

Программирование на языке Python

Обработка массивов

Массивы (списки) в Python

Создание массива:

```
A = [1, 5, 0, -1, 12]
```

```
print(A[1])
```

A[0] ↑ A[2] ↑ A[4]

5

A[1] A[3]

```
print(2*A[0]+A[3])
```

1

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



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

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=" " )
```

пробел после
вывода очередного
числа

1 2 3 4 5

или так:

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

разбить список на
элементы

Заполнение случайными числами

```
from random import randint  
A = []  
for i in range(5):  
    A.append(randint(1, 6))  
print(A)
```

наращиваем с
каждым шагом

?

В чём отличие?

Или так:

```
from random import randint  
A = 5*[0]  
for i in range(5):  
    A[i] = randint(1, 6)  
print(A)
```

сначала выделили
память, потом
меняем

Подсчёт элементов

```
A = [1, 2, 3, 4, 5, 6, 7]
k = 0
for i in range(7):
    if A[i] > 3: k += 1
print(k)
```



Что выведет?

4

Кумир:

```
k := 0
нц для i от 1 до 7
    если A[i] > 3 то
        k := k + 1
    все
кц
вывод k
```

Паскаль:

```
k := 0;
for i:=1 to 7 do
    if A[i] > 3 then
        k = k + 1;
writeln(k);
```



Элементы массива нумеруются с 1!

Подсчёт элементов

```
A = [1, 21, 3, 46, 53, 6, 17]
```

```
k = 0
```

```
for i in range(7):
```

```
    if A[i] % 3 == 0: k += 1
```

```
print(k)
```



Что выведет?

3

Варианты условий:

```
if A[i] % 10 == 6: k += 1
```

2

```
if (A[i] % 10 == 6 and  
    A[i] % 3 == 0): k += 1
```

1

```
if (A[i] >= 10 and  
    A[i] < 100): k += 1
```

4

Суммирование элементов

```
A = [1, 21, 3, 46, 53, 6, 115]
```

```
s = 0
```

```
for i in range(7):
```

```
    if A[i] % 3 == 0: s += A[i]
```

```
print(s)
```



Что выведет?

30

Варианты условий:

```
if A[i] % 10 == 6: s += A[i]
```

52

```
if (A[i] % 10 == 6 and
```

```
    A[i] % 3 == 0): s += A[i]
```

6

```
if (A[i] >= 10 and
```

```
    A[i] < 100): s += A[i]
```

120

Задачи

- «3»: Напишите программу, которая находит в массиве количество элементов, делящихся на 5.
- «4»: Напишите программу, которая находит среднее арифметическое всех элементов массива, которые делятся на 3 и заканчиваются на 1.
- «5»: Напишите программу, которая находит среднее арифметическое всех элементов массива, двоичная запись которых содержит ровно 4 цифры.
- «6»: Напишите программу, которая находит элемент массива, двоичная запись которого содержит больше всего единиц.

Максимум

```
A = [1, 21, 3, 46, 53, 6, 117]
```

```
m = 0
```

МЕНЬШЕ ВСЕХ

```
for i in range(7):
```

```
    if A[i] > m: m = A[i]
```

```
print(m)
```



Что выведет?

117

Кумир:

```
m := 0
```

```
нц для i от 1 до 7
```

```
    если A[i] > m то
```

```
        m := A[i]
```

```
    все
```

```
кц
```

```
вывод m
```

Паскаль:

```
m := 0;
```

```
for i:=1 to 7 do
```

```
    if A[i] > m then
```

```
        m = A[i];
```

```
writeln(m);
```

Минимум

```
A = [1, 21, 3, 46, 53, 6, 117]
```

```
m = 999
```

больше всех

```
for i in range(7):
    if A[i] < m: m = A[i]
print(m)
```



Что выведет?

1

Кумир:

```
m := 999
нц для i от 1 до 7
    если A[i] < m то
        m := A[i]
    все
кц
вывод m
```

Паскаль:

```
m := 999;
for i:=1 to 7 do
    if A[i] < m then
        m = A[i];
writeln(m);
```

Если значения в массиве неизвестны...

```
A = [...как-то получили...]  
N = len(A) # длина массива  
m = A[0]  
for i in range(N):  
    if A[i] < m: m = A[i]  
print(m)
```



Что записать в m?



Как сэкономить один шаг цикла?

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

пропустить
A[0]

Python: `m = min(A)`

Задачи

«3»: Напишите программу, которая находит минимальный и максимальный из чётных элементов массива. Гарантируется, что все элементы массива находятся в диапазоне $[-100; 100]$ и среди них есть хотя бы один чётный элемент.

«4»: Напишите программу, которая находит минимальный и максимальный из элементов массива, заканчивающихся на "5". Если в массиве нет таких элементов, нужно вывести слово "нет".

Задачи

«5»: Напишите программу, которая находит минимальный из чётных элементов массива и его номер. Если в массиве нет таких элементов, нужно вывести слово "нет".

Пример:

Массив: [1, 12, 3, 4, 5, 18, 24]

Минимум: A[3] = 4

Пример:

Массив: [1, 13, 3, 19, 5, 71, 241]

Минимум: нет

Сортировка

Сортировка – это расстановка элементов массива в заданном порядке (возрастания, убывания, ...).

Было:

9 6 2 7 3 1 5 4 8 0

Стало:

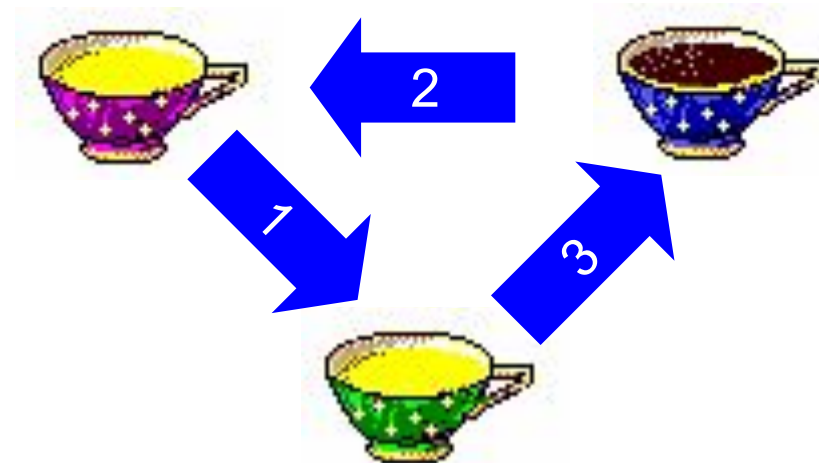
0 1 2 3 4 5 6 7 8 9



Основная операция –
перестановка элементов!

Перестановка элементов

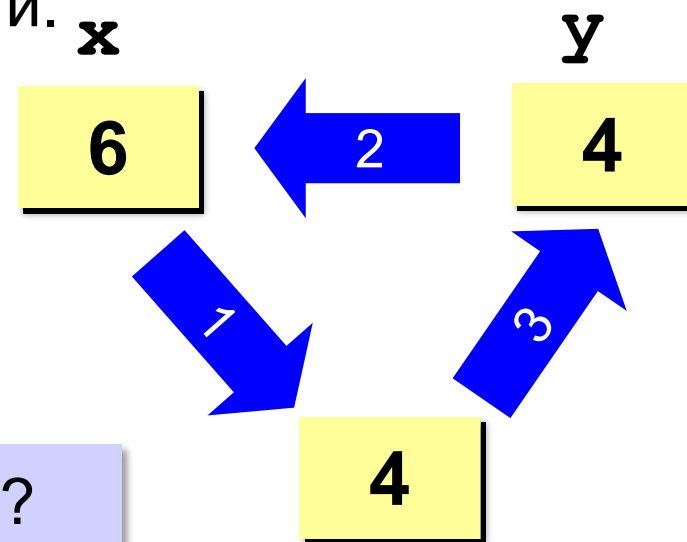
Задача: поменять местами содержимое двух чашек.



Задача: поменять местами содержимое двух ячеек памяти.

~~$x = y$
 $y = x$~~

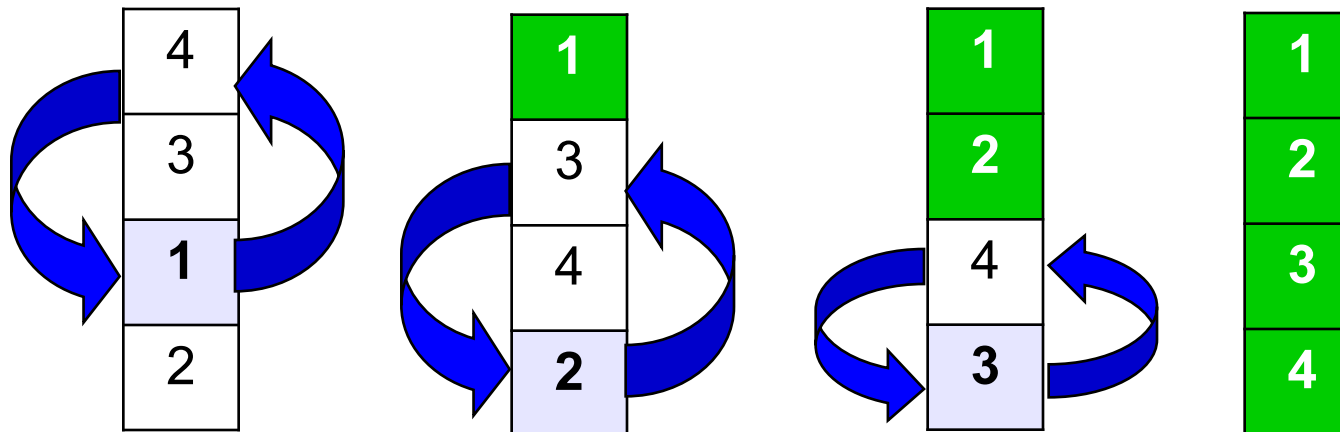
$c = x$
 $x = y$
 $y = c$



? Можно ли обойтись без c ?

Python: $x, y = y, x$

Метод выбора (минимального элемента)



Идея:

- найти минимальный элемент и поставить на первое место (поменять местами с $A[0]$)
- из оставшихся найти минимальный элемент и поставить на второе место (поменять местами с $A[1]$), и т.д.



Сколько раз сделать цикл?

$N-1$

Как найти номер минимального элемента?

```
A = [1, 21, 3, -46, 53, -6, 117]
N = len(A) # длина массива
m = A[0]   # считаем A[0] = min
nM = 0     # номер минимального
for i in range(N):
    if A[i] < m:
        m = A[i]
        nM = i # новый номер
print(nM)
```



Нельзя ли обойтись без переменной `m`?

`m = A[nM]`

Как найти номер минимального элемента?

```
A = [1, 21, 3, -46, 53, -6, 117]
N = len(A) # длина массива
nM = 0      # номер минимального
for i in range(N):
    if A[i] < A[nM]:
        nM = i # новый номер
print(nM)
```

часть
алгоритма
сортировки

Python:

```
m = min(A)
nM = A.index(m)
```

Сортировка выбором

```
A = [1, 21, 3, -46, 53, -6, 117]
N = len(A) # длина массива
for k in range(N-1):
    nM = k
    for i in range(k, N):
        if A[i] < A[nM]:
            nM = i
    A[k], A[nM] = A[nM], A[k]
print(A)
```

ПОИСК
МИНИМАЛЬНОГО

перестановка



Почему эта программа не работает?

искать минимальный,
начиная с номера k!

Python:

```
A.sort()
```

Задания

«3»: Заполнить массив из 10 элементов случайными числами в интервале [0..99] и отсортировать его по убыванию последней цифры.

Пример:

Исходный массив:

14 25 13 12 76 58 21 87 10 98

Результат:

98 58 87 76 25 14 13 12 21 10

Задания

«4»: Заполнить массив из 10 элементов случайными числами в интервале [0..99] и отсортировать его по возрастанию суммы цифр (*подсказка: их всего две*).

Пример:

Исходный массив:

14 25 13 12 76 58 21 87 10 98

Результат:

10 21 12 13 14 25 76 58 87 98

Задания

«5»: Заполнить массив из 10 элементов случайными числами в интервале $[0..100]$ и отсортировать первую половину по возрастанию, а вторую – по убыванию.

Пример:

Исходный массив:

14 25 13 30 76 | 58 32 11 41 97

Результат:

13 14 25 30 76 | 97 58 41 32 11