

Программирование на языке Python

Символьные строки

Массивы (списки)

Поиск в массиве

Программирование на языке Python

Символьные строки

Символьные строки

Начальное значение:

```
s = "Привет!"
```



Строка – это последовательность символов!

Вывод на экран:

```
print ( s )
```

Сложение:

```
s1 = "Привет"
```

```
s2 = "Вася"
```

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

"Привет, Вася!"

Умножение:

```
s = "Ау"
```

```
s5 = s*5
```

```
s5 = s + s + s + s + s
```

АУАУАУАУАУ



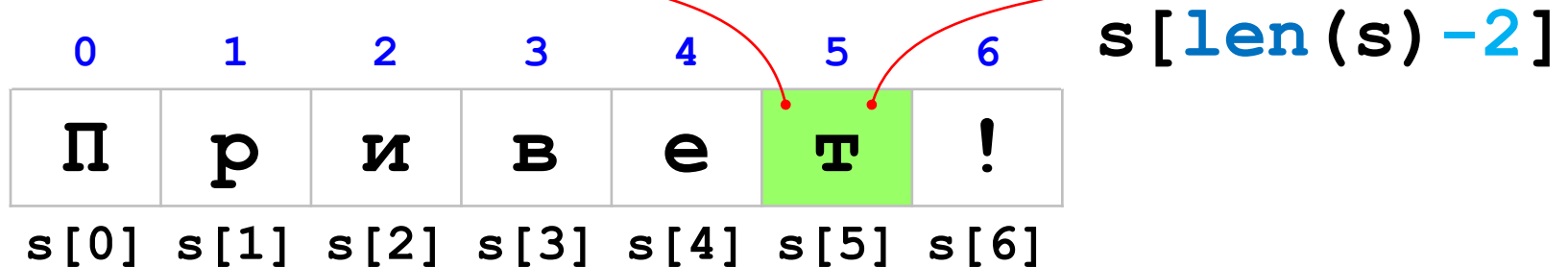
Что получим?

Символьные строки

Вывод символа на экран:

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

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



Длина строки:

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

Символьные строки

Ввод с клавиатуры:

```
s = input ( "Введите имя: " )
```

Изменение строки:запрещено!

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



Строка – это неизменяемый объект!

... НО МОЖНО СОСТАВИТЬ НОВУЮ СТРОКУ:

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

```
s = "информатика"  
print(s[-2]+s[3]+s[-4])
```

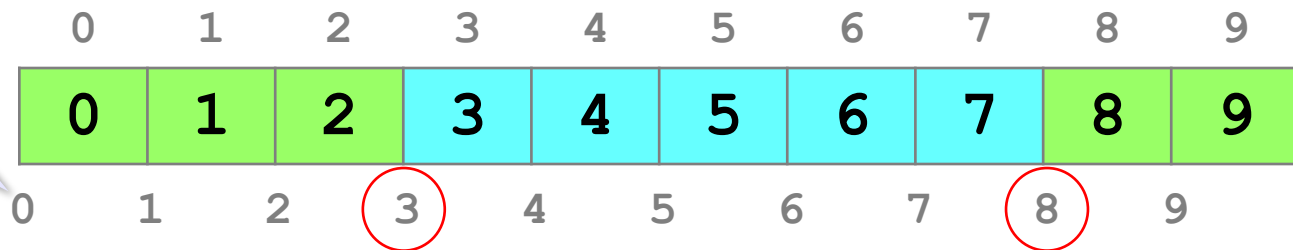
СОСТАВИТЬ «КОТ»

Срезы

```
s = "0123456789"
```

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

разрезы



Срезы строк

```
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"
```

$\text{len}(s) - 2$

```
s = "0123456789"
```

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

$\text{len}(s) - 6$

$\text{len}(s) - 2$

Операции со строками

Удаление:

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

Вставка:

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

Программирование на языке Python

Массивы (списки)

Что такое массив?



Как ввести 10000 переменных?

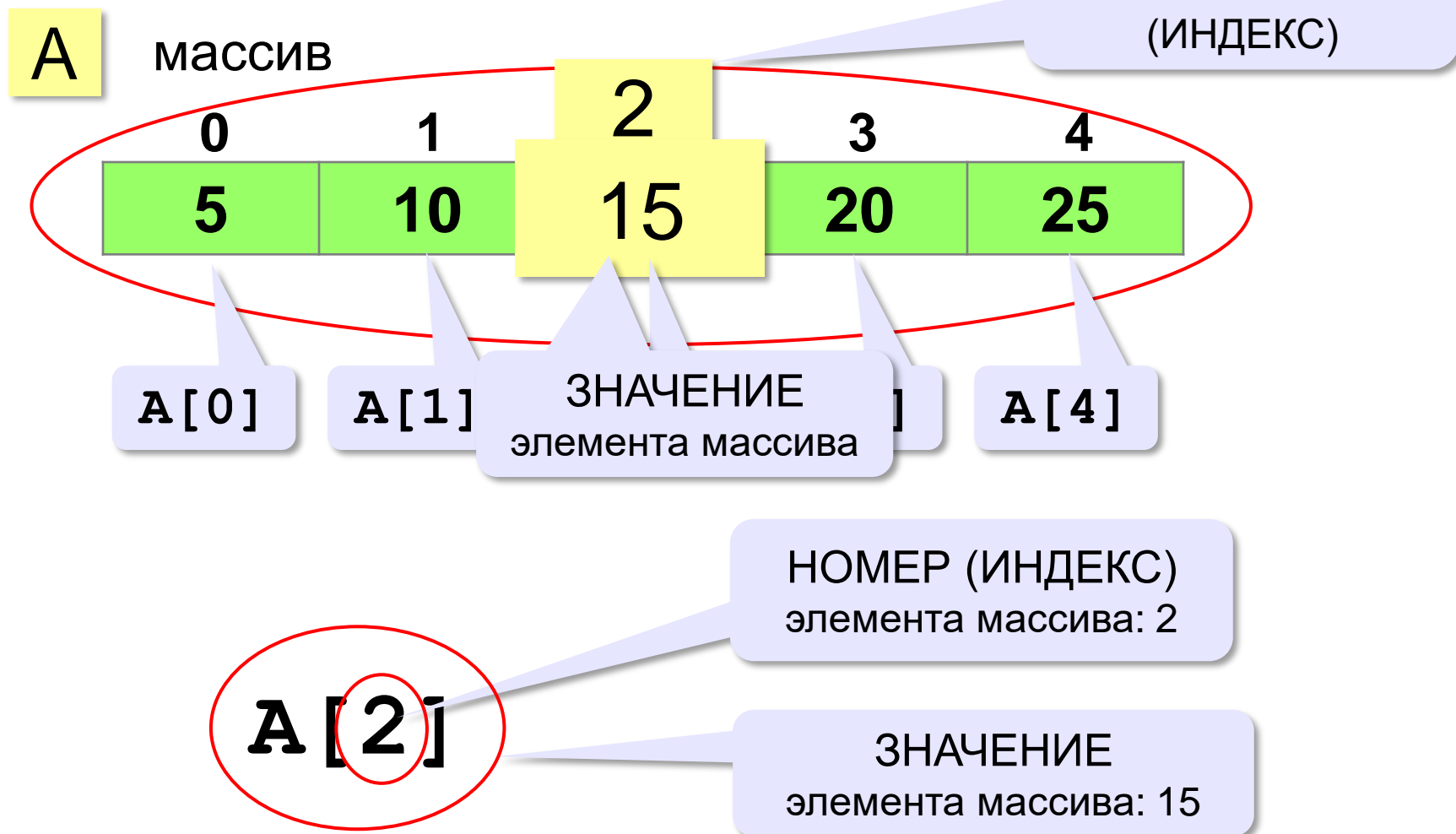
Массив – это группа переменных одного типа, расположенных в памяти рядом (в соседних ячейках) и имеющих общее имя. Каждая ячейка в массиве имеет уникальный номер (индекс).

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
```

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



Что будет?

Создание массива из N элементов:


```
N = 10
```

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

Заполнение массива

Целыми числами (начиная с 0!):

```
N = 10          # размер массива
A = [0]*N       # выделить память
for i in range(N):
    A[i] = i
```



В краткой форме:

```
N = 10          # размер массива
A = [ i for i in range(N) ]
```



Как заполнить, начиная с 1?



Как заполнить квадратами чисел?

Заполнение случайными числами

из библиотеки (модуля)
random

взять функцию randint

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

В краткой форме:

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

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

Ввод массива с клавиатуры

Создание массива:

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

Ввод по одному элементу в строке:

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

или кратко:

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

Ввод массива с клавиатуры

Ввод всех чисел 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:

```
A = [int(x) for x in input().split()]
```

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]
```

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 )
```

Что выведет программа?

```
A = [2, 3, 1, 4, 6, 5]
```

```
print( A[3] )           # 4
```

```
print( A[0]+2*A[5] )    # 12
```

```
A[1] = A[0] + A[5]      # 7  
print( 3*A[1]+A[4] )    # 27
```

```
A[2] = A[1]*A[4]        # 18  
print( 2*A[1]+A[2] )    # 24
```

```
for k in range(6):  
    A[k] += 2           # [4,5,3,6,8,7]  
print( 2*A[3]+3*A[4] ) # 36
```

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?

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

Перебор элементов

Задача. Найти сумму чётных элементов массива.

```
summa = 0
for x in A:
    if x % 2 == 0:
        summa += x
print ( summa )
```



Как определить, что элемент чётный?

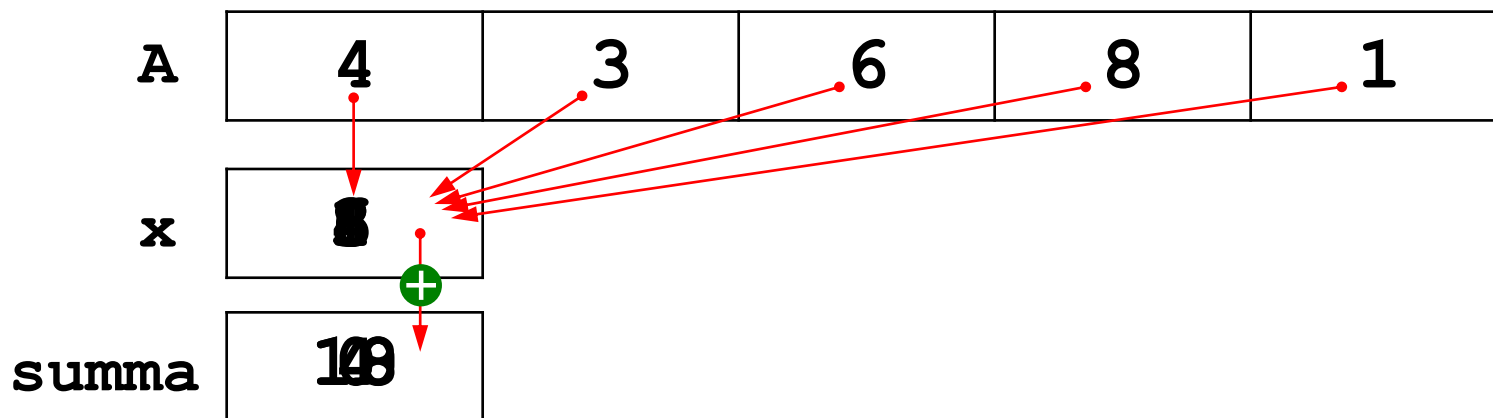
или так:

```
B = [x for x in A
      if x % 2 == 0]
print ( sum(B) )
```

сумма массива B

Как работает цикл?

```
summa = 0  
for x in A:  
    if x % 2 == 0:  
        summa += x
```



Среднее арифметическое

Задача. Найти среднее арифметическое элементов массива, которые оканчиваются на цифру 5.

```
count = 0
summa = 0
for x in A:
    if x % 10 == 5:
        count += 1
        summa += x
print ( summa/count )
```



Как определить, что оканчивается на 5?

среднее
арифметическое

или так:

```
B = [ x for x in A
      if x % 10 == 5 ]
print ( sum(B)/len(B) )
```

отбираем нужные

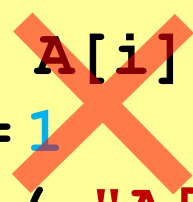
Программирование на языке Си

Поиск в массиве

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

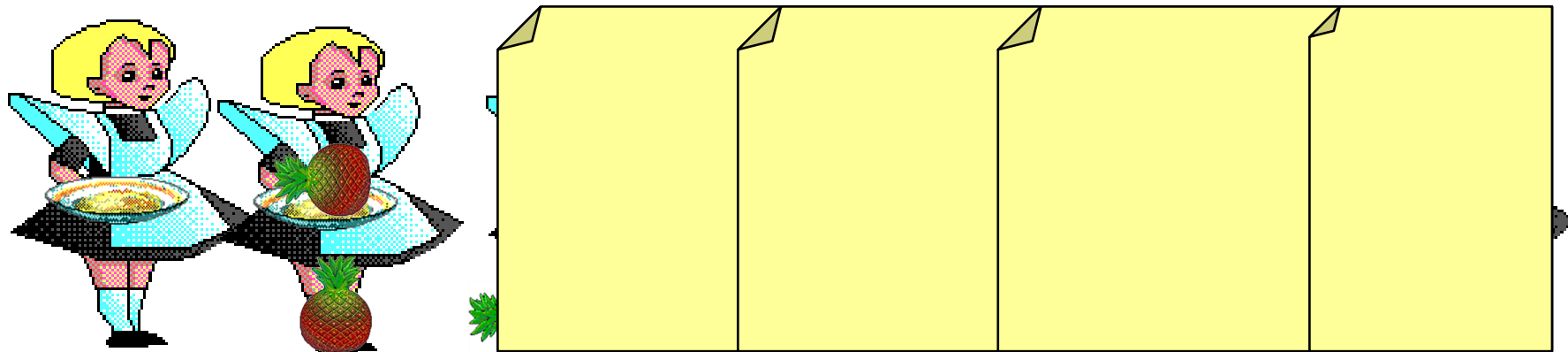
```
nX = -1
for i in range ( N ):
    if A[i] == X:
        nX = i
        break
if nX >= 0:
    print ( "A[" , nX, "]" = " , X, sep = " " )
else:
    print ( "Не нашли!" )
```

досрочный
выход из цикла

Максимальный элемент

Задача: найти в массиве максимальный элемент.

Алгоритм:



Решение:

- 1) считаем, что первый элемент – максимальный
- 2) просмотреть остальные элементы массива:
если очередной элемент $> M$,
то записать $A[i]$ в M
- 3) вывести значение M

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...)