

# Программирование на языке Python

§ 62. Массивы

§ 63. Алгоритмы обработки массивов

§ 64. Сортировка

§ 65. Двоичный поиск

§ 66. Символьные строки

§ 67. Матрицы

§ 68. Работа с файлами

# Программирование на языке Python

## § 62. Массивы

# Что такое массив?

---



Как ввести 10000 переменных?

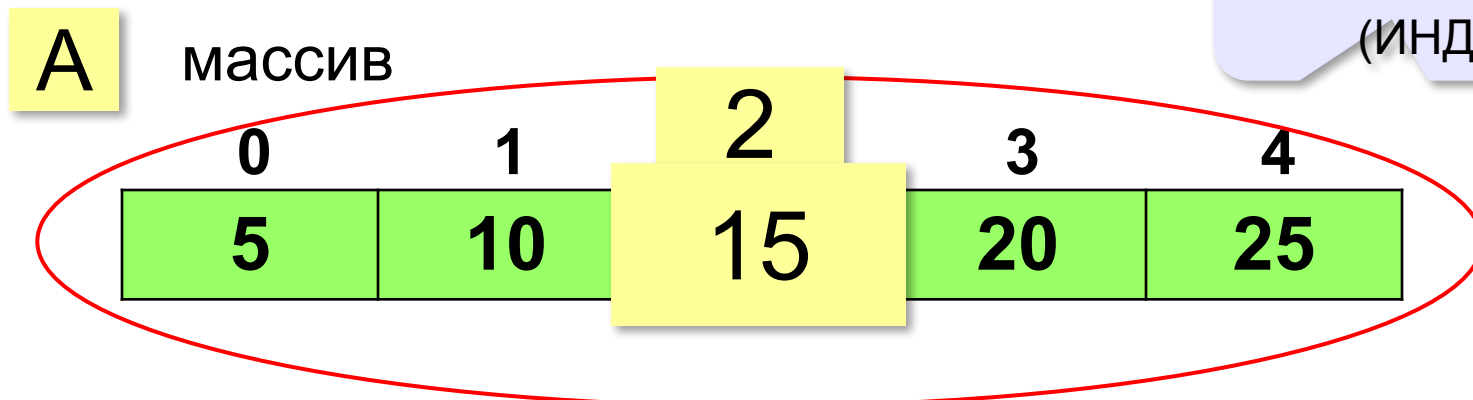
**Массив** – это группа переменных одного типа, расположенных в памяти рядом (в соседних ячейках) и имеющих общее имя. Каждая ячейка в массиве имеет уникальный номер (индекс).

**Надо:**

- выделять память
- записывать данные в нужную ячейку
- читать данные из ячейки

# Что такое массив?

**!** Массив = таблица!



**A[0]**   **A[1]**   **ЗНАЧЕНИЕ**  
элемента массива   **A[4]**

**A[2]**

НОМЕР (ИНДЕКС)  
элемента массива: 2

ЗНАЧЕНИЕ  
элемента массива: 15

# Массивы в Python: списки

---


```
A = [1, 3, 4, 23, 5]
```

```
A = [1, 3] + [4, 23] + [5]
```

```
[1, 3, 4, 23, 5]
```

```
A = [0] * 10
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

 Что будет?

```
A = list ( range (10) )
```

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

# Генераторы списков

```
A = [ i for i in range(10) ]
```

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



Что будет?

```
A = [ i*i for i in range(10) ]
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
from random import randint
```

```
A = [ randint(20, 100)  
      for x in range(10) ]
```

случайные  
числа

```
A = [ i for i in range(100)  
      if isPrime(i) ]
```

условие  
отбора

# Добавление элементов

```
A = [1, 2, 3]
```

В конец списка

```
x = 5
```

```
A.append ( x+3 ) # [1, 2, 3, 8]
```

**Метод** – операция, которую можно применить к списку.

```
A = [1, 2, 3]
```

```
A.insert ( 1, 35 ) # [1, 35, 2, 3]
```

?

В начало?

A[1]

```
A.insert ( 0, 90 )
```

```
A = [90] + A
```

## Удаление элементов

```
A = [1, 2, 3]
x = A.pop(1) # x = 2, A = [1, 3]
```

удалить A[1]

```
A = [1, 2, 3]
x = A.pop() # x = 3, A = [1, 2]
```

удалить последний

```
A = [11, 29, 37, 45]
A.remove(37) # A = [11, 29, 45]
```



# Ввод массива с клавиатуры

## Создание массива:

```
N = 10  
A = [0]*N
```

## Ввод с клавиатуры:

```
for i in range(N):  
    print ( "A[" , i , "]" = " ,  
           sep = " " , end = " " )  
    A[i] = int( input() )
```

```
A[0] = 5  
A[1] = 12  
A[2] = 34  
A[3] = 56  
A[4] = 13
```

```
sep = "  
end = "
```

не разделять  
элементы

не переходить на  
новую строку

# Ввод массива с клавиатуры

## Ввод без подсказок:

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

## Ввод в одной строке:

```
data = input()      # "1 2 3 4 5"  
s = data.split()   # ["1", "2", "3", "4", "5"]  
A = [ int(x) for x in s ]  
                  # [1, 2, 3, 4, 5]
```

## или так:

```
s = input().split() # ["1", "2", "3", "4", "5"]  
A = list( map(int, s) ) # [1, 2, 3, 4, 5]
```

построить  
СПИСОК

применить `int` ко  
ВСЕМ ЭЛЕМЕНТАМ `s`

# Вывод массива на экран

---

Как список:

```
print ( A ) [1, 2, 3, 4, 5]
```

В строчку через пробел:

```
for i in range(N):  
    print ( A[i], end=" " ) 1 2 3 4 5
```

или так:

```
for x in A:  
    print ( x, end=" " ) 1 2 3 4 5
```

или так:

```
print ( *A ) ↔ print ( 1, 2, 3, 4, 5 )
```

# Как обработать все элементы массива?

Создание массива:

```
N = 5  
A = [0] * N
```

Обработка:

```
# обработать A[0]  
# обработать A[1]  
# обработать A[2]  
# обработать A[3]  
# обработать A[4]
```



1) если N велико (1000, 1000000)?

2) при изменении N программа не должна меняться!

# Как обработать все элементы массива?

Обработка с переменной:

```
i = 0;  
# обработать A[i]  
i += 1  
# обработать A[i]  
i += 1  
# обработать A[i]  
i += 1  
# обработать A[i]  
i += 1  
# обработать A[i]  
i += 1
```



Обработка в цикле:

```
i = 0  
while i < N:  
    # обработать A[i]  
    i += 1
```

Цикл с переменной:

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



# Заполнение случайными числами

```
from random import randint
N = 10
A = [0]*N
for i in range(N):
    A[i] = randint(20, 100)
```

или так:

```
from random import randint
N = 10
A = [ randint(20, 100)
      for x in range(N) ]
```

случайные  
числа  
[20, 100]

# Перебор элементов

---

Общая схема (можно изменять  $A[i]$ ):

```
for i in range(N):  
    ... # сделать что-то с A[i]
```

```
for i in range(N):  
    A[i] += 1
```

Если не нужно изменять  $A[i]$ :

```
for x in A:  
    ... # сделать что-то с x
```

$x = A[0], A[1], \dots, A[N-1]$

```
for x in A:  
    print ( x )
```

# Подсчёт количества нужных элементов

Задача. В массиве записаны данные о росте баскетболистов. Сколько из них имеет рост больше 180 см, но меньше 190 см?

**?** Как решать?

```
count = 0
for x in A:
    if 180 < x and x < 190:
        count += 1
```

Python:  
`180 < x < 190`



# Сумма элементов массива

---

```
summa = 0
for x in A:
    if 180 < x < 190:
        summa += x
print ( summa )
```

или так:

```
print ( sum(A) )
```

# Перебор элементов

## Среднее арифметическое:

```
count = 0
summa = 0
for x in A:
    if 180 < x < 190:
        count += 1
        summa += x
print ( summa/count )
```

среднее  
арифметическое

## или так:

```
B = [ x for x in A
      if 180 < x < 190 ]
print ( sum(B)/len(B) )
```

отбираем нужные

# Задачи

---

**«А»:** Заполните массив случайными числами в интервале  $[0,100]$  и найдите среднее арифметическое его значений.

**Пример:**

**Массив :**

**1 2 3 4 5**

**Среднее арифметическое 3.000**

**«В»:** Заполните массив случайными числами в интервале  $[0,100]$  и подсчитайте отдельно среднее значение всех элементов, которые  $<50$ , и среднее значение всех элементов, которые  $\geq 50$ .

**Пример:**

**Массив :**

**3 2 52 4 60**

**Ср. арифм. элементов  $[0, 50)$  : 3.000**

**Ср. арифм. элементов  $[50, 100]$  : 56.000**

# Задачи

---

**«С»:** Заполните массив из  $N$  элементов случайными числами в интервале  $[1, N]$  так, чтобы в массив обязательно вошли все числа от 1 до  $N$  (постройте случайную перестановку).

**Пример:**

**Массив :**

3 2 1 4 5

# Программирование на языке Python

## **§ 63. Алгоритмы обработки массивов**

# Поиск в массиве

Найти элемент, равный X:

```
i = 0
while A[i] != X:
    i += 1
print ( "A[" , i , "]=" , X , sep = " " )
```



Что плохо?

```
i = 0
while i < N and A[i] != X:
    i += 1
if i < N:
    print ( "A[" , i , "]=" , X , sep = " " )
else:
    print ( "Не нашли!" )
```



Что если такого нет?

# Поиск в массиве

## Вариант с досрочным выходом:

номер найденного  
элемента

```
nX = -1
for i in range ( N ):
    if A[i] == X:
        nX = i
        break
if nX >= 0:
    print ( "A[" , nX, "]" = " , X, sep = " " )
else:
    print ( "Не нашли!" )
```

досрочный  
выход из цикла

# Поиск в массиве

## Варианты в стиле Python:

```
for i in range ( N ) :
    if A[i] == X:
        print ( "A[" , i , "]" = " , X , sep = "" )
        break
else:
    print ( "Не нашли!" )
```

если не было досрочного выхода из цикла

```
if X in A:
    nX = A.index (X)
    print ( "A[" , nX , "]" = " , X , sep = "" )
else:
    print ( "Не нашли!" )
```



# Задачи

---

«А»: Заполните массив случайными числами в интервале  $[0,5]$ . Введите число  $X$  и найдите все значения, равные  $X$ .

**Пример:**

Массив :

1 2 3 1 2

Что ищем :

2

Нашли:  $A[2]=2$ ,  $A[5]=2$

**Пример:**

Массив :

1 2 3 1 2

Что ищем :

6

Ничего не нашли.

# Задачи

---

**«В»:** Заполните массив случайными числами в интервале  $[0,5]$ . Определить, есть ли в нем элементы с одинаковыми значениями, стоящие рядом.

**Пример:**

**Массив :**

1 2 3 3 2 1

**Есть :** 3

**Пример:**

**Массив :**

1 2 3 4 2 1

**Нет**

# Задачи

---

**«С»:** Заполните массив случайными числами. Определить, есть ли в нем элементы с одинаковыми значениями, не обязательно стоящие рядом.

**Пример:**

**Массив :**

3 2 1 3 2 5

**Есть :** 3, 2

**Пример:**

**Массив :**

3 2 1 4 0 5

**Нет**

# Максимальный элемент

---

```
M = A[0]
```

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

```
    if A[i] > M:
```

```
        M = A[i]
```

```
print ( M )
```



Если `range(N)` ?

## Варианты в стиле Python:

```
M = A[0]
```

```
for x in A:
```

```
    if x > M:
```

```
        M = x
```



Как найти его номер?

```
M = max ( A )
```

# Максимальный элемент и его номер

```
M = A[0]; nMax = 0
for i in range(1, N):
    if A[i] > M:
        M = A[i]
        nMax = i
print( "A[" , nMax, "]=" , M, sep = " " )
```



Что можно улучшить?



По номеру элемента можно найти значение!

```
nMax = 0
for i in range(1, N):
    if A[i] > A[nMax]:
        nMax = i
print( "A[" , nMax, "]=" , A[nMax] , sep = " " )
```

# Максимальный элемент и его номер

---

## Вариант в стиле Python:

```
M = max (A)
nMax = A . index (M)
print ( "A[" , nMax , "]" = " , M , sep = " " )
```

номер заданного  
элемента (первого из...)

# Задачи

---

**«А»:** Заполнить массив случайными числами и найти минимальный и максимальный элементы массива и их номера.

**Пример:**

**Массив :**

1 2 3 4 5

**Минимальный элемент:  $A[1]=1$**

**Максимальный элемент:  $A[5]=5$**

**«В»:** Заполнить массив случайными числами и найти два максимальных элемента массива и их номера.

**Пример:**

**Массив :**

5 5 3 4 1

**Максимальный элемент:  $A[1]=5$**

**Второй максимум:  $A[2]=5$**

# Задачи

---

**«С»:** Введите массив с клавиатуры и найдите (за один проход) количество элементов, имеющих максимальное значение.

**Пример:**

**Массив :**

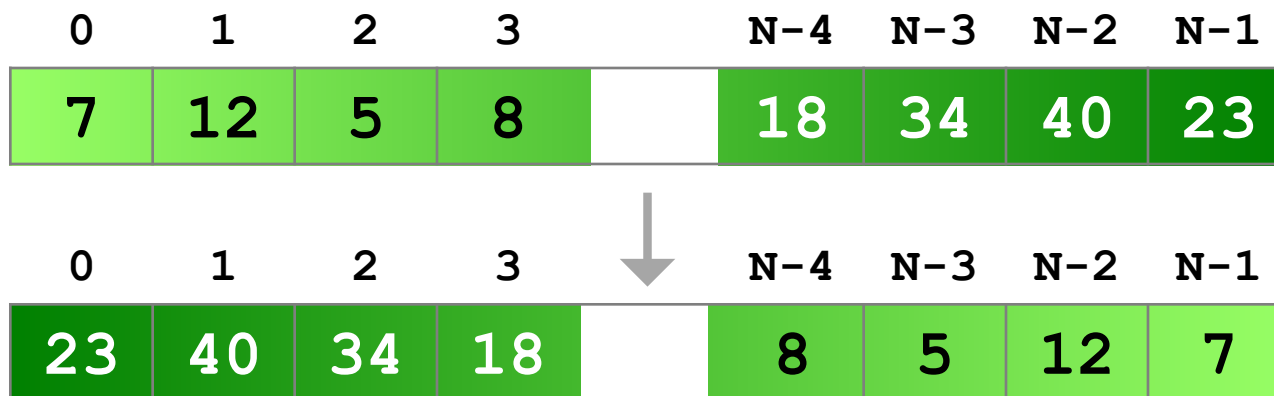
**3 4 5 5 3 4 5**

**Максимальное значение 5**

**Количество элементов 3**



# Реверс массива



«Простое» решение:

остановиться на середине!

```
for i in range(N//2):
    поменять местами A[i] и A[N-1-i]
```



Что плохо?

# Реверс массива

---

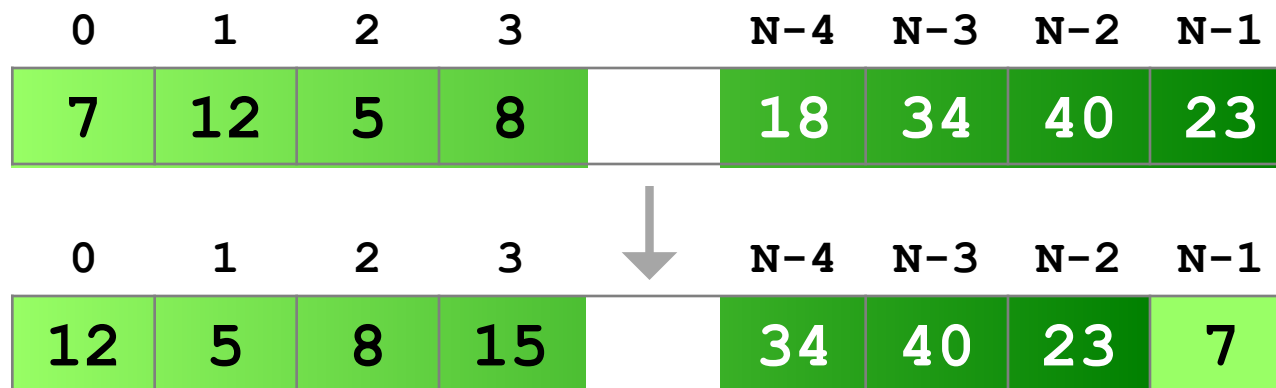
```
for i in range(N//2):  
    c = A[i]  
    A[i] = A[N-1-i]  
    A[N-1-i] = c
```

## Варианты в стиле Python:

```
for i in range(N//2):  
    A[i], A[N-i-1] = A[N-i-1], A[i]
```

```
A.reverse()
```

# Циклический сдвиг элементов



«Простое» решение:

```
for i in range(N-1):
    A[i] = A[i+1]
```

?

Почему не до N?

?

Что плохо?

# Срезы в Python

0	1	2	3		N-4	N-3	N-2	N-1
7	12	5	8		18	34	40	23

**!** Последний элемент не входит в срез!

$A[1:3] \rightarrow [12, 5]$

$A[2:3] \rightarrow [5]$

$A[:3] \rightarrow A[0:3] \rightarrow [7, 12, 5]$

с начала

$A[3:N-2] \rightarrow [8, \dots, 18, 34]$

$A[3:] \rightarrow A[3:N] \rightarrow [8, \dots, 18, 34, 40, 23]$

до конца

копия массива

$A[:] \rightarrow [7, 12, 5, 8, \dots, 18, 34, 40, 23]$

# Срезы в Python – отрицательные индексы

0	1	2	3		N-4	N-3	N-2	N-1
7	12	5	8		18	34	40	23
-N	-N+1	-N+2	-N+3		-4	-3	-2	-1

$A[1:-1]$   
 $A[1:N-1]$

→ [12, 5, 8, ..., 18, 34, 40]

$A[-4:-2]$   
 $A[N-4:N-2]$

→ [18, 34]

# Срезы в Python – шаг

0	1	2	3	4	5	6	7	8
7	12	5	8	76	18	34	40	23

шаг

`A[1:6:2]` → [12, 8, 18]

`A[::3]` → [7, 8, 34]

`A[8:2:-2]` → [23, 34, 76]

`A[::-1]` → [23, 40, 34, 18, 76, 8, 5, 12, 7]

реверс!

`A.reverse()`

# Задачи

---

**«А»:** Заполнить массив случайными числами и выполнить циклический сдвиг элементов массива вправо на 1 элемент.

**Пример:**

**Массив :**

1 2 3 4 5 6

**Результат:**

6 1 2 3 4 5

**«В»:** Массив имеет четное число элементов. Заполнить массив случайными числами и выполнить реверс отдельно в первой половине и второй половине.

**Пример:**

**Массив :**

1 2 3 4 5 6

**Результат:**

3 2 1 6 5 4

# Задачи

---

**«С»:** Заполнить массив случайными числами в интервале  $[-100, 100]$  и переставить элементы так, чтобы все положительные элементы стояли в начала массива, а все отрицательные и нули – в конце. Вычислите количество положительных элементов.

## Пример:

**Массив :**

20 -90 15 -34 10 0

**Результат:**

20 15 10 -90 -34 0

**Количество положительных элементов : 3**



# Отбор нужных элементов

**Задача.** Отобрать элементы массива **A**, удовлетворяющие некоторому условию, в массив **B**.

**Простое решение:**

```
B = []  
сделать для i от 0 до N-1  
    если условие выполняется для A[i] то  
        добавить A[i] к массиву B
```

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



Какие элементы выбираем?

добавить **x** в конец  
массива **B**

# Отбор нужных элементов

---

**Задача.** Отобрать элементы массива **A**, удовлетворяющие некоторому условию, в массив **B**.

**Решение в стиле Python:**

перебрать все  
элементы A

```
B = [ x for x in A  
      if x % 2 == 0 ]
```

если **x** – чётное  
число

# Особенности работы со списками

```
A = [1, 2, 3]
```

```
B = A
```

A → [1, 2, 3]  
B → [1, 2, 3]

→

A → [0, 2, 3]  
B → [0, 2, 3]

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

```
A = [1, 2, 3]
```

```
B = A[:]
```

копия массива A

A → [1, 2, 3]

→

A → [0, 2, 3]

B → [1, 2, 3]

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

B → [1, 2, 3]

# Копирование списков

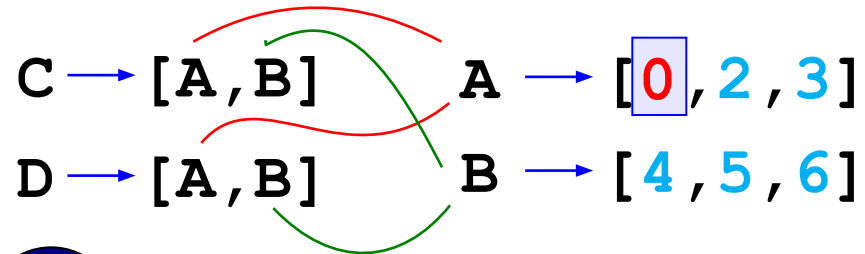
## «Поверхностное» копирование:

```
import copy
A = [1, 2, 3]
B = copy.copy(A)
```

A → [1, 2, 3]

B → [4, 5, 6]

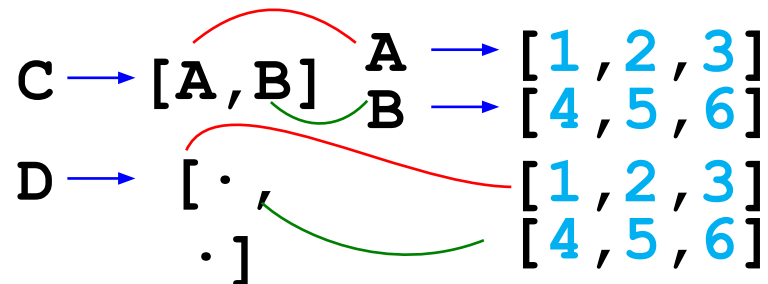
```
A = [1, 2, 3]
B = [4, 5, 6]
C = [A, B]
D = copy.copy(C)
C[0][0] = 0
```



Влияет на C и D!

## «Глубокое» копирование:

```
D = copy.deepcopy(C)
```



# Задачи

---

**«А»:** Заполнить массив случайными числами в интервале  $[-10, 10]$  и отобразить в другой массив все чётные отрицательные числа.

**Пример:**

**Массив А:**

-5 6 7 -4 -6 8 -8

**Массив В:**

-4 -6 -8

**«В»:** Заполнить массив случайными числами в интервале  $[0, 100]$  и отобразить в другой массив все простые числа. Используйте логическую функцию, которая определяет, является ли переданное ей число простым.

**Пример:**

**Массив А:**

12 13 85 96 47

**Массив В:**

13 47

# Задачи

---

**«С»:** Заполнить массив случайными числами и отобразить в другой массив все числа Фибоначчи. Используйте логическую функцию, которая определяет, является ли переданное ей число числом Фибоначчи.

**Пример:**

**Массив А:**

12 13 85 34 47

**Массив В:**

13 34

# Программирование на языке Python

## § 64. Сортировка

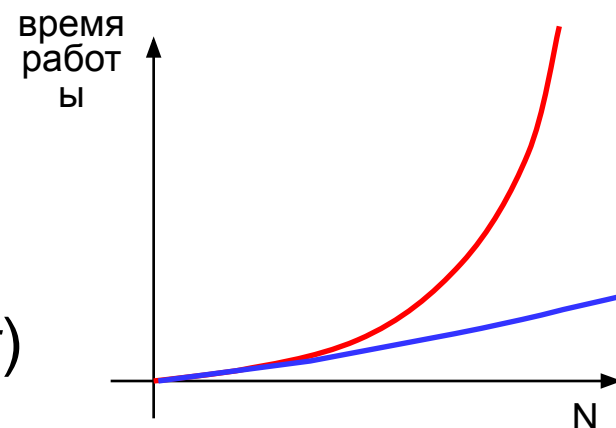
# Что такое сортировка?

**Сортировка** – это расстановка элементов массива в заданном порядке.

...по возрастанию, убыванию, последней цифре, сумме делителей, по алфавиту, ...

## Алгоритмы:

- простые и понятные, но неэффективные для больших массивов
  - **метод пузырька**
  - **метод выбора**
- сложные, но эффективные
  - **«быстрая сортировка»** (*QuickSort*)
  - сортировка «кучей» (*HeapSort*)
  - сортировка слиянием (*MergeSort*)
  - пирамидальная сортировка



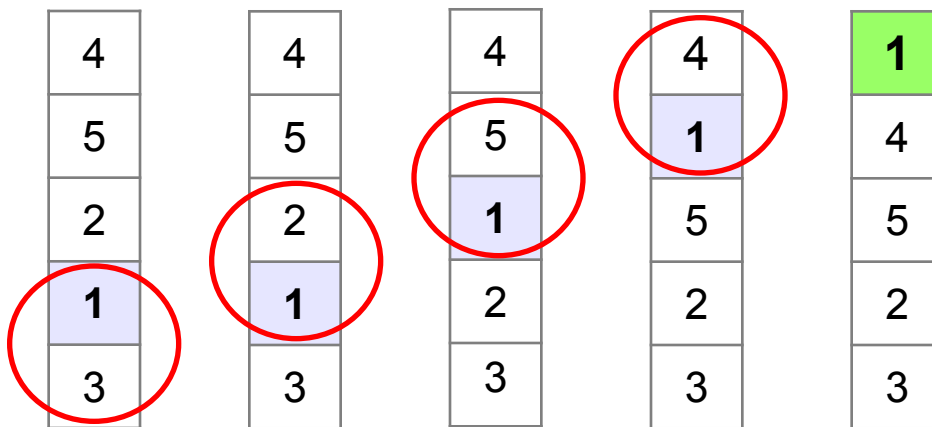


# Метод пузырька (сортировка обменами)

*Идея:* пузырек воздуха в стакане воды поднимается со дна вверх.

Для массивов – **самый маленький** («легкий» элемент перемещается вверх («всплывает»)).

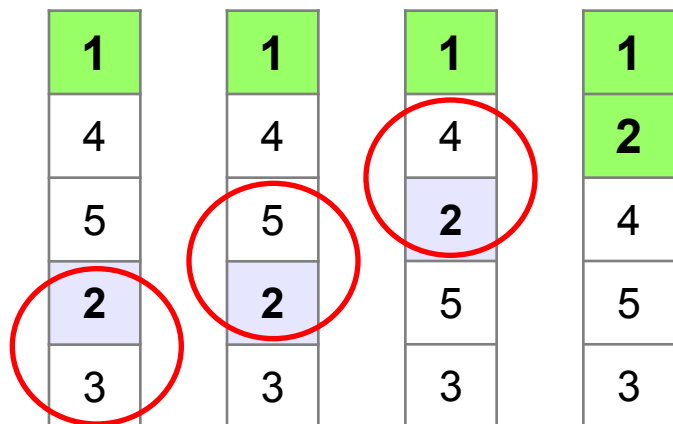
## 1-й проход:



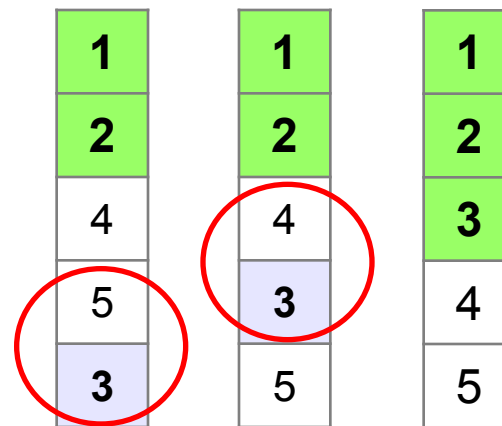
- сравниваем два соседних элемента; если они стоят «неправильно», меняем их местами
- за 1 проход по массиву **один** элемент (самый маленький) становится на свое место

# Метод пузырька

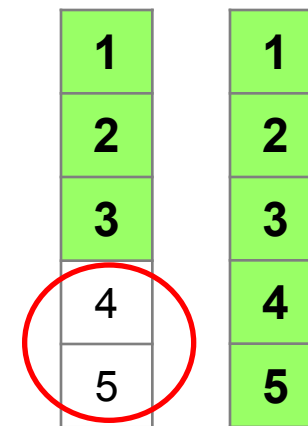
2-й проход:



3-й проход:



4-й проход:



Для сортировки массива из  $N$  элементов нужен  $N-1$  проход (достаточно поставить на свои места  $N-1$  элементов).

# Метод пузырька

## 1-й проход:

```
сделать для j от N-2 до 0 шаг -1
    если A[j+1] < A[j] то
        # поменять местами A[j] и A[j+1]
```

единственное  
отличие!

## 2-й проход:

```
сделать для j от N-2 до 1 шаг -1
    если A[j+1] < A[j] то
        # поменять местами A[j] и A[j+1]
```

# Метод пузырька

от  $N-2$  до  $0$  шаг  $-1$

1-й проход:

```
for j in range(N-2, -1, -1):  
    if A[j+1] < A[j]:  
        # поменять местами A[j] и A[j+1]
```

единственное  
отличие!

2-й проход:

```
for j in range(N-2, 0, -1):  
    if A[j+1] < A[j]:  
        # поменять местами A[j] и A[j+1]
```

# Метод пузырька

```
for i in range(N-1):  
    for j in range(N-2, i-1, -1):  
        if A[j+1] < A[j]:  
            A[j], A[j+1] = A[j+1], A[j]
```



Как написать метод «камня»?



Как сделать рекурсивный вариант?

# Задачи

---

- «А»: Напишите программу, в которой сортировка выполняется «методом камня» – самый «тяжёлый» элемент опускается в конец массива.
- «В»: Напишите вариант метода пузырька, который заканчивает работу, если на очередном шаге внешнего цикла не было перестановок.
- «С»: Напишите программу, которая сортирует массив по убыванию суммы цифр числа. Используйте функцию, которая определяет сумму цифр числа.

## Метод выбора (минимального элемента)

---

*Идея:* найти минимальный элемент и поставить его на первое место.

```
for i in range(N-1):  
    найти номер nMin минимального  
        элемента из A[i]..A[N]  
    if i != nMin:  
        поменять местами A[i] и A[nMin]
```

# Метод выбора (минимального элемента)

```
for i in range(N-1):  
    nMin = i  
    for j in range(i+1, N):  
        if A[j] < A[nMin]:  
            nMin = j  
    if i != nMin:  
        A[i], A[nMin] = A[nMin], A[i]
```



# Задачи

---

**«А»:** Массив содержит четное количество элементов. Напишите программу, которая сортирует первую половину массива по возрастанию, а вторую – по убыванию. Каждый элемент должен остаться в «своей» половине.

**Пример:**

**Массив :**

5 3 4 2 **1 6 3 2**

**После сортировки :**

2 3 4 5 **6 3 2 1**

# Задачи

---

**«В»:** Напишите программу, которая сортирует массив и находит количество различных чисел в нем.

**Пример:**

**Массив :**

**5 3 4 2 1 6 3 2 4**

**После сортировки:**

**1 2 2 3 3 4 4 5 6**

**Различных чисел: 5**

**«С»:** Напишите программу, которая сравнивает число перестановок элементов при использовании сортировки «пузырьком» и методом выбора. Проверьте ее на разных массивах, содержащих 1000 случайных элементов, вычислите среднее число перестановок для каждого метода.

# Сортировка слиянием

## Слияние отсортированных

### массивов:

A

6	34	67	82	98
---	----	----	----	----

B

44	55	78
----	----	----

C

--	--	--	--	--	--	--	--

```
Na = len(A); Nb = len(B)
iA = iB = 0; C = []
while iA < Na and iB < Nb:
    if A[iA] <= B[iB]:
        C.append(A[iA]); iA += 1
    else:
        C.append(B[iB]); iB += 1
C = C + A[iA:] + B[iB:]
```

пока оба массива  
непустые

добавить остаток

# Сортировка слиянием

```
def mergeSort( A ) :  
    if len(A) <= 1: return  
    mid = len(A) // 2  
    L = A[:mid]  
    R = A[mid:]  
    mergeSort(L)  
    mergeSort(R)  
    C = merge(L, R)  
    for i in range(len(A)) :  
        A[i] = C[i]
```

ВЫХОД ИЗ  
РЕКУРСИИ

РЕКУРСИВНЫЕ  
ВЫЗОВЫ

СЛИЯНИЕ

КОПИРУЕМ  
РЕЗУЛЬТАТ В  
МАССИВ **A**



Почему нельзя  $A = C$ ?

# Сортировка слиянием

---

## Процедура слияния:

```
def merge( A, B ) :
    Na = len(A) ; Nb = len(B)
    iA = iB = 0 ; C = []
    while iA < Na and iB < Nb :
        if A[iA] <= B[iB] :
            C.append(A[iA]) ; iA += 1
        else :
            C.append(B[iB]) ; iB += 1
    C = C + A[iA:] + B[iB:]
    return C
```

# Сортировка слиянием

---

 работает быстро

 нужен дополнительный массив

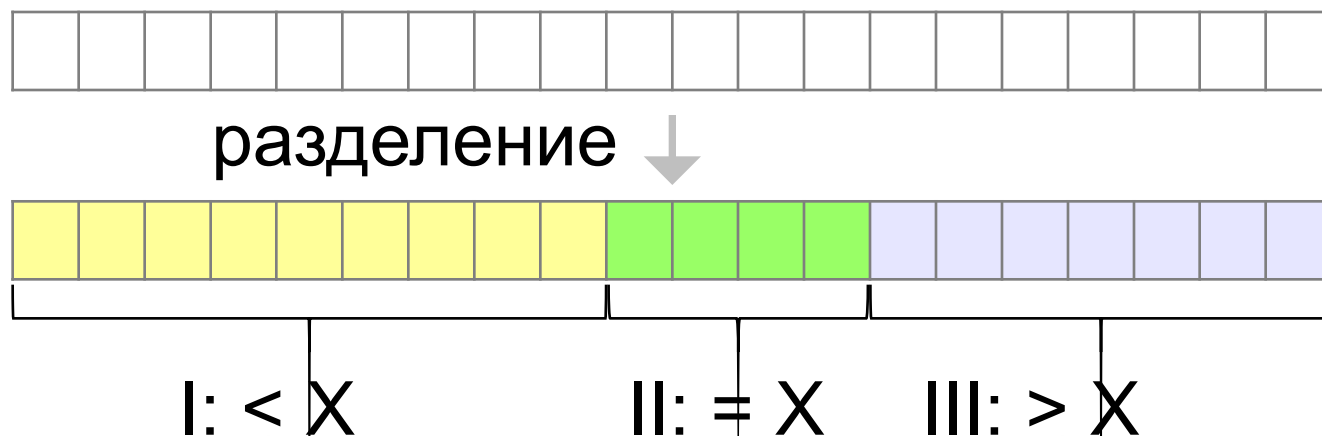
**«Разделяй и властвуй» (*divide and conquer*):**

- 1) задача разбивается на несколько подзадач меньшего размера;
- 2) эти подзадачи решаются с помощью рекурсивных вызовов того же (или другого) алгоритма;
- 3) решения подзадач объединяются, и получается решение исходной задачи.

# Быстрая сортировка (*QuickSort*)



Ч.Э.Хоар

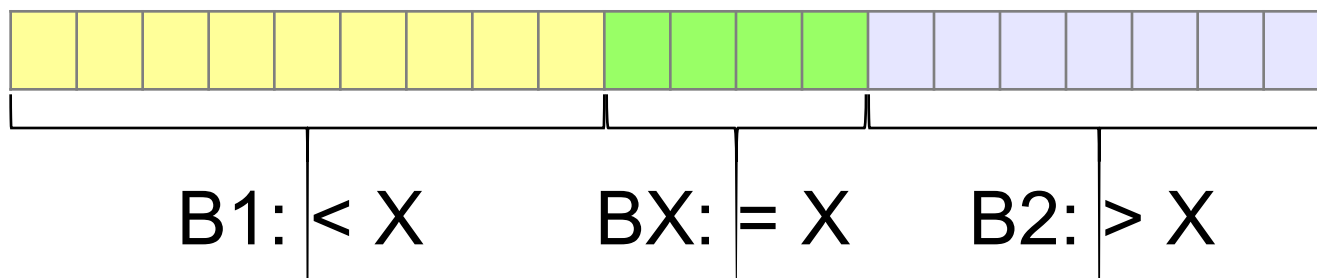


**!** Эти части нужно так же отсортировать!

**?** Как лучше выбирать  $X$ ?

**Медиана** – такое значение  $X$ , что слева и справа от него в отсортированном массиве стоит одинаковое число элементов (*долго искать ...*).

# Быстрая сортировка (QuickSort)



```
import random
def qSort ( A ) :
    if len(A) <= 1: return A
    X = random.choice(A)
    B1 = [ b for b in A if b < X ]
    BX = [ b for b in A if b == X ]
    B2 = [ b for b in A if b > X ]
    return qSort(B1) + BX + qSort(B2)
```



Где рекурсия?

расход  
памяти!

```
Asort = qSort( A )
```

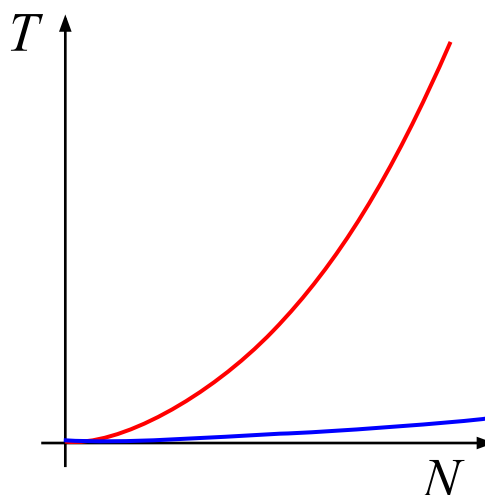


Что плохо?



# Сравнение алгоритмов сортировки

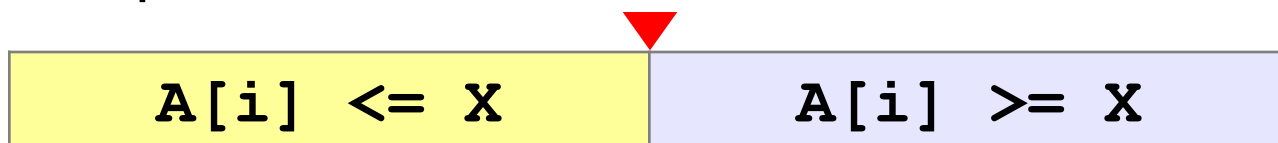
N	Метод пузырька	Метод выбора	Сортировка слиянием	Быстрая сортировка
1000	0,08 с	0,05 с	0,006 с	0,002 с
5000	1,8 с	1,3 с	0,033 с	0,006 с
15000	17,3 с	11,2 с	0,108 с	0,019 с

 $\sim N^2$ 

 $\sim N \log N$

## Быстрая сортировка «на месте»

**Шаг 1:** выбрать некоторый элемент массива  $X$

**Шаг 2:** переставить элементы так:



при сортировке элементы не покидают «свою область»!

**Шаг 3:** так же отсортировать две получившиеся области

Разделяй и властвуй (англ. *divide and conquer*)

78	6	82	67	55	44	34
----	---	----	----	----	----	----



Как лучше выбрать  $X$ ?

# Быстрая сортировка

## Разделение:

1) выбрать любой элемент массива ( $x=67$ )

53	48	82	67	6	48	95
L	L				R	R

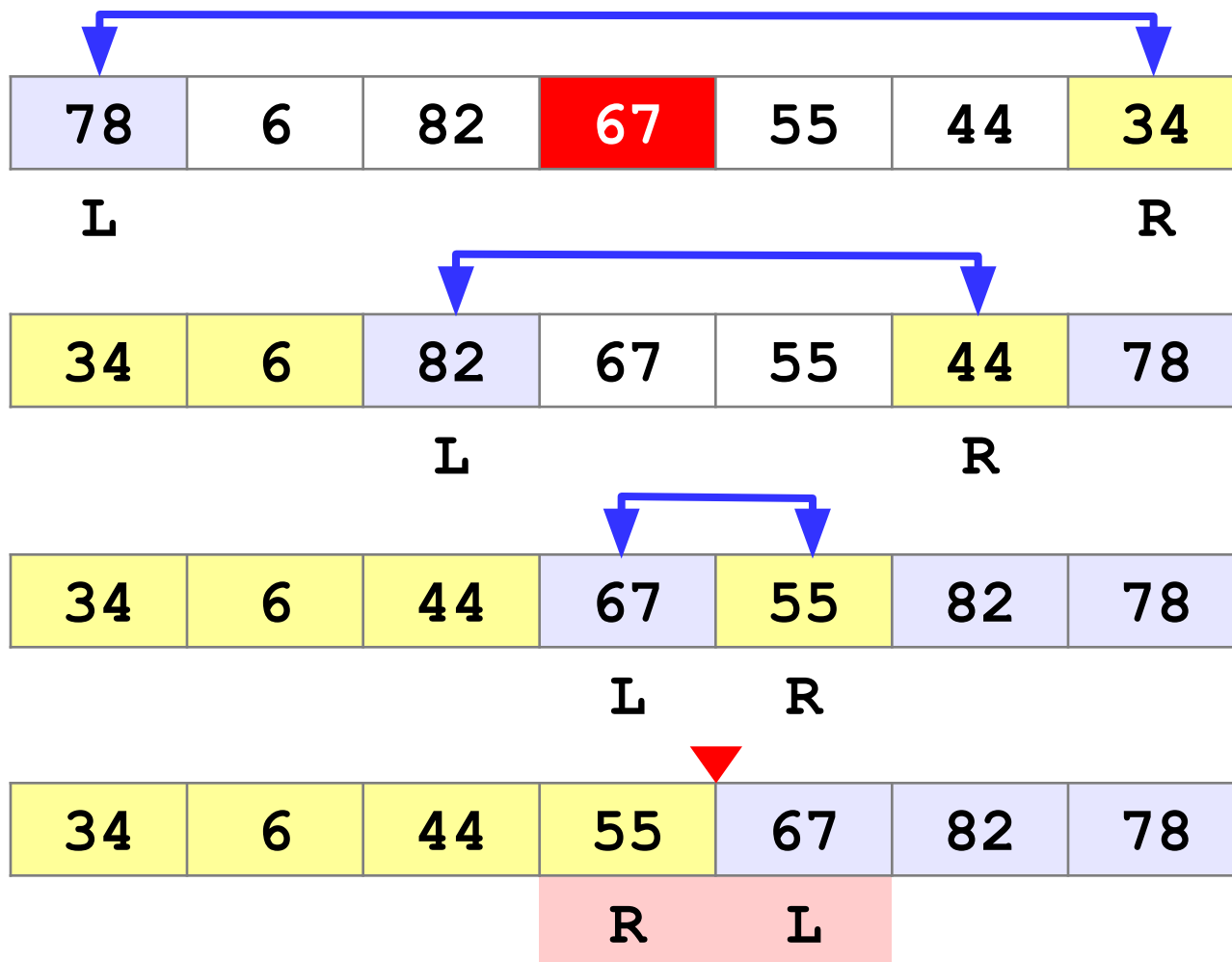
2) установить  $L = 0$ ,  $R = N-1$

3) увеличивая  $L$ , найти первый элемент  $A[L]$ ,  
который  $\geq x$  (должен стоять справа)

4) уменьшая  $R$ , найти первый элемент  $A[R]$ ,  
который  $\leq x$  (должен стоять слева)

5) если  $L \leq R$  то поменять местами  $A[L]$  и  $A[R]$   
и перейти к п. 3  
иначе **СТОП**.

# Быстрая сортировка



**L > R : разделение закончено!**

# Быстрая сортировка

---

## Основная программа:

```
N = 7
```

```
A = [0]*N
```

```
# заполнить массив
```

массив

начало

конец

```
qSort( A, 0, N-1 ) # сортировка
```

```
# вывести результат
```

# Быстрая сортировка

МАССИВ

начало

КОНЕЦ

```
def qSort ( A, nStart, nEnd ) :  
    if nStart >= nEnd: return  
    L = nStart; R = nEnd  
    X = A[ (L+R) // 2 ]  
    while L <= R:  
        while A[L] < X: L += 1  
        while A[R] > X: R -= 1  
        if L <= R:  
            A[L], A[R] = A[R], A[L]  
            L += 1; R -= 1  
    qSort ( A, nStart, R )  
    qSort ( A, L, nEnd )
```

разделение  
на 2 части

меняем местами

рекурсивные  
вызовы

# Быстрая сортировка

---

Случайный выбор элемента-разделителя:

```
from random import randint
def qSort ( A, nStart, nEnd ) :
    ...
    X = A [ randint ( L, R ) ]
    ...
```

или так:

```
from random import choice
def qSort ( A, nStart, nEnd ) :
    ...
    X = choice ( A [ L : R + 1 ] )
    ...
```

# Сортировка в Python

По возрастанию:

```
B = sorted( A )
```

алгоритм  
*Timsort*

По убыванию:

```
B = sorted( A, reverse = True )
```

По последней цифре:

```
def lastDigit ( n ):  
    return n % 10  
B = sorted( A, key = lastDigit )
```

или так:

```
B = sorted( A, key = lambda x: x % 10 )
```

«лямбда»-функция  
(функция без имени)



# Сортировка в Python – на месте

---

По возрастанию:

```
A.sort()
```

По убыванию:

```
A.sort ( reverse = True )
```

По последней цифре:

```
def lastDigit ( n ):  
    return n % 10  
A.sort ( key = lastDigit )
```

или так:

```
A.sort ( key = lambda x: x % 10 )
```

# Задачи

---

**«А»:** Массив содержит четное количество элементов.

Напишите программу, которая сортирует по возрастанию отдельно элементы первой и второй половин массива.

Каждый элемент должен остаться в «своей» половине.

Используйте алгоритм быстрой сортировки.

**Пример:**

Массив :

5 3 4 2 1 6 3 2

После сортировки :

2 3 4 5 6 3 2 1

# Задачи

---

**«В»:** Напишите программу, которая сортирует массив и находит количество различных чисел в нем. Используйте алгоритм быстрой сортировки.

**Пример:**

**Массив :**

5 3 4 2 1 6 3 2 4

**После сортировки:**

1 2 2 3 3 4 4 5 6

**Различных чисел: 5**

# Задачи

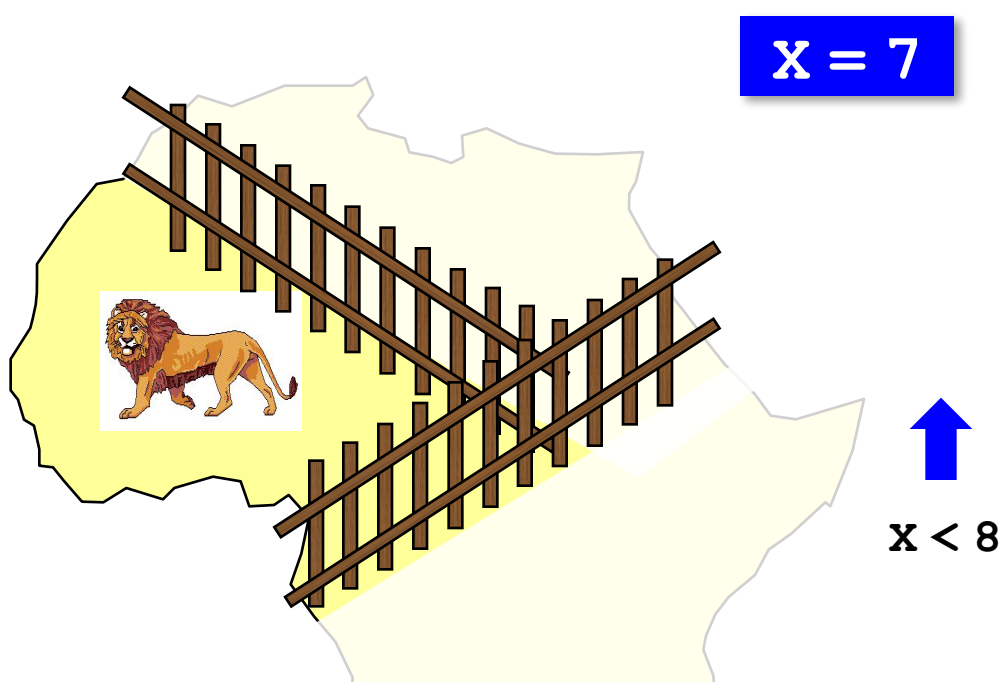
---

- «С»: Напишите программу, которая сравнивает число перестановок элементов при использовании сортировки «пузырьком», методом выбора и алгоритма быстрой сортировки. Проверьте ее на разных массивах, содержащих 1000 случайных элементов, вычислите среднее число перестановок для каждого метода.
- «D»: Попробуйте построить массив из 10 элементов, на котором алгоритм быстрой сортировки с выбором среднего элемента показывает худшую эффективность (наибольшее число перестановок). Сравните это количество перестановок с эффективностью метода пузырька (для того же массива).

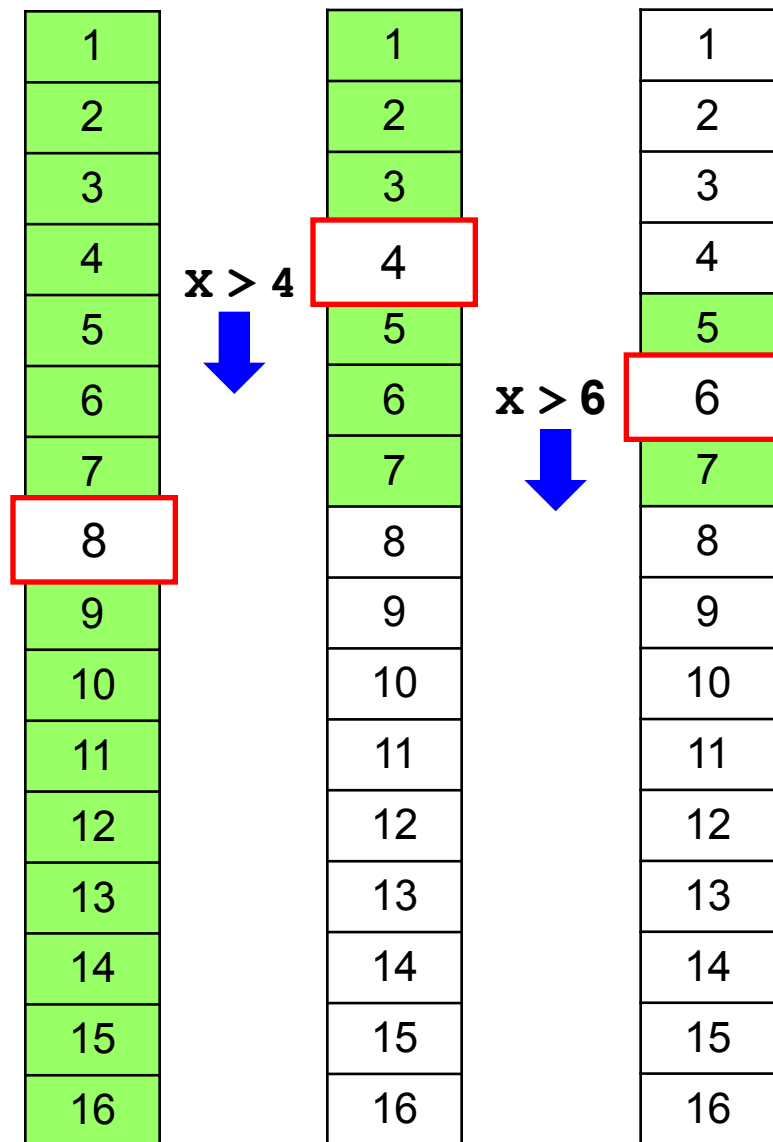
# Программирование на языке Python

## § 65. Двоичный поиск

# Двоичный поиск

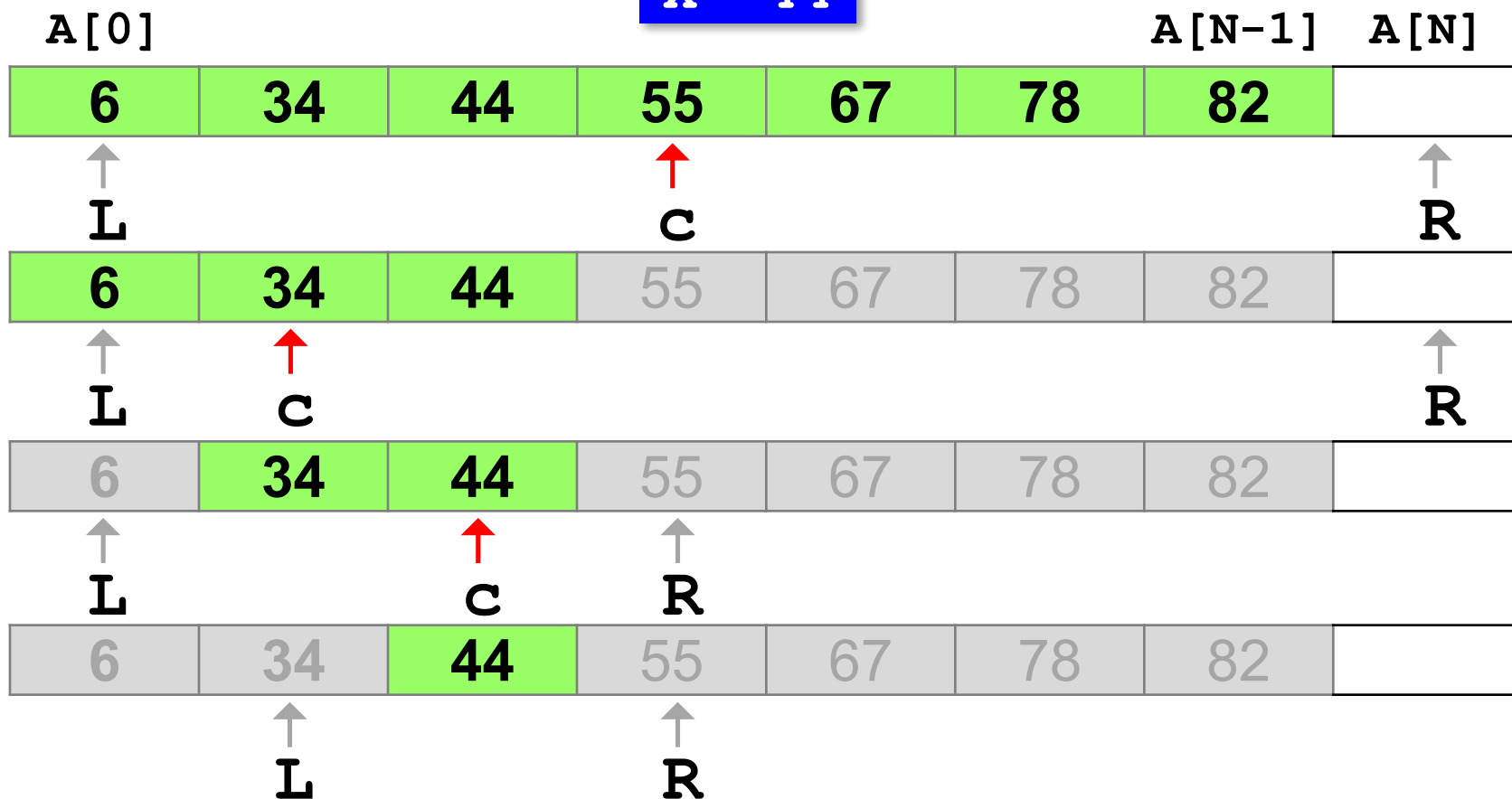


1. Выбрать средний элемент  $A[s]$  и сравнить с  $X$ .
2. Если  $X == A[s]$ , то нашли (**стоп**).
3. Если  $X < A[s]$ , искать дальше в первой половине.
4. Если  $X > A[s]$ , искать дальше во второй половине.



# Двоичный поиск

**X = 44**



**L = R - 1 : поиск завершен!**

# ДВОИЧНЫЙ ПОИСК

```
L = 0; R = N      # начальный отрезок
while L < R - 1:
    c = (L + R) // 2 # нашли середину
    if X < A[c]:    # сжатие отрезка
        R = c
    else: L = c
if A[L] == X:
    print ( "A[" , L, "]" = " , X, sep = "" )
else:
    print ( "Не нашли!" )
```



# Двоичный поиск

## Число сравнений:

N	линейный поиск	двоичный поиск
2	2	2
16	16	5
1024	1024	11
1048576	1048576	21



▪ скорость выше, чем при линейном поиске



▪ нужна предварительная сортировка



Когда нужно применять?

# Задачи

---

**«А»:** Заполнить массив случайными числами и отсортировать его. Ввести число  $X$ . Используя двоичный поиск, определить, есть ли в массиве число, равное  $X$ . Подсчитать количество сравнений.

**Пример:**

**Массив :**

1 4 7 3 9 2 4 5 2

**После сортировки:**

1 2 2 3 4 4 5 7 9

**Введите число  $X$ :**

2

**Число 2 найдено.**

**Количество сравнений: 2**

# Задачи

---

«В»: Заполнить массив случайными числами и отсортировать его. Ввести число  $X$ . Используя двоичный поиск, определить, сколько чисел, равных  $X$ , находится в массиве.

**Пример:**

**Массив:**

1 4 7 3 9 2 4 5 2

После сортировки:

1 2 2 3 4 4 5 7 9

Введите число  $X$ :

4

Число 4 встречается 2 раз (а) .

**Пример:**

**Массив:**

1 4 7 3 9 2 4 5 2

После сортировки:

1 2 2 3 4 4 5 7 9

Введите число  $X$ :

14

Число 14 не встречается.

# Задачи

---

«С»: Заполнить массив случайными числами и ввести число и отсортировать его. Ввести число X. Используя двоичный поиск, определить, есть ли в массиве число, равное X. Если такого числа нет, вывести число, ближайшее к X.

**Пример:**

Массив:

1 4 7 3 9 2 4 5 2

После сортировки:

1 2 2 3 4 4 5 12 19

Введите число X:

12

Число 12 найдено.

**Пример:**

Массив:

1 4 7 3 9 2 4 5 2

После сортировки:

1 2 2 3 4 4 5 12 19

Введите число X:

11

Число 11 не найдено. Ближайшее число 12.

# Программирование на языке Python

## § 66. Символьные строки

# Символьные строки

Начальное значение:

```
s = "Привет!"
```



Строка – это последовательность символов!

Вывод на экран:

```
print ( s )
```

```
print ( s[5] )
```

```
print ( s[-2] )
```

0	1	2	3	4	5	6
П	р	и	в	е	т	!
s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]

s[len(s)-2]

Длина строки:

```
n = len ( s )
```

# Символьные строки

---

Ввод с клавиатуры:

```
s = input ( "Введите имя: " )
```

Изменение строки:

```
s[4] = "a"
```



Строка – это неизменяемый объект!

... но можно составить новую строку:

```
s1 = s + "a"
```

# Символьные строки

Задача: заменить в строке все буквы "а" на буквы "б".

**!** Строка – это неизменяемый объект!

```
s = input ( "Введите строку: " )
s1 = ""      # строка-результат
for c in s:
    if c == "а":
        c = "б"
    s1 = s1 + c
print ( s1 )
```

перебрать все  
символы в строке

добавить символ к  
строке-результату



# Задачи

---

**«А»:** Ввести с клавиатуры символьную строку и заменить в ней все буквы «а» на «б» и все буквы «б» на «а» (заглавные на заглавные, строчные на строчные).

**Пример:**

**Введите строку:**

**ааббААББссСС**

**Результат:**

**ббааББААссСС**

# Задачи

---

**«В»:** Ввести с клавиатуры символьную строку и определить, сколько в ней слов. Словом считается последовательности непробельных символов, отделенная с двух сторон пробелами (или стоящая с краю строки). Слова могут быть разделены несколькими пробелами, в начале и в конце строки тоже могут быть пробелы.

## Пример:

Введите строку:

**Вася пошел гулять**

Найдено слов: 3

# Задачи

---

**«С»:** Ввести с клавиатуры символьную строку и найдите самое длинное слово и его длину. Словом считается последовательности непробельных символов, отделенная с двух сторон пробелами (или стоящая с краю строки). Слова могут быть разделены несколькими пробелами, в начале и в конце строки тоже могут быть пробелы.

## Пример:

Введите строку:

**Вася пошел гулять**

Самое длинное слово: **гулять**, длина **6**

# Сравнение строк

---

```
print( "Введите 2 строки: " )
s1 = input()
s2 = input()

if s1 == s2:
    print( s1, "=", s2 )
elif s1 < s2:
    print( s1, "<", s2 )
else:
    print( s1, ">", s2 )
```

# Сравнение строк

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

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

5STEAM < STEAM < Steam < steam

steam < ПАР < Пар < пАр < пар < парк

# Операции со строками

Объединение (конкатенация) :

```
s1 = "Привет"
```

```
s2 = "Вася"
```

```
s = s1 + ", " + s2 + "!"
```

"Привет, Вася!"

Срезы:

```
s = "0123456789"
```

```
s1 = s[3:8] # "34567"
```

ЭТОТ СИМВОЛ НЕ  
ВХОДИТ!

0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9

# Операции со строками

---

## Срезы:

```
s = "0123456789"  
s1 = s[:8] # "01234567"
```

от начала строки

```
s = "0123456789"  
s1 = s[3:] # "3456789"
```

до конца строки

```
s1 = s[::-1] # "9876543210"
```

реверс строки

# Операции со строками

## Срезы с отрицательными индексами:

```
s = "0123456789"  
s1 = s[: -2]           # "01234567"
```

N-2

```
s = "0123456789"  
s1 = s[-6: -2]        # "4567"
```

N-6

N-2



# Удаление и вставка СИМВОЛОВ



Строка – это неизменяемый объект!

## Удаление:

```
s = "0123456789"
s1 = s[:3] + s[9:] # "0129"
      "012"      "9"
```

## Вставка:

```
s = "0123456789"
s1 = s[:3] + "ABC" + s[3:]
      "012ABC3456789"
```

# Стандартные функции

---

## Верхний/нижний регистр:

```
s = "aAbBcC"  
s1 = s.upper()    # "AABVCC"  
s2 = s.lower()    # "aabbcc"
```

## Проверка на цифры:

```
s = "abc"  
print ( s.isdigit() )    # False  
s1 = "123"  
print ( s1.isdigit() )    # True
```

... и много других.

# Поиск в строках

```
s = "Здесь был Вася."  
n = s.find ( "с" )      # n = 3  
if n >= 0:  
    print ( "Номер символа", n )  
else:  
    print ( "Символ не найден." )
```



Находит первое слева вхождение подстроки!

Поиск с конца строки:

```
s = "Здесь был Вася."  
n = s.rfind ( "с" )     # n = 12
```

# Пример обработки строк

**Задача:** Ввести имя, отчество и фамилию. Преобразовать их к формату «фамилия-инициалы».

## Пример:

Введите имя, отчество и фамилию:

**Василий Алибабаевич Хрюндиков**

Результат:

**Хрюндиков В.А.**

Алибабаевич Хрюндиков

## Алгоритм:

- найти первый пробел и выделить имя
- удалить имя с пробелом из основной строки
- найти первый пробел и выделить отчество
- удалить отчество с пробелом из основной строки
- «сцепить» фамилию, первые буквы имени и фамилии, точки, пробелы...

Хрюндиков

Хрюндиков В.А.

# Пример обработки строк

```
print ( "Введите имя, отчество и фамилию:" )
s = input ()
n = s.find ( " " )
name = s[:n]      # вырезать имя
s = s[n+1:]
n = s.find ( " " )
name2 = s[:n]     # вырезать отчество
s = s[n+1:]      # осталась фамилия
s = s + " " + name[0] + "." + name2[0] + "."
print ( s )
```

# Пример обработки строк

---

## Решение в стиле Python:

```
print ( "Введите имя, отчество и фамилию:" )
s = input ()
fio = s.split ()
s = fio[2] + " " + fio[0][0] + "." + fio[1][0] + "."
print ( s )
```

Василий	Алибабаевич	Хрюндиков
fio[0]	fio[1]	fio[2]

# Задачи

---

**«А»:** Ввести с клавиатуры в одну строку фамилию, имя и отчество, разделив их пробелом. Вывести фамилию и инициалы.

## Пример:

**Введите фамилию, имя и отчество:**

**Иванов Петр Семёнович**

**П.С. Иванов**

# Задачи

---

«В»: Ввести адрес файла и «разобрать» его на части, разделенные знаком " / ". Каждую часть вывести в отдельной строке.

## Пример:

Введите адрес файла:

**C: /фото/2013/Поход/vasya.jpg**

C:

фото

2013

Поход

vasya.jpg



# Задачи

---

**«С»:** Напишите программу, которая заменяет во всей строке одну последовательность символов на другую.

**Пример:**

Введите строку:

`(X > 0) and (Y < X) and (Z > Y) and (Z <> 5)`

Что меняем: `and`

Чем заменить: `&`

Результат

`(X > 0) & (Y < X) & (Z > Y) & (Z <> 5)`

# Преобразования «строка» – «число»

## Из строки в число:

```
s = "123"
N = int ( s )          # N = 123
s = "123.456"
X = float ( s )       # X = 123.456
```

## Из числа в строку:

```
N = 123
s = str ( N )         # s = "123"
s = "{:5d}".format(N) # s = " 123"

X = 123.456
s = str ( X )        # s = "123.456"
s = "{:7.2f}".format(X) # s = " 123.46"
s = "{:10.2e}".format(X) # s = " 1.23e+02"
```

# Задачи

---

**«А»:** Напишите программу, которая вычисляет сумму трех чисел, введенную в форме символьной строки. Все числа целые.

**Пример:**

**Введите выражение :**

**12+3+45**

**Ответ: 60**

**«В»:** Напишите программу, которая вычисляет выражение, состоящее из трех чисел и двух знаков (допускаются только знаки «+» или «-»). Выражение вводится как символьная строка, все числа целые.

**Пример:**

**Введите выражение :**

**12-3+45**

**Ответ: 54**

# Задачи

---

**«С»:** Напишите программу, которая вычисляет выражение, состоящее из трех чисел и двух знаков (допускаются знаки «+», «-», «\*» и «/»). Выражение вводится как символьная строка, все числа целые. Операция «/» выполняется как целочисленное деление.

**Пример:**

**Введите выражение :**

**12\*3+45**

**Ответ: 81**

# Задачи

---

«D»: Напишите программу, которая вычисляет выражение, состоящее из трех чисел и двух знаков (допускаются знаки «+», «-», «\*» и «/») **и круглых скобок**. Выражение вводится как символьная строка, все числа целые. Операция «/» выполняется как целочисленное деление (`div`).

**Пример:**

**Введите выражение :**

**2 \* (3 + 45) + 4**

**Ответ: 100**

# Строки в процедурах и функциях

Задача: построить функцию, которая возвращает первое слово в предложении.

```
def firstWord( s ):  
    p = s.find( ' ' )  
    return s[:p]  
    return s  
else:  
    return s[:p]
```



Что плохо?

Однажды весной, в час...

Однажды весной, в час...

Однажды

```
word = firstWord( s )  
print( word )
```

Однажды

# Замена подстроки

**Задача:** построить функцию, которая заменяет в строке `s` все вхождения слова-образца `wOld` на слово-замену `wNew`.

пока слово `wOld` есть в строке `s`  
удалить слово `wOld` из строки  
вставить на это место слово `wNew`

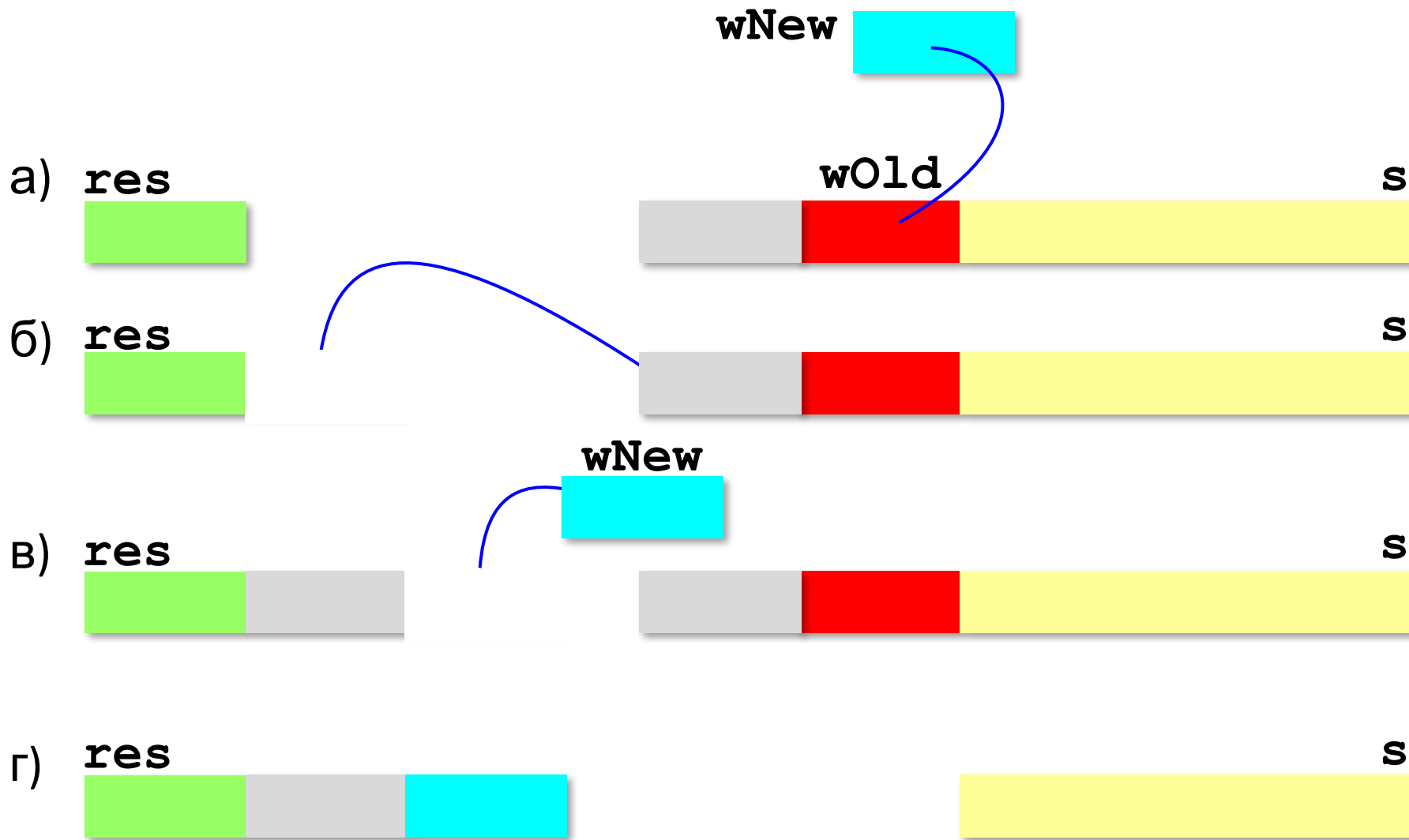
 Что плохо?

`wOld`: "12"

`wNew`: "A12B"

**зацикливание**

# Замена всех экземпляров подстроки





## Замена всех экземпляров подстроки

```
s = "12.12.12"  
s = replaceAll ( s, "12", "A12B" )  
print ( s )
```

рабочая строка `s`

"12.12.12"

результат `res`

" "

# Замена всех экземпляров подстроки

```
def replaceAll ( s, wOld, wNew ) :  
    lenOld = len (wOld)  
    res = ""  
    while len(s) > 0 :  
        p = s.find ( wOld )  
        if p < 0 :  
            res = res + s  
            return  
        if p > 0 : res = res + s [:p]  
        res = res + wNew  
        if p + lenOld >= len (s) :  
            s = ""  
        else :  
            s = s [p + lenOld : ]  
    return res
```

искать образец

если не нашли

взять начало  
перед образцом

добавить  
слово-замену

строка кончилась

взять «хвост»

# Замена всех экземпляров подстроки

---

Встроенная функция:

```
s = "12.12.12"  
s = s.replace( "12", "A12B" )  
print ( s )
```

# Задачи

---

**«А»:** Напишите функцию, которая отсекает всю часть строки после первого слова.

**Пример:**

**Введите строку:** Однажды в студёную зимнюю пору...

**Первое слово:** Однажды

# Задачи

---

«В»: Напишите функцию, которая заменяет расширение файла на заданное новое расширение.

**Пример:**

Введите имя файла: `qq`

Введите новое расширение: `tmp`

Результат: `qq.tmp`

**Пример:**

Введите имя файла: `qq.exe`

Введите новое расширение: `tmp`

Результат: `qq.tmp`

**Пример:**

Введите имя файла: `qq.work.xml`

Введите новое расширение: `tmp`

Результат: `qq.work.tmp`

# Задачи

---

**«С»:** Напишите функцию, которая заменяет во всей строке все римские числа на соответствующие десятичные числа.

**Пример:**

Введите строку:

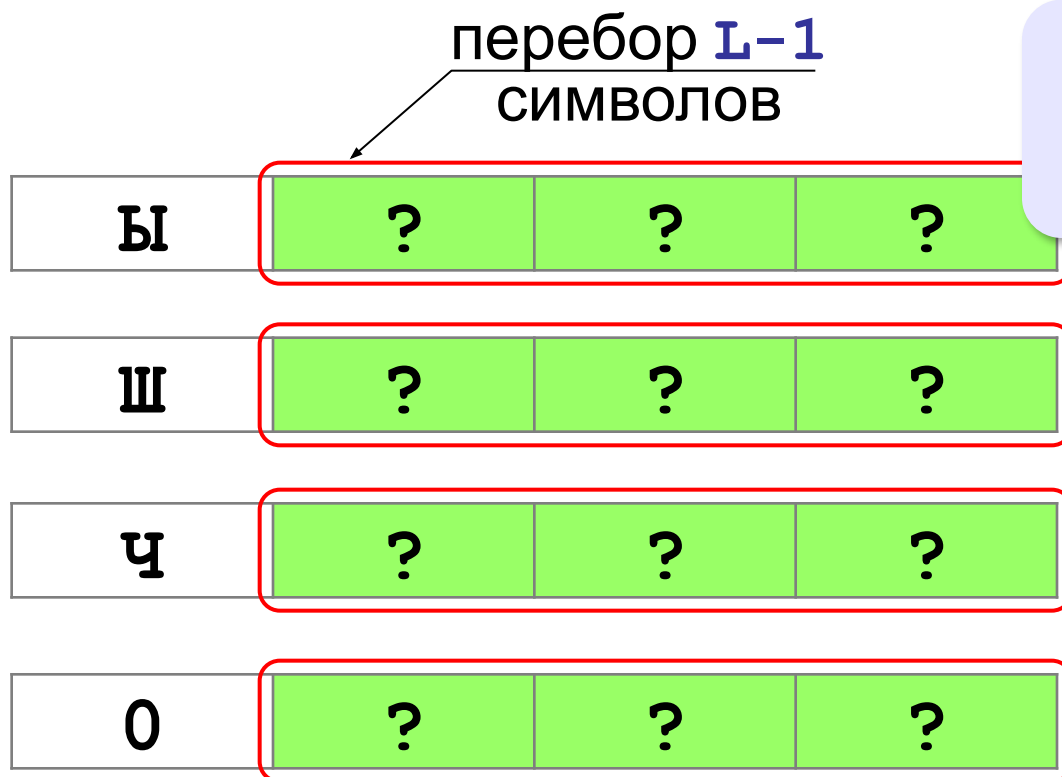
**В MMXIII году в школе CXXIII состоялся очередной выпуск XI классов.**

Результат:

**В 2013 году в школе 123 состоялся очередной выпуск 11 классов.**

# 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  $L$  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 ( "ЬШЧО" , "" , 3 )
```

# Рекурсивный перебор + счётчик

```
count = 0
```

```
def TumbaWords ( A, w, L ):
```

```
    global count
```

будем менять глобальную переменную

```
    if len(w) == L:
```

```
        print ( w )
```

```
        count += 1
```

увеличение счётчика

```
    return
```

```
    for c in A:
```

```
        TumbaWords ( A, w+c, L )
```

```
TumbaWords ( "ЬШЧО", "", 3 )
```

```
print( count )
```

# Рекурсивный перебор + условие

```
count = 0
```

```
def TumbaWords ( A, w, L ) :  
    global count  
    if len(w) == L :  
        if not "OO" in w :  
            print ( w )  
            count += 1  
        return  
    for c in A :  
        TumbaWords ( A, w+c, L )
```

условие  
отбора

```
TumbaWords ( "ЬШЧО" , "" , 3 )  
print( count )
```

# Рекурсивный перебор + условие (функция)

```
def valid ( s ) :  
    if not "OO" in w :  
        return True  
    else :  
        return False
```

```
return not "OO" in w
```

```
def TumbaWords ( A, w, L ) :  
    global count  
    if len(w) == L :  
        if valid(w) :  
            print ( w )  
            count += 1  
    return  
    for c in A :  
        TumbaWords ( A, w + c, L )
```

условие  
отбора

# Задачи

---

- «А»:** 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.
- «В»:** 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.

# Задачи

---

**«С»:** 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, не обязательно стоящие рядом.

Программа не должна строить другие слова, не соответствующие условию.

# 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

---

**«В»:** 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**

# Задачи

---

**«С»:** Вводится несколько строк (не более 20), в которых сначала записаны инициалы и фамилии работников фирмы. Ввод заканчивается пустой строкой. Отсортировать строки в алфавитном порядке по фамилии.

## Пример:

Введите ФИО:

**А.Г. Урядников**

**Б.В. Тепловозов**

**В.Д. Арбузов**

Список в алфавитном порядке:

**В.Д. Арбузов**

**Б.В. Тепловозов**

**А.Г. Урядников**

# Программирование на языке Python

## § 67. Матрицы

# Что такое матрица?

	○	×
	○	×
○	×	

нет знака

A

НОЛИК

крестик

строка 1,  
столбец 2

A[1][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]]
```

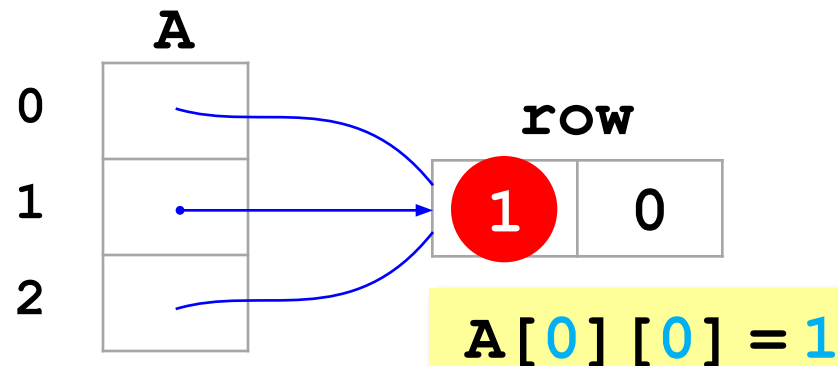
**!** Нумерация элементов с нуля!

A[0][0]	A[0][1]	A[0][2]
A[1][0]	A[1][1]	A[1][2]
A[2][0]	A[2][1]	A[2][2]

# Создание матриц

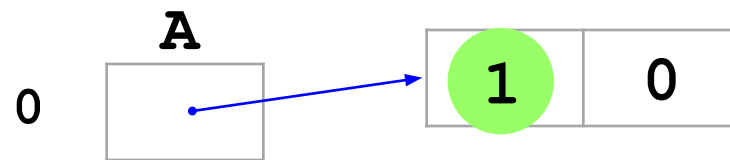
## Нулевая матрица:

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



## а правильно так:

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



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

# Вывод матриц

```
print ( A )
```

```
[[1, 12, 3], [4, 5, 146], [7, 118, 99]]
```

```
def printMatrix ( A ):
```

```
    for row in A:
```

```
        for x in row:
```

```
            print ( "{:4d}".format(x) , end = "" )
```

```
        print ()
```

```
1   12   3
4    5 146
7 118  99
```



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

---

**«А»:** 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 А:**

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

---

«В»: 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 А:**

```
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
```

# Задачи

«С»: Заполните матрицу, содержащую  $N$  строк и  $M$  столбцов, натуральными числами по спирали и змейкой, как на рисунках:

а)

1	2	3	4
10	11	12	5
9	8	7	6

б)

1	3	4	9
2	5	8	10
6	7	11	12

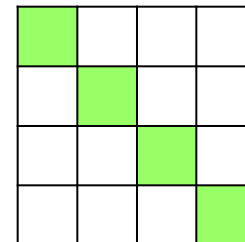
в)

1	6	7	12
2	5	8	11
3	4	9	10

# Перебор элементов матрицы

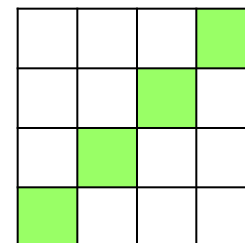
## Главная диагональ:

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



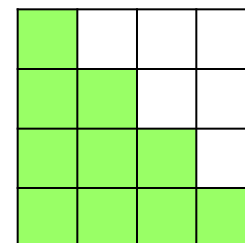
## Побочная диагональ:

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



## Главная диагональ и под ней:

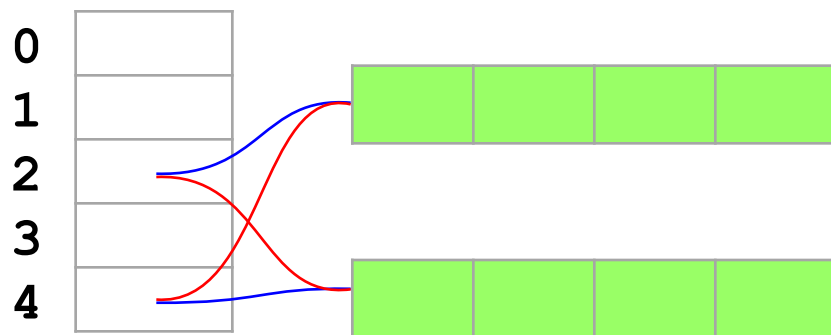
```
for i in range(N):  
    for j in range(i+1):  
        # работаем с A[i][j]
```



# 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 А:**

12 14 67 45

32 87 45 63

69 45 14 30

40 12 35 65

**Результат:**

12 0 0 0

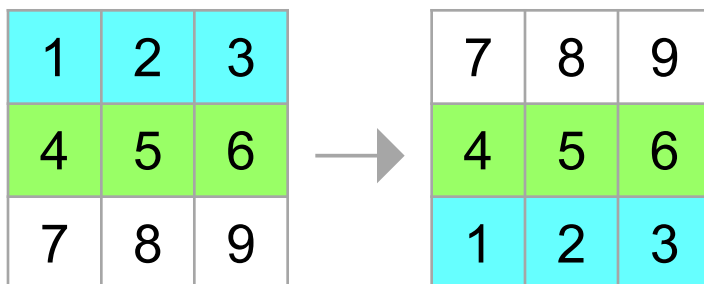
32 87 0 0

69 45 14 0

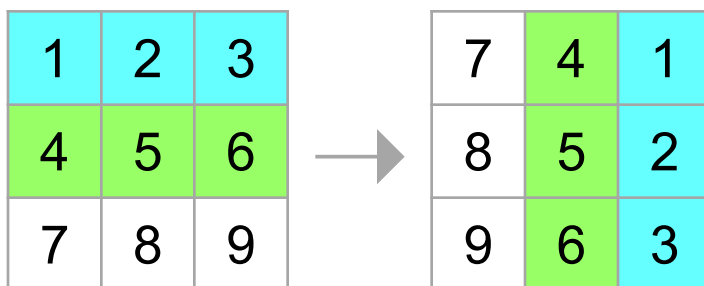
40 12 35 65

# Задачи

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



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



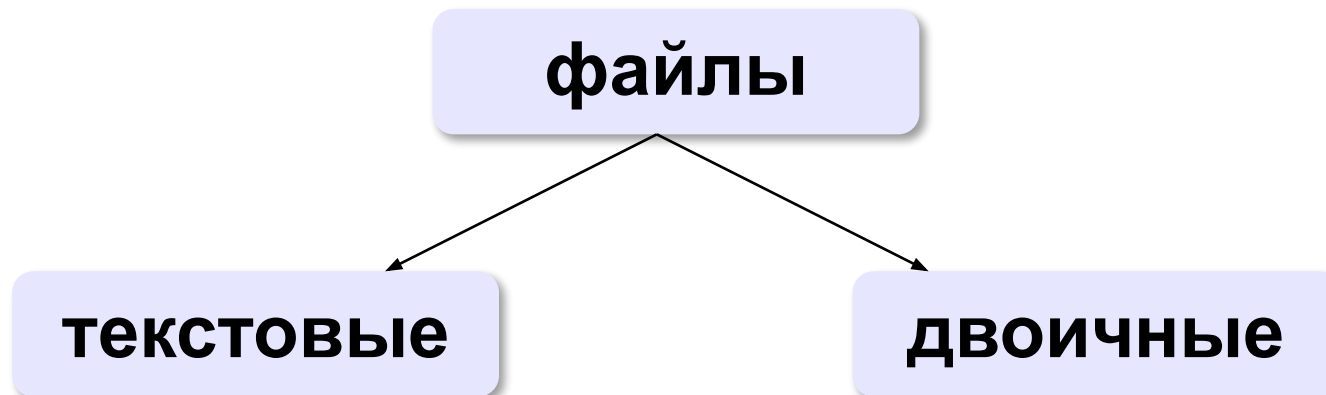


# Программирование на языке Python

## § 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

---

- «А»: Напишите программу, которая находит среднее арифметическое всех чисел, записанных в файле в столбик, и выводит результат в другой файл.
  
- «В»: 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.
  
- «С»: 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

---

- «А»: В файле в столбик записаны числа. Отсортировать их по возрастанию последней цифры и записать в другой файл.
- «В»: В файле в столбик записаны числа. Отсортировать их по возрастанию суммы цифр и записать в другой файл. Используйте функцию, которая вычисляет сумму цифр числа.
- «С»: 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 строке 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

---

**«А»:** 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.

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

**П. Иванов**

**И. Петров**

...

# Задачи

---

**«С»:** 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 данные учеников, которые получили больше 80 баллов. Список должен быть отсортирован по убыванию балла. Формат выходных данных:

**П. Иванов 98**

**И. Петров 96**

**...**

# Конец фильма

---

**ПОЛЯКОВ Константин Юрьевич**

д.т.н., учитель информатики

ГБОУ СОШ № 163, г. Санкт-Петербург

[kpolyakov@mail.ru](mailto:kpolyakov@mail.ru)

**ЕРЕМИН Евгений Александрович**

к.ф.-м.н., доцент кафедры мультимедийной

дидактики и ИТО ПГГПУ, г. Пермь

[eremin@pspu.ac.ru](mailto:eremin@pspu.ac.ru)

# Источники иллюстраций

---

1. [www.mcdonalds.com](http://www.mcdonalds.com)
2. иллюстрации художников издательства «Бином»
3. авторские материалы