ПРОГРАММИРОВАНИЕ PYTHON

Списки (list):

Часть 1

Одномерные массивы

(Лекция 9)

Одномерные массивы

1. Что такое массив?
2. Индексы
3. Обращение к элементу массива
4. Создание массива
5. Выход за границы массива
6. Перебор элементов массива
7. Заполнение массива
8. ГЕНЕРАТОРЫ
9. Генератор с IF
10. Заполнение массива в обратном порядке

11. Заполнение случайными числами
12. Вывод массива на экран
13. Ввод с клавиатуры
14. Ввод списка строк через пробел
15. Ввод с клавиатуры списка (массива) целых чисел через пробел
16 Использование функции map
17 Ввод с клавиатуры списка (массива) целых чисел через пробел в одну строчку

1. Что такое массив?

Как ввести 10000 переменных?

Массив – это группа переменных одного типа, расположенных в памяти рядом (в соседних ячейках) и имеющих общее имя.

Надо:

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

Массив можно составить не только из чисел, но и из данных любых типов, например, символьных строк:

```
A = ["Вася","Петя","Коля","Маша","Даша"]
```

Длина массива (количество элементов в нём) определяется с помощью функции len:

```
N = len(A)
```

Таким образом, в любой момент массив «знает» свой размер.

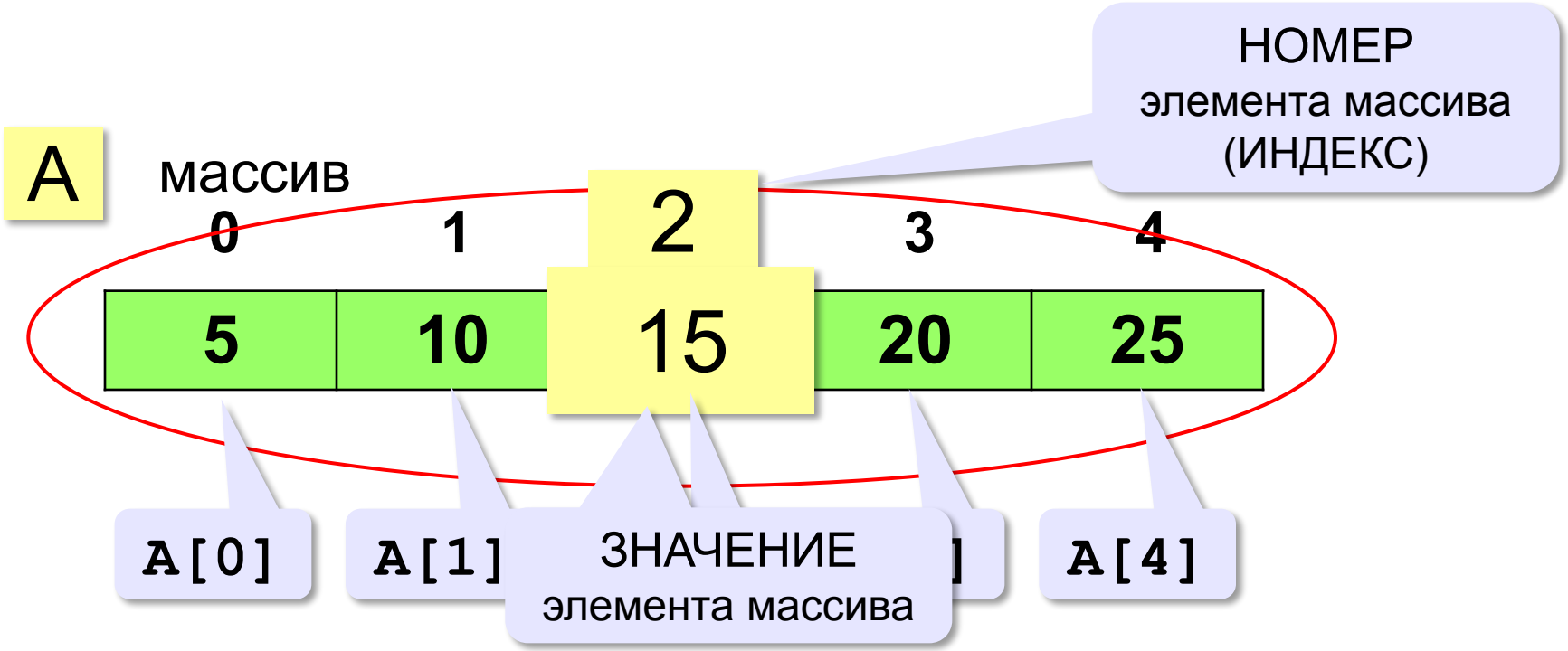
2. Индексы

Для обращения к элементам списка надо использовать **индексы**, которые представляют номер элемента в списка.

Индексы начинаются с нуля. То есть второй элемент будет иметь индекс 1. Для обращения к элементам с конца можно использовать отрицательные индексы, начиная с -1. То есть у последнего элемента будет индекс -1, у предпоследнего - -2 и так далее.

```
numbers = [1, 2, 3, 4, 5]
print(numbers[0])   # 1
print(numbers[2])   # 3
print(numbers[-3])  # 3

numbers[0] = 125  # изменяем первый элемент списка
print(numbers[0])   # 125
```

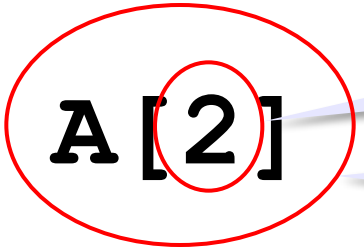
3. Обращение к элементу массива



Индекс элемента — это значение, которое указывает на конкретный элемент массива.

! Нумерация с нуля!

Обращение к элементу массива



ИНДЕКС элемента массива: 2

ЗНАЧЕНИЕ элемента массива

0	1	2	3	4
23	12	7	43	51

```
i = 1
A[2] = A[i] + 2*A[i-1] + A[2*i+1]
print( A[2]+A[4] )
```

? Что получится?

```
A[2] = A[1] + 2*A[0] + A[3]
print( A[2]+A[4] )
```

```
101
152
```

4. Создание массива

```
A = [11, 22, 35, 41, 53]
```

11	22	35	41	53

```
A = [11, 22] + [35, 41] + [53]
```

```
A = [11]*5
```

11	11	11	11	11

5. Выход за границы массива Что неверно?

```
A = [1, 2, 3, 4, 5]
x = 1
print( A[x-3] )
A[x+4] = A[x-1] + A[2*x]
```

? Что плохо?

```
print( A[-2] )
A[5] = A[0] + A[2]
```

Выход за границы массива — это обращение к элементу с индексом, который не существует в массиве.

6. Перебор элементов массива

```
N = 10
A = [0]*N  # память уже выделена
```

Перебор элементов: просматриваем все элементы массива и, если нужно, выполняем с каждым из них некоторую операцию.

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

7. Заполнение массива

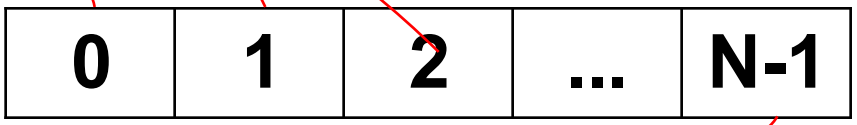
```
[0, 2, 3, …, N-1]
```

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

? Что произойдёт?

В развёрнутом виде

```
A[0] = 0
A[1] = 1
A[2] = 2
...
A[N-1] = N-1
```



В стиле Python:

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

```
#Заполнение массива
N = 3
A = [0]*N
print(A, type(A),len(A))
for i in range(N):
# здесь работаем с A[i]
    A[i]=i
    print("A[", i, "]=", A[i])
# [0, 0, 0] <class 'list'> 3
# A[ 0 ]= 0
# A[ 1 ]= 1
# A[ 2 ]= 2
```

8. ГЕНЕРАТОРЫ

Две операции – создание и заполнение массива – можно объединить в одну с помощью *генератора* – выражения, напоминающего цикл:

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

- Цикл **`for i in range(N)`** перебирает все значения i от 0 до N–1.
- Выражение перед словом **`for`** (в данном случае – i) – это то, что записывается в очередной элемент массива для каждого i.
- В приведённом примере массив заполняется значениями, которые последовательно принимает переменная i, то есть при N=10 мы построим такой массив:

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

```
# Генератор_1
"""
```

Две операции – создание и заполнение массива – можно объединить в одну с помощью генератора – выражения, напоминающего цикл:

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

N=5
A = [i for i in range(N)]
print(A)

#[0, 1, 2, 3, 4]
```

```
#В стиле Python
N = 3
A = [0]*N
print(A, type(A),len(A),id(A))
A = [ i  for i in range(N) ]
print(A, type(A),len(A),id(A))
# [0, 0, 0] <class 'list'> 3 56218408
# [0, 1, 2] <class 'list'> 3 57764456
```

Тот же результат можно получить, если использовать функцию list для того, чтобы создать список из данных, которые получаются с помощью функции range:

```
A = list( range(N) )
```

Для заполнения массива квадратами этих чисел можно использовать такой генератор:

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

#9 Генератор с IF

```
A = [i for i in range(100) if i % 7 == 0]
print(A)
```

```
#[0, 7, 14, 21, 28, 35, 42, 49, 56, 63,
70, 77, 84, 91, 98]
```

В конце записи генератора можно добавить условие отбора.
В этом случае в массив включаются лишь те из элементов, перебираемых в цикле, которые удовлетворяют этому условию.
Например следующий генератор составляет массив из всех чисел в диапазоне от 0 до 99, которые делятся на 7:

```
A = [i for i in range(100)
        if i % 7 == 0]
```

Обратите внимание, что длина этого массива будет меньше 100, и цикл

```
for i in range(100):
print( A[i] )
```

приведёт к ошибке – выходу за границы массива.

10.Заполнение массива в обратном порядке

N	…	3	2	1

```
A[0] = N
A[1] = N-1
A[2] = N-2
...
A[N-1] = 1
```

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

? Как меняется X?

X = N, N-1, …, 2, 1

начальное значение

уменьшение на 1

Заполнение массива в обратном порядке

N	...	3	2	1

A[i] = X

Как связаны i и X?

i	X
0	N
1	N-1
2	N-2
...	...
N-1	1

+1

–1

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

В стиле Python:

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

Сумма i и X не меняется!

```
i + X = N
X = N - i
```

11. 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) ]
```

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

```
#Заполнение случайными числами
#В краткой форме:
from random import randint
N = 10
A = [ randint(20,100)
      for i in range(N) ]
print(A)
```

12. Вывод массива на экран

Весь массив сразу:

```
print( A )
```

```
[1,2,3,4,5]
```

По одному элементу:

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

в столбик

или так:

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

для всех элементов в массиве A

? Как вывести в строчку?

```
for x in A:
  print( x, end=" " )
```

пробел между элементами

```
# Заполнение массива в обратном порядке
N = 3
A = [0]*N
print(A, type(A),len(A),id(A))
for i in range(N):
    A[i] = N-i
print(A, type(A),len(A),id(A))
# [0, 0, 0] <class 'list'> 3 54514472
# [3, 2, 1] <class 'list'> 3 54514472
```

```
#Заполнение массива в обратном порядке
#В стиле Python:
N = 3
A = [0]*N
print(A, type(A),len(A),id(A))
A = [ N-i for i in range(N) ]
print(A, type(A),len(A),id(A))
# [0, 0, 0] <class 'list'> 3 60740392
# [3, 2, 1] <class 'list'> 3 62286440
```

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
```

пробел после вывода очередного числа

или так:

```
for x in A:
  print ( x, end=" " )
```

```
1 2 3 4 5
```

или так:

```
print ( *A )
```

⟺

```
print (1, 2, 3, 4, 5)
```

разбить список на элементы

#Вывод массива – способы 1 и 2
"""
Массив – это набор элементов, поэтому во многих языках программирования
нельзя вывести массив одной командой. Однако в языке Python такая возможность есть
"""

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

"""
Можно вывести элементы массива на экран по одному, ис-пользуя цикл:
"""

```
for i in range(len(A)):
    print( A[i], end=" " )

# 1 2 3 4 5
```

"""
Параметр end определяет, что после вывода каждого элемента добавляется
пробел, а не символ перехода на новую строку.
"""

#Вывод массива – способ 3
"""
Здесь не используются переменная-индекс i и функция len,
а просто перебираются все элементы массива.
На каждой итерации цикла в переменную x
заносится значение очередного элемента массива
(в порядке возрастания индексов).
Такой цикл перебора очень удобен,
если не нужно изменять значения элементов массива.
"""

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

#Вывод массива – способ 4
"""
В языке Python существует ещё один замечательный способ вывода всех элементов массива через пробел (без скобок):
Знак * перед именем массива означает, что нужно преобразо-вать массив в набор отдельных значений,
"""

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

13. Ввод с клавиатуры

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

Что плохо?

или так:

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

С подсказкой для ввода:

```
for i in range(N):
  s = "A[" + str(i) + "]="
  A[i] = int(input(s))
```

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

Ввод с клавиатуры (Python)

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()]
```

```
# Ввод массива с клавиатуры 1
"""
```

Иногда небольшие массивы вводятся с клавиатуры.
Строим цикл, который выполняет оператор ввода отдельно для каждого элемента массива:

```
"""

N=3
A = [0 for i in range(N)]
for i in range(N):
    A[i] = int( input() )
print(A)
```

```
# Ввод массива с клавиатуры 2
"""
Вместо цикла можно использовать генератор,
который сразу создаёт массив и заполняет
его введёнными числами:
"""
N=3
A = [ int(input()) for i in range(N) ]
print(A)
```

Ввод массива с клавиатуры 3

"""

Значительно удобнее, если перед вводом появляется сообщение с подсказкой:

"""

```
N=3
A = [0 for i in range(N)]
for i in range(N):
    print( "A[{}]=".format(i), end="" )
    A[i] = int( input() )
print(A)
```

Ввод массива с клавиатуры 3

"""

Значительно удобнее, если перед вводом появляется сообщение с подсказкой:

"""

```
N=3
A = [0 for i in range(N)]
for i in range(N):
    # print( "A[{}]=".format(i), end="" )
    print("A[" + str(i), "]=", end="", sep="")
    A[i] = int( input() )
print(A)
# A[0]=1
# A[1]=2
# A[2]=3
# [1, 2, 3]
```

```
# Ввод массива с клавиатуры 4
a=[]
n=int(input("размер массива ="))
for i in range(n):
    a.append(int(input()))
print(a)
"""
размер массива =3
4
5
6
[4, 5, 6]
"""
```

```
# Ввод массива с клавиатуры 5
#Добавление элемента в список осуществляется
#с помощью метода append().
a=[]
n=int(input("размер массива ="))
for i in range(n):
    print(i,"-элемент=",end="")
    a.append(int(input()))

print(a)
"""
размер массива =3
0 -элемент=32
1 -элемент=45
2 -элемент=77
[32, 45, 77]
"""
```

```
# Ввод массива с клавиатуры 6
# Добавление элемента в список осуществляется
# с помощью метода append().
a = []
n = int(input("размер массива ="))
for i in range(n):
    # print(i, "-элемент=", end="")
    print("a["+str(i),"]=", end="",sep="")
    a.append(int(input()))

print(a)
# размер массива =2
# a[0]=3
# a[1]=4
# [3, 4]
```

#14 Ввод списка строк через пробел

#Использование split()

"""

Метод Python split string разбивает
строку с помощью указанного спецсимвола
и возвращает список (массив)подстрок.

"""

```
z="ff gg hh jj"
A=z.split()
print(A)
#['ff', 'gg', 'hh', 'jj']
```

 # 15 Ввод с клавиатуры списка (массива)
##целых чисел через пробел

```
A = input().split()
print(A)
for i in range(len(A)):
   A[i] = int(A[i])
print(A)
"""
2 4 6 8 9
['2', '4', '6', '8', '9']
[2, 4, 6, 8, 9]
"""
```

#16 Использование функции map

"""

В Python функция map принимает два аргумента: ***функцию и аргумент составного типа данных, например, список.***

map применяет к каждому элементу списка переданную функцию.

Например, вы прочитали из файла список чисел, изначально все эти числа имеют строковый тип данных, чтобы работать с ними - нужно превратить их в целое число:

"""

```
A = list(map(int, input().split()))
print(A)
#3 44 55 234 32
#[3, 44, 55, 234, 32]
```

 # 17Ввод с клавиатуры списка (массива)
#целых чисел через пробел в одну строчку

"""

Используя функции языка map и list
то же самое можно сделать в одну строку:

"""

```
A = list(map(int, input().split()))
print(A)
#3 44 55 234 32
#[3, 44, 55, 234, 32]
```

Ввод с клавиатуры (Python)

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()]
```

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`

```
# Ввод с клавиатуры (Python)
# Ввод всех чисел
# без подсказки
# в одной строке
# через пробел:
data = input("Введите числа через пробел: ")
# 11 22 33 44 55
print(data,type(data),"len=", len(data))
#11 22 33 44 55 <class 'str'> len= 14
s = data.split()
print(s,type(s),"len=",len(s))
#['11', '22', '33', '44', '55'] <class 'list'> len= 5
A = [int(x) for x in s ]
print(A)
# [11, 22, 33, 44, 55]
```

```
# Ввод с клавиатуры (Python)
# Ввод всех чисел
# без подсказки
# в одной строке
# через пробел:
A = [int(x) for x in input("Введите цифры:").split()]
print(A, type(A),len(A))
#[11, 22, 33] <class 'list'> 3
```

#21
'''
В Python функция map принимает два аргумента: функцию и аргумент составного типа данных, например, список.
map применяет к каждому элементу списка переданную функцию.
Используя функции языка map и list то же самое можно сделать в одну строку:
'''

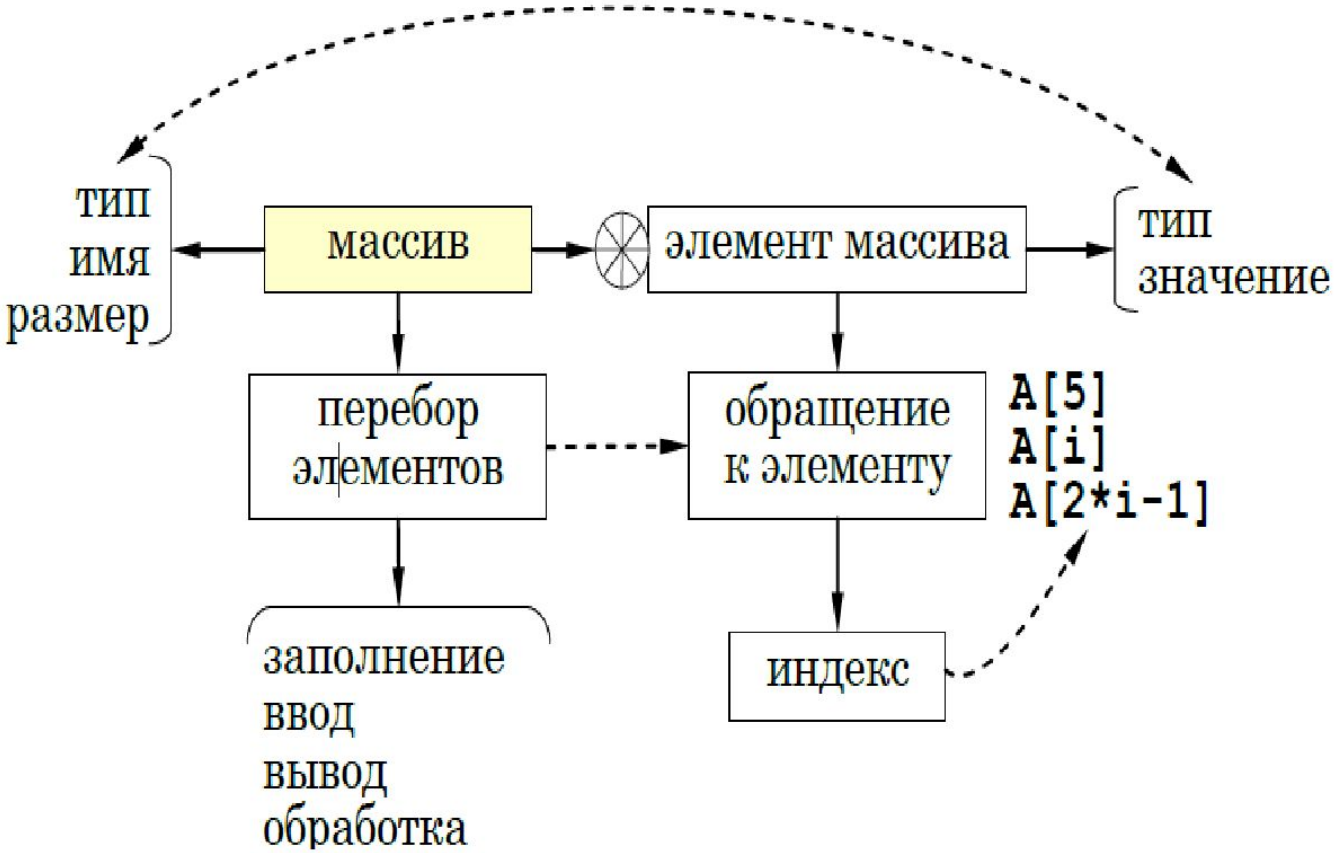
```
A = list(map(int, input('Введите целые числа через пробел. Конец ввода- Enter: ').split()))

print(A)
```

#22 Join – присоединиться
#Описание Функция join() возвращает строку, в которой строковые элементы
#последовательности были соединены с помощью сепаратора str.

```
A = ['red', 'green', 'blue']
b='$'.join(A)#
print(type(b),b)#<class 'str'> red$green$blue
print(' '.join(A)) # red green blue
print(''.join(A)) # redgreenblue
print('***'.join(A)) # red***green***blue
```

#23

```
old_list = ['1', '2', '3', '4', '5', '6', '7']
new_list = list(map(int, old_list))
print (new_list)
[1, 2, 3, 4, 5, 6, 7]
```


тип
имя
размер
массив
элемент массива
тип
значение
перебор
элементов
обращение
к элементу
A[5]
A[i]
A[2*i-1]
заполнение
ввод
вывод
обработка
индекс

Выводы:

- Массив – это группа переменных одного типа, расположенных в памяти друг за другом и имеющих общее имя. Массивы используют для того, чтобы было удобно работать с большим количеством данных.
- Индекс элемента массива – это значение, которое указывает на конкретный элемент массива.
- При обращении к элементу массива индекс указывают в квадратных скобках. Это может быть число, имя переменной целого типа или арифметическое выражение, результат которого – целое число.
- Для перебора элементов массива удобно использовать цикл по переменной, которая изменяется от минимального до максимального значения индекса.