# Программирование на языке Python

## § 62. Массивы

# Что такое массив?

Как ввести 10000 переменных?

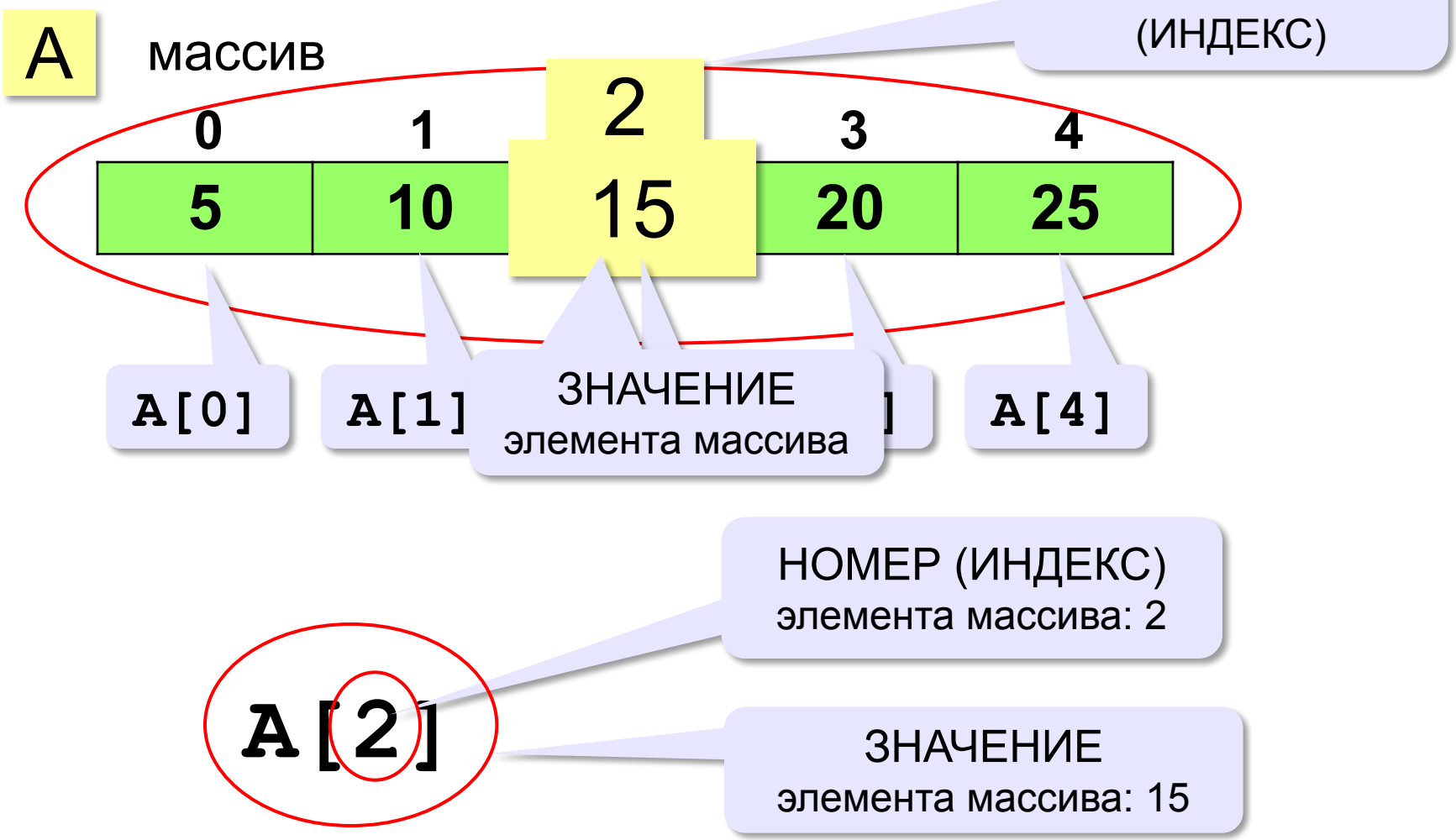
**Массив** – это группа переменных одного типа, расположенных в памяти рядом (в соседних ячейках) и имеющих общее имя. Каждая ячейка в массиве имеет уникальный номер (индекс).

**Надо**:

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

# Что такое массив?

Массив = **таблица**!



# Массивы в Python: списки

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

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

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

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


Что будет?

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

```
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

# 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]
```

# 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 = ""` — не разделять элементы

`end = ""` — 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)
```

# 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], ..., A[N-1]

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

# Подсчёт нужных элементов

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

Как решать?

Python:
```
180 < x < 190
```

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

# Перебор элементов

**Сумма:**

```
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

**«A»:** 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
```

**«B»:** Заполните массив случайными числами в интервале [0,100] и подсчитайте отдельно среднее значение всех элементов, которые <50, и среднее значение всех элементов, которые ≥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

**«B»:** 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

**«A»:** 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
```

**«B»:** 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
```

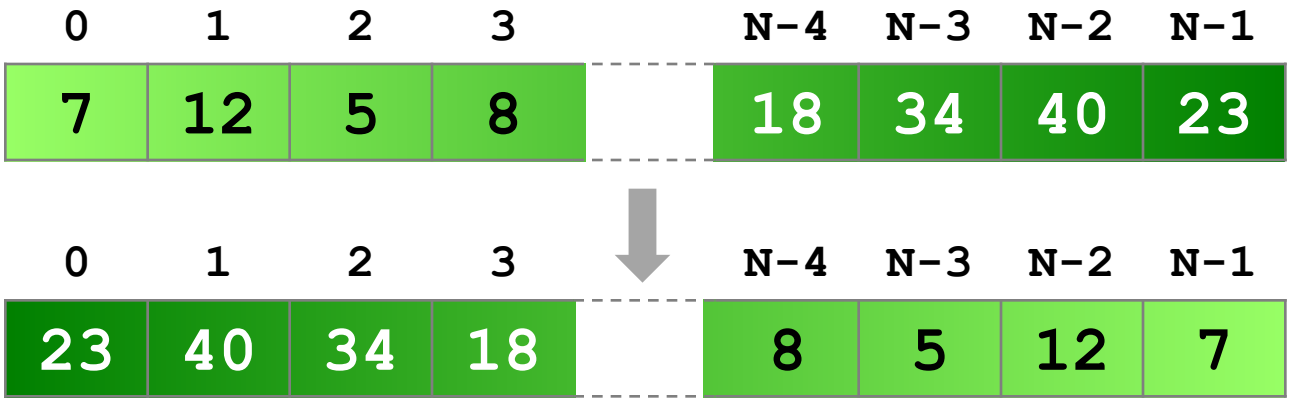
# Задачи

**«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

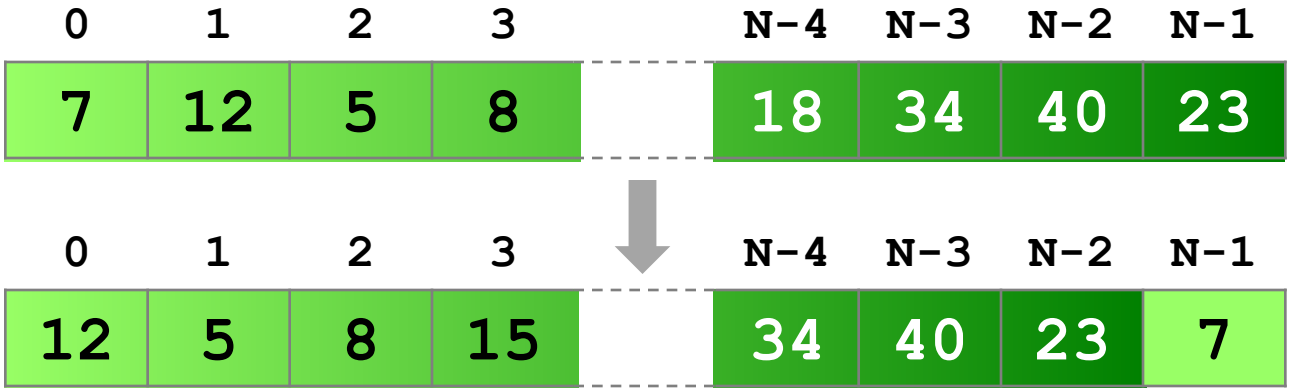
```
for i in range(N//2):
    c=A[i]
    A[i]=A[N-1-i]
    A[N-1-i]=c
```

## 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



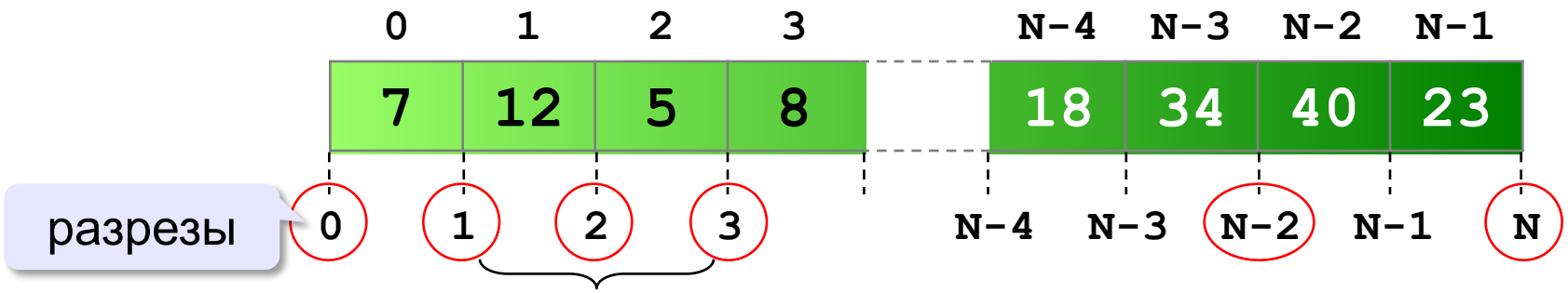

**«Простое» решение:**

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

? Почему не до **N**?

? Что плохо?

# Срезы в Python




```
A[1:3] → [12, 5]
A[2:3] → [5]
A[:3] → A[0:3] → [7, 12, 5]
```

с начала

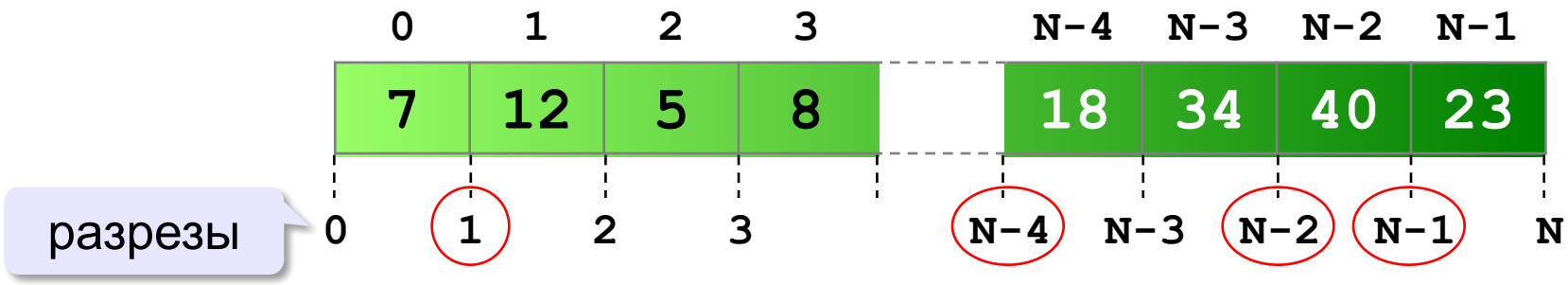
```
A[3:N-2] → [8,…,18,34]
A[3:] → A[3:N] → [8,…,18,34,40,23]
```

до конца

```
A[:] → [7,12,5,8,…,18,34,40,23]
```

копия массива

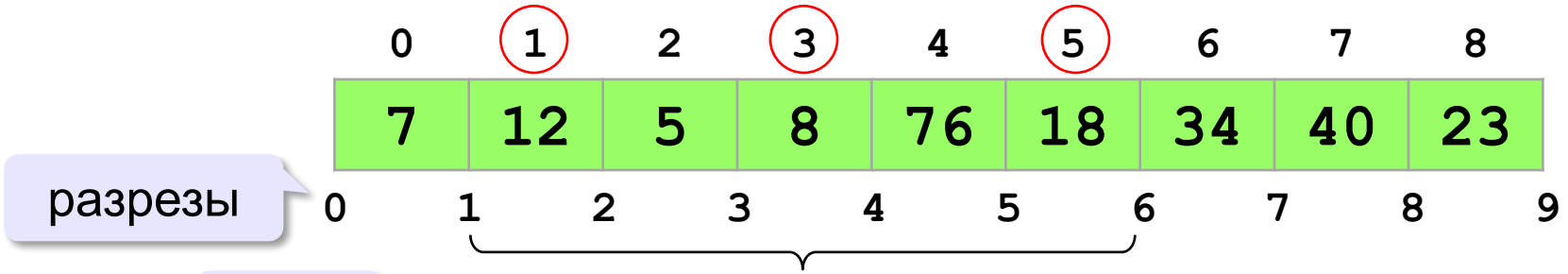
# Срезы в Python – отрицательные индексы




```
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




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

**«A»:** 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
```

**«B»:** 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

# Задачи

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

**Пример**:

```
Массив A:
-5 6 7 -4 -6 8 -8
Массив B:
-4 -6 -8
```

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

**Пример**:

```
Массив A:
12 13 85 96 47
Массив B:
13 47
```

# Задачи

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

**Пример**:

```
Массив A:
12 13 85 34 47
Массив B:
13 34
```