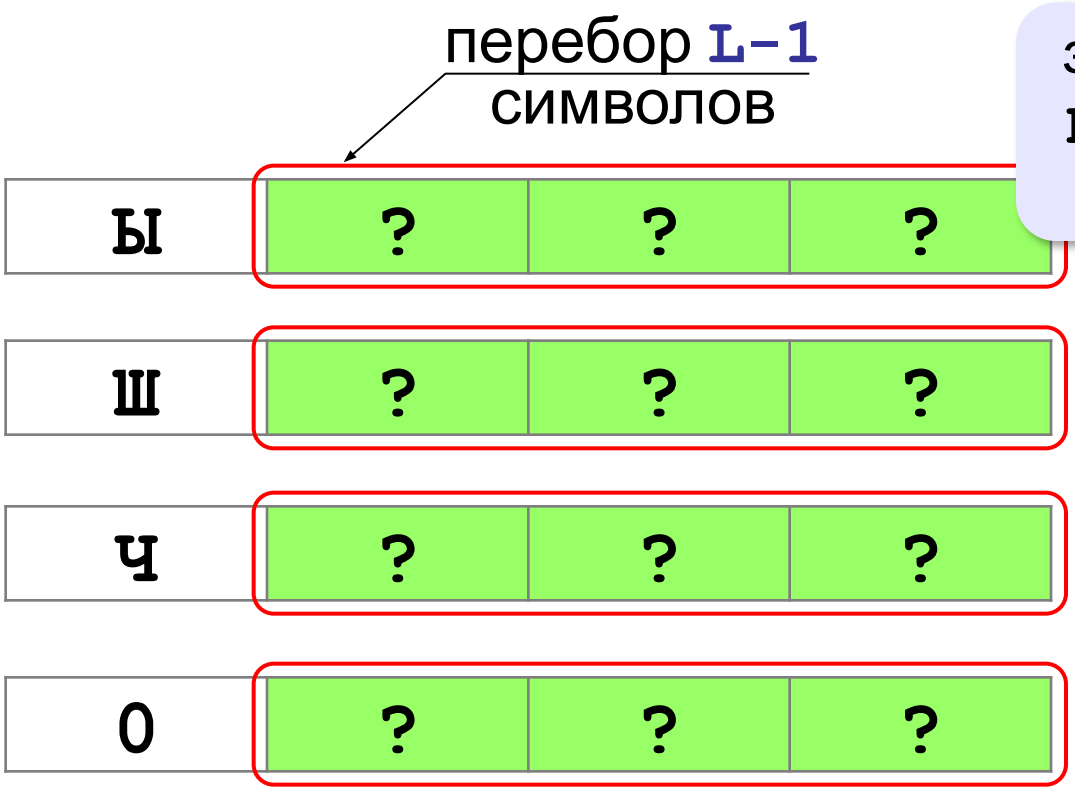
# Рекурсивный перебор

*Задача*. В алфавите языка племени «тумба-юмба» четыре буквы: «**Ы**», «**Ш**», «**Ч**» и «**О**». Нужно вывести на экран все слова, состоящие из **L** букв, которые можно построить из букв этого алфавита.




задача для слов длины **K** сведена к задаче для слов длины **L-1**!

# Рекурсивный перебор

```
перебор L символов
   w[0]="Ы"
    # перебор последних L-1 символов
   w[0]="Ш"
    # перебор последних L-1 символов
   w[0]="Ч"
    # перебор последних L-1 символов
   w[0]="О"
    # перебор последних L-1 символов
```

# Рекурсивный перебор

алфавит

слово

нужная длина слова

```
def TumbaWords ( A, w, L ):
  if len(w) == L:
    print ( w )
    return
  for c in A:
    TumbaWords ( A, w+c, L )
```

слово полной длины

по всем символам алфавита

```
# основная программа
TumbaWords ( "ЫШЧО", "", 3 );
```

# Задачи

**«A»:** В алфавите языке племени «тумба-юмба» четыре буквы: «Ы», «Ш», «Ч» и «О». Нужно вывести на экран все возможные слова, состоящие из K букв, в которых вторая буква «Ы». Подсчитайте количество таких слов.

**«B»:** В алфавите языке племени «тумба-юмба» четыре буквы: «Ы», «Ш», «Ч» и «О». Нужно вывести на экран все возможные слова, состоящие из K букв, в которых есть по крайней мере две одинаковые буквы, стоящие рядом. Подсчитайте количество таких слов.
Программа не должна строить другие слова, не соответствующие условию.

# Сравнение строк

**Пар ? пар ? парк**

Сравнение по кодам символов:

| | 0 | 1 | ... | 8 | 9 |
|---|---|---|---|---|---|
| CP-1251 | 48 | 49 | ... | 56 | 57 |
| UNCODE | 48 | 49 | ... | 56 | 57 |
| | **A** | **B** | ... | **Y** | **Z** |
| CP-1251 | 65 | 66 | ... | 89 | 90 |
| UNCODE | 65 | 66 | ... | 89 | 90 |
| | **a** | **b** | ... | **y** | **z** |
| CP-1251 | 97 | 98 | ... | 121 | 122 |
| UNCODE | 97 | 98 | ... | 121 | 122 |

# Сравнение строк

| | А | Б | ... | Ё | ... | Ю | Я |
|---|---|---|---|---|---|---|---|
| CP-1251 | 192 | 193 | ... | 168 | ... | 222 | 223 |
| UNCODE | 1040 | 1041 | ... | 1025 | ... | 1070 | 1071 |

| | а | б | ... | ё | ... | ю | я |
|---|---|---|---|---|---|---|---|
| CP-1251 | 224 | 225 | ... | 184 | ... | 254 | 255 |
| UNCODE | 1072 | 1073 | ... | 1105 | ... | 1102 | 1103 |

**5STEAM < STEAM < Steam < steam**

**steam < ПАР < Пар < пАр < пар < парк**

# Сортировка строк

```
aS = []     # пустой список строк
print ( "Введите строки для сортировки:" )
while True:
   s1 = input()
   if s1 == "": break
   aS.append ( s1 ) # добавить в список
aS.sort()           # сортировка
print ( aS )
```

# Задачи

**«А»:** Вводится 5 строк, в которых сначала записан порядковый номер строки с точкой, а затем – слово. Вывести слова в алфавитном порядке.

**Пример**:

```
Введите 5 строк:
1. тепловоз
2. арбуз
3. бурундук
4. кефир
5. урядник
Список слов в алфавитном порядке:
арбуз, бурундук, кефир, тепловоз, урядник
```

# Задачи

**«B»:** Вводится несколько строк (не более 20), в которых сначала записан порядковый номер строки с точкой, а затем – слово. Ввод заканчивается пустой строкой. Вывести введённые слова в алфавитном порядке.

**Пример**:

```
Введите слова:
1. тепловоз
2. арбуз

Список слов в алфавитном порядке:
арбуз, тепловоз
```

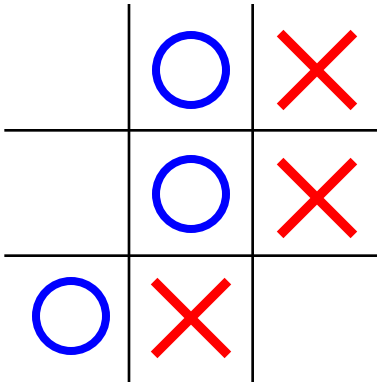
# Программирование на языке Python

## § 67. Матрицы

# Что такое матрица?

нолик

нет знака



крестик

строка 1, столбец 2

? Как закодировать?

**Матрица** — это прямоугольная таблица, составленная из элементов одного типа (чисел, строк и т.д.). Каждый элемент матрицы имеет два индекса – номера строки и столбца.

# Создание матриц

Матрица – это список списков!

```
A = [[-1, 0, 1],
     [-1, 0, 1],
     [0, 1, -1]]
```

перенос на другую строку внутри скобок

**или так:**

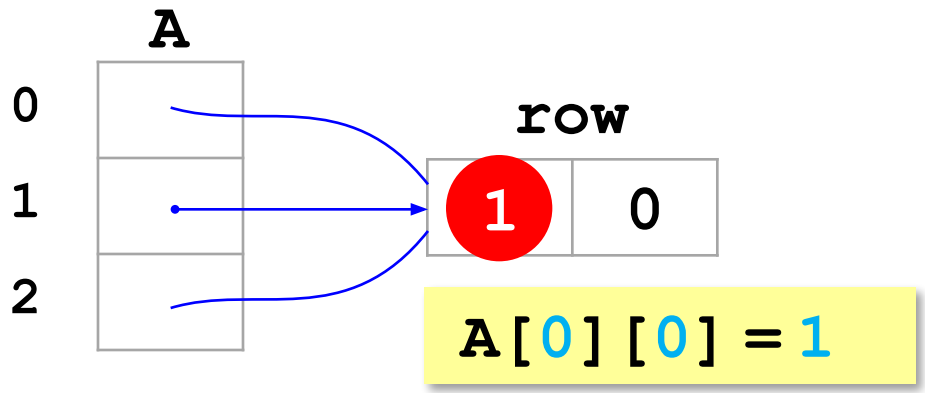
```
A = [[-1, 0, 1], [-1, 0, 1], [0, 1, -1]]
```

Нумерация элементов с нуля!

# Создание матриц

## Нулевая матрица:

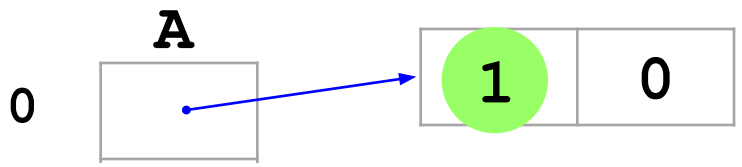
```
N = 3
M = 2
row = [0]*M
A = [row]*N
```




```
A[0][0] = 1
```

## а правильно так:

```
A = []
for i in range(N):
    A.append( [0]*M )
```




```
A[0][0] = 1
```

# Вывод матриц

```
print ( A )
```

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

```
def printMatrix ( A ):
  for row in A:
    for x in row:
      print ( "{:4d}".format(x), end="" )
    print ()
```

```
   1   2   3
   4   5   6
   7   8   9
```

? Зачем форматный вывод?

# Простые алгоритмы

## Заполнение случайными числами:

```
import random
for i in range(N):
  for j in range(M):
    A[i][j] = random.randint( 20, 80 )
    print( "{:4d}".format(A[i][j]),
           end="" )
  print()
```

! Вложенный цикл!

## Суммирование:

```
s = 0
for i in range(N):
  for j in range(M):
    s += A[i][j]
print( s )
```

```
s = 0
for row in A:
  s += sum(row)
print( s )
```

# Задачи

**«A»:** Напишите программу, которая заполняет квадратную матрицу случайными числами в интервале [10,99], и находит максимальный и минимальный элементы в матрице и их индексы.

**Пример**:

```
Матрица A:
12 14 67 45
32 87 45 63
69 45 14 11
40 12 35 15
Максимальный элемент A[2,2]=87
Минимальный элемент A[3,4]=11
```

# Задачи

**«B»:** Яркости пикселей рисунка закодированы числами от 0 до 255 в виде матрицы. Преобразовать рисунок в черно-белый по следующему алгоритму:

1) *вычислить среднюю яркость пикселей по всему рисунку*
2) *все пиксели, яркость которых меньше средней, сделать черными (записать код 0), а остальные – белыми (код 255)*

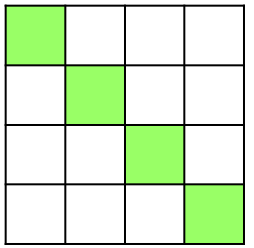
**Пример:**

```
Матрица A:
12 14 67 45
32 87 45 63
69 45 14 11
40 12 35 15
Средняя яркость 37.88
Результат:
  0   0 255 255
  0 255 255 255
255 255   0   0
255   0   0   0
```

# Перебор элементов матрицы

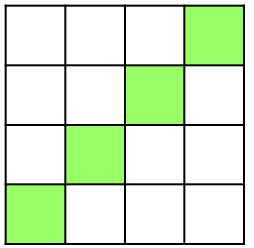
## Главная диагональ:

```
for i in range(N):
  # работаем с A[i][i]
```



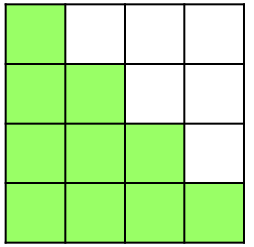
## Побочная диагональ:

```
for i in range(N):
  # работаем с A[i][N-1-i]
```



## Главная диагональ и под ней:

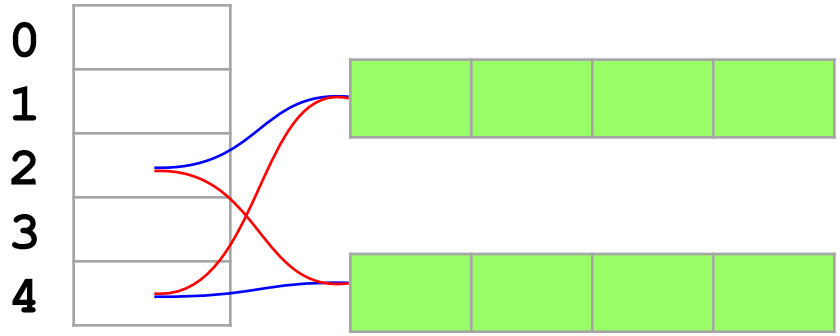
```
for i in range(N):
  for j in range(i+1):
    # работаем с A[i][j]
```



# Перестановка строк и столбцов

## 2-я и 4-я строки:

```
A[2], A[4] = A[4], A[2]
```




## 2-й и 4-й столбцы:

```
for i in range(N):
    A[i][2], A[i][4] = A[i][4], A[i][2]
```

# Выделение строк и столбцов

## 1-я строка:

```
R = A[1][:]
```

~~`R = A[1]`~~

? Почему плохо?

## 2-й столбец:

```
C = []
for row in A:
    C.append(row[2])
```

или так:

```
C = [ row[2] for row in A ]
```

## главная диагональ:

```
D = [ A[i][i] for i in range(N) ]
```

# Задачи

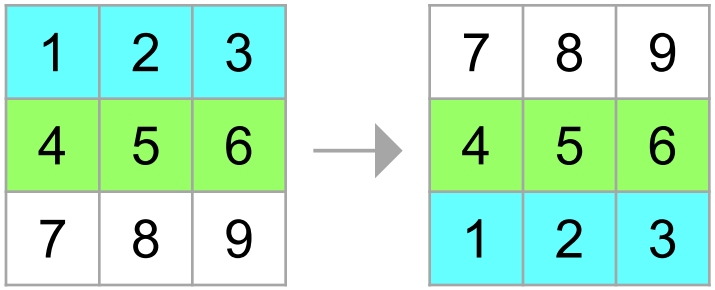
**«А»:** Напишите программу, которая заполняет квадратную матрицу случайными числами в интервале [10,99], а затем записывает нули во все элементы выше главной диагонали. Алгоритм не должен изменяться при изменении размеров матрицы.

**Пример**:

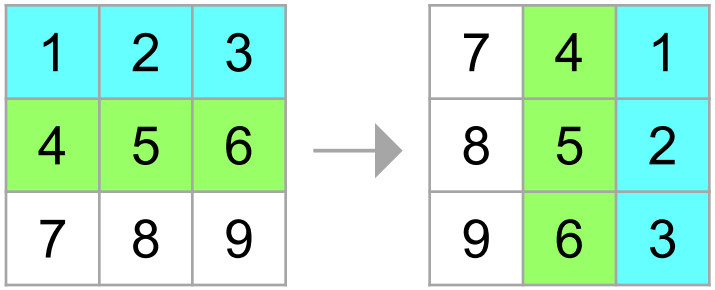
```
Матрица A:
12 14 67 45
32 87 45 63
69 45 14 30
40 12 35 65
Результат:
12  0  0  0
32 87  0  0
69 45 14  0
40 12 35 65
```

# Задачи

**«B»:** Пиксели рисунка закодированы числами (обозначающими цвет) в виде матрицы, содержащей N строк и M столбцов. Выполните отражение рисунка сверху вниз:



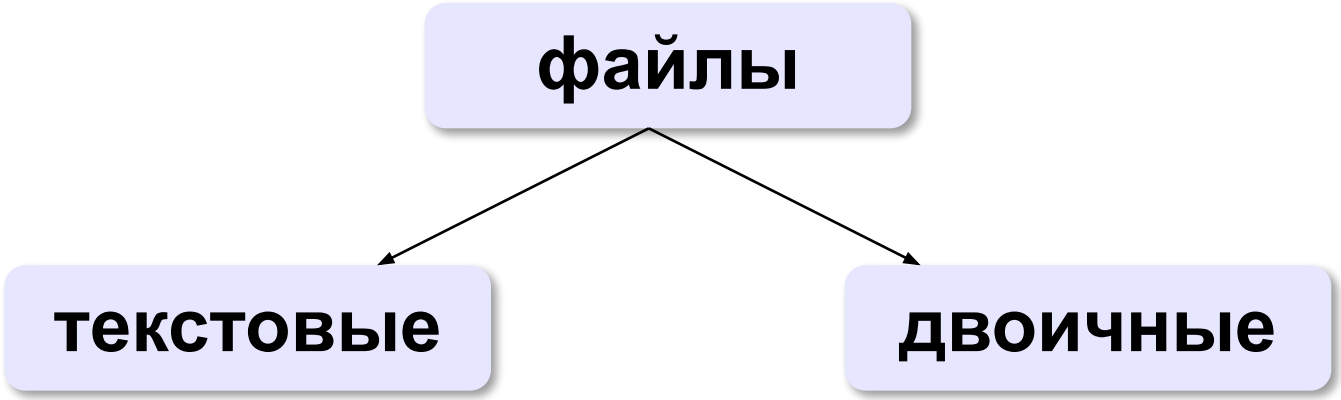

**«C»:** Пиксели рисунка закодированы числами (обозначающими цвет) в виде матрицы, содержащей N строк и M столбцов. Выполните поворот рисунка вправо на 90 градусов:



# Программирование на языке Python

## § 68. Работа с файлами

# Как работать с файлами?




«*plain text*»:
- текст, разбитый на строки;
- из специальных символов только символы перехода на новую строку

- любые символы
- рисунки, звуки, видео, …

# Принцип сэндвича

файловые переменные-*указатели*

по умолчанию – на чтение (режим `"r"`)

```
Fin = open ( "input.txt" )
Fout = open ( "output.txt", "w" )
  # здесь работаем с файлами
Fin.close()
Fout.close()
```

```
"r" - чтение
"w" - запись
"a" - добавление
```

# Ввод данных

```
Fin = open( "input.txt" )
```

**Чтение строки**:

```
s = Fin.readline()      # "1 2"
```

**Чтение строки и разбивка по пробелам**:

```
s = Fin.readline().split()      # ["1","2"]
```

**Чтение целых чисел**:

```
s = Fin.readline().split()      # ["1","2"]
a, b = int(s[0]), int(s[1])
```

**или так**:

```
a, b = [int(x) for x in s]
```

**или так**:

```
a, b = map( int, s )
```

# Вывод данных в файл

```
a = 1
b = 2
Fout = open( "output.txt", "w" )
Fout.write ( "{:d} + {:d} = {:d}\n".format(
                  a, b, a+b) )
Fout.close()
```

Все данные преобразовать в строку!

# Чтение неизвестного количества данных

*Задача*. В файле записано в столбик неизвестное количество чисел. Найти их сумму.

```
пока не конец файла
  прочитать число из файла
  добавить его к сумме
```

```
Fin = open ( "input.txt" )
sum = 0
while True:
  s = Fin.readline()
  if not s: break
  sum += int(s)
Fin.close()
```

если конец файла, вернёт пустую строку

# Чтение неизвестного количества данных

*Задача*. В файле записано в столбик неизвестное количество чисел. Найти их сумму.

```
sum = 0
Fin = open ( "input.txt" )
lst = Fin.readlines()
for s in lst:
  sum += int(s)
Fin.close()
```

прочитать все строки в список строк

# Чтение неизвестного количества данных

*Задача*. В файле записано в столбик неизвестное количество чисел. Найти их сумму.

```
sum = 0
with open ( "input.txt" ) as Fin:
  for s in Fin:
    sum += int(s)
```

**или так**:

```
sum = 0
for s in open ( "input.txt" ):
  sum += int(s)
```

Не нужно закрывать файл!

# Задачи

**«A»:** Напишите программу, которая находит среднее арифметическое всех чисел, записанных в файле в столбик, и выводит результат в другой файл.

**«B»:** Напишите программу, которая находит минимальное и максимальное среди чётных положительных чисел, записанных в файле, и выводит результат в другой файл. Учтите, что таких чисел может вообще не быть.

**«C»:** В файле в столбик записаны целые числа, сколько их – неизвестно. Напишите программу, которая определяет длину самой длинной цепочки идущих подряд одинаковых чисел и выводит результат в другой файл.

# Обработка массивов

*Задача*. В файле записаны в столбик целые числа. Вывести в другой текстовый файл те же числа, отсортированные в порядке возрастания.

В чем отличие от предыдущей задачи?

Для сортировки нужно удерживать все элементы в памяти одновременно.

# Обработка массивов

## Ввод массива:

```
A = []
while True:
   s = Fin.readline()
   if not s: break
   A.append ( int(s) )
```

## Ввод в стиле Python:

```
s = Fin.read().split()
A = list ( map(int, s) )
```

## Сортировка:

```
A.sort()
```

# Обработка массивов

## Вывод результата:

```
Fout = open ( "output.txt", "w" )
Fout.write ( str(A) )
Fout.close()
```

```
[1, 2, 3]
```

## или так:

```
for x in A:
    Fout.write ( str(x)+"\n" )
```

```
1
2
3
```

## или так:

```
for x in A:
    Fout.write ( "{:4d}".format(x) )
```

```
   1   2   3
```

# Задачи

**«A»:** В файле в столбик записаны числа. Отсортировать их по возрастанию последней цифры и записать в другой файл.

**«B»:** В файле в столбик записаны числа. Отсортировать их по возрастанию суммы цифр и записать в другой файл. Используйте функцию, которая вычисляет сумму цифр числа.

**«C»:** В двух файлах записаны отсортированные по возрастанию массивы неизвестной длины. Объединить их и записать результат в третий файл. Полученный массив также должен быть отсортирован по возрастанию.

# Обработка строк

*Задача*. В файле записано данные о собаках: в каждой строчке кличка собаки, ее возраст и порода:

```
Мухтар 4 немецкая овчарка
```

Вывести в другой файл сведения о собаках, которым меньше 5 лет.

```
пока не конец файла Fin
  прочитать строку из файла Fin
  разобрать строку – выделить возраст
  если возраст < 5 то
    записать строку в файл Fout
```

# Чтение данных из файла

**Чтение одной строки**:

```
s = Fin.readline()
```

**Разбивка по пробелам**:

```
data = s.split()
```

**Выделение возраста**:

```
sAge = data[1]
age = int ( sAge )
```

**Кратко всё вместе**:

```
s = Fin.readline()
age = int ( s.split()[1] )
```

# Обработка строк

## Полная программа:

```
Fin = open ( "input.txt" )
Fout = open ( "output.txt", "w" )
while True:
  s = Fin.readline()
  if not s: break
  age = int ( s.split()[1] )
  if age < 5:
    Fout.write ( s )
Fin.close()
Fout.close()
```

# Обработка строк

**или так:**

```
lst = Fin.readlines()
for s in lst:
  age = int( s.split()[1] )
  if age < 5:
    Fout.write( s )
```

**или так:**

```
for s in open( "input.txt" ):
  age = int( s.split()[1] )
  if age < 5:
    Fout.write( s )
```

# Задачи

**«A»:** В файле записаны данные о результатах сдачи экзамена. Каждая строка содержит фамилию, имя и количество баллов, разделенные пробелами:

```
<Фамилия> <Имя> <Количество баллов>
```

Вывести в другой файл фамилии и имена тех учеников, которые получили больше 80 баллов.

**«B»:** В предыдущей задаче добавить к полученному списку нумерацию, сократить имя до одной буквы и поставить перед фамилией:

```
1. Иванов
2. Петров
...
```