Язык программирования Python

Массивы в языке Python

Массив – совокупность пронумерованных величин одного типа, объединённых общим именем.

В языке Python нет такой структуры данных, как массив.

Для хранения группы однотипных объектов используют ***списки*** (тип данных **`list`**).

Индекс – порядковый номер элемента в массиве.

Нумерация элементов массива всегда начинается с нуля.

Каждый элемент массива обозначается *индексированным именем*:

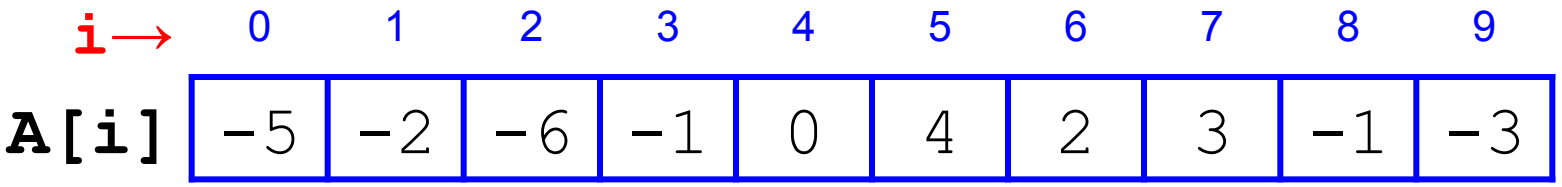
`Имя[индекс]`

Например:

`A[1]` – второй элемент массива A (с индексом 1).

Массивы бывают *одномерные* (*линейные*) и *двумерные* (*прямоугольные*). Далее рассматриваются одномерные массивы.

Одномерный (линейный) массив **A**



A – *имя массива*

i – *индекс элемента*

A[0] = -5, A[1] = -2, . . . A[9] = -3

Индексом может быть не только целое число, но и целое значение переменной или арифметического выражения.

Например: **A[2*i-1] = -2** (при **i=1**).

Перед использованием в программе массив необходимо создать. Обращение к несуществующему элементу вызовет ошибку.

Количество элементов в массиве определяется с помощью функции **len** (length – «длина») . Например: `N = len(A)`

Вывод массива на экран

1 способ. Весь массив выводится как один объект в квадратных скобках, элементы разделяются запятыми.

```
print (A)
```

На экране:

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

2 способ. Вывод элементов с помощью цикла в одной строке через пробел.

```
for i in range(len(A)):
    print (A[i], end=" ")
print()   # переход на новую строку
```

На экране:

```
1 2 3 4 5
```

Вывод массива на экран

3 способ. Вывод элементов с помощью цикла в столбик.

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

На экране:

```
1
2
3
4
5
```

4 способ. Вывод элементов с помощью цикла в столбик с указанием индексов.

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

На экране:

```
A[ 0 ]= 1
A[ 1 ]= 2
A[ 2 ]= 3
A[ 3 ]= 4
A[ 4 ]= 5
```

Заполнение массивов

1 способ. Создание массива указанием значений элементов

Массив создается перечислением элементов через запятую в квадратных скобках.

```
A = [1, -2, -3, 5, 7]
```

Если все элементы одинаковые, используется следующий оператор.

```
# массив из 5 элементов
# заполненный нулями
A = [0] * 5
```

Заполнение массивов

2 способ. Ввод с клавиатуры (при небольшом количестве элементов)

```
N = 5                       # размер массива в переменной
B = [0] * N                   # заполнение массива нулями
print ("Введите", N, "элементов массива:")
for i in range(N):              # перебор индексов
    B[i] = int(input())       # ввод числа с клавиатуры
```

Можно в цикл добавить подсказку с индексом вводимого элемента.

```
for i in range(N):                # перебор индексов
    print ("B[", i, "] = ", end="")  # вывод подсказки
    B[i] = int(input())           # ввод числа
```

На экране:

```
B[ 0 ] = 1
B[ 1 ] = 2
B[ 2 ] = 3
B[ 3 ] = 4
B[ 4 ] = 5
```

Заполнение массивов

3 способ. Вычисление элементов по формуле (функция от индекса)

```
N = 5           # размер массива в переменной
C = [0] * N       # заполнение массива нулями
for i in range(N):  # перебор индексов
    C[i] = i**2     # индекс в квадрате
print (C)           # вывод массива
```

На экране:

```
[0, 1, 4, 9, 16]
```

Заполнение массивов

4 способ. Заполнение случайными числами

Функция **randint(a, b)** создаёт случайное целое число из отрезка [a, b].

```
N = 5                       # размер массива в переменной
D = [0] * N                   # заполнение массива нулями
from random import randint  # подключение функции randint
for i in range(N):            # перебор индексов
    D[i] = randint(-5, 5)   # случайные числа от -5 до 5
print (D)                     # вывод массива
```

Возможный результат на экране:

```
[0, -4, -2, 1, 5]
```

Задача 1

Определить средний балл 10 учеников, сдававших ЕГЭ по информатике.

```
# Средний балл учеников
N = 10              # размер массива
A = [0] * N         # заполнение массива нулями
# Ввод значений элементов массива с клавиатуры
print ("Введите оценки:")
for i in range(N):
    print (i+1, "оценка: ", end="")
    A[i] = int(input())
s = 0                       # нач. знач. суммы
for i in range(N):          # перебор индексов
    s = s + A[i]            # добавление к сумме
sb = s/10                   # среднее арифметиче
print ("Средний балл:", sb)
```

```
Введите оценки:
1 оценка: 3
2 оценка: 4
3 оценка: 4
4 оценка: 3
5 оценка: 3
6 оценка: 5
7 оценка: 4
8 оценка: 5
9 оценка: 5
10 оценка: 3
Средний балл: 3.9
```

Задача 2

Подсчитать количество элементов массива, которые больше заданного значения.

```
# Количество элементов массива, соответствующих условию
N = 10
A = [0]*N           # создание массива
from random import randint  # подключение функции randint
for i in range(N):          # заполнение массива
    A[i] = randint(0, 99)   # случайными числами от 0 до 99
print (A)           # вывод массива
x = int(input("x = "))   # ввод значения для условия
k = 0               # начальное значение счетчика
for i in range(N):          # просмотр всех элементов массива
    if A[i] > x:        # если очередной соответ. условию
        k = k+1         # увеличиваем счетчик
print ("Количество элементов больше данного", k)
```

```
[30, 81, 28, 35, 35, 94, 9, 76, 25, 40]
x = 50
Количество элементов больше данного 3
```

Многомерные массивы

Пример:

```
mass = [[1, 2, 3, 4], [5, 6, 7, 8]]  # элементы массива начинаются с 0
print(mass[0])
print(mass[1])
print(mass[0][3])
```

Результат:

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

Считывание массива

Пример:

```
mass = [1,2,3,4,5]
for i in mass:      #Перебирает все элементы массива
    print(i, end=“ “)
```

Результат:

```
1 2 3 4 5
```

Пример:

```
mass = [1,2,3,4,5], [6,7,8,9,10]
for i in mass: # переменная i принимает значение от 1 до 2
    for k in i:
        print(k,end=" ")
```

Результат:

```
1 2 3 4 5 6 7 8 9 10
```

Использование функции .append

Пример:

```
a1 = []
a1.append(23)
print(a1)
a1.append(64)
print(a1)
Результат:
[23]
[23,64]
```

Использование функции .append

Пример:

```
a1 = []
from random import *
for j in range(3):
    a2 = []
    for i in range(3):
        a2.append(randint(1,10))
    a1.append(a2)
print(a1)
print(a1[2][2])
```

Результат:

```
[[8, 10, 10], [3, 7, 10], [10, 7, 8]]
8
```

Функция добавления данных в конец списка .extend

Пример:

```
a=[1,2,3,5]
b=[3,4,5]
a.append(b)
print(a)
```

Результат:

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

Пример:

```
a=[1,2,3,5]
b=[3,4,5]
a.extend(b)
print(a)
```

Результат:

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

Функции удаления
(remove(значение)))

Пример:

mass = [1,2,3,4,55,55]

mass.remove(55)

print(mass)

Результат:

[1, 2, 3, 4, 55]

Удаляет первое совпавшее число.

Пример:

mass = [1,2,3,4,897],[1,2,3,55]

mass[1]**.remove(55)**

print(mass)

Результат:

([1, 2, 3, 4, 897], [1, 2, 3])

Если число не будет входить в массив, то программа даёт ошибку

Функции удаления
(pop(индекс))

Пример:

```
mass = [1, 2, 3, 4]
print(mass)
mass.pop(1)
print(mass)
```

Результат:

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

Пример:

```
mass = [1, 2, 3, 4],[1,3,4]
print(mass)
mass[0].pop(1)
print(mass)
```

Результат:

```
([1, 2, 3, 4], [1, 3, 4])
([1, 3, 4], [1, 3, 4])
```

Функции удаления *(del имя массива[индекс][индекс]))*

Пример:

```
mass = ['ноль', 1, 3, 'IV']
print(mass)
del mass[0]
print(mass)
```

Результат:

```
['ноль', 1, 3, 'IV']
[1, 3, 'IV']
```

Пример:

```
mass = ['ноль', 1, 3, 'IV'],[1,2,3]
print(mass)
del mass[0][1:4] # удаляет диапазон
print(mass)
```

Результат:

```
(['ноль', 1, 3, 'IV'], [1, 2, 3])
(['ноль'], [1, 2, 3])
```

Функция копирования copy.copy()

Пример:

```
import copy
a=[1,2,3],[4,5,6]
b=[]
b=copy.copy(a)
print(b)
```

Результат:

```
([1, 2, 3], [4, 5, 6])
```

Функция определения количества элементов .count()

Пример:

```
a=[1,2,3,453,3,5],[3,2,9]
b=a[0].count(3)  #указывается номер элемента
print(b)
```

Результат:

```
2
```

Функция поиска индекса элемента .index()

Пример:

```
a=[1,2,3,453,3,5]
b=a.index(3)
print(b)
```

Результат:

```
2
```

Пример:

```
a=[1,2,453,5,6,3]
b=a.index(4,0)
print(b, end=' ')
```

Результат:

```
NameError: name 'c'
is not defined
```

```
a=[1,2,453,5,3,6,3]
b=0
c=int(input('Какое число вы хотите найти:'))
while b<len(a):
    b=a.index(c,b)
    print(b, end=' ')
    b=b+1
```

Функция вставки .insert(позиция, значение)

Пример:

```
a=[1,2,3,4,5]
print(a)
a.insert(0,9)
print(a)
```

Результат:

```
[1, 2, 3, 4, 5]
[9, 1, 2, 3, 4, 5]
```

Функция разворачивания списка .reverse()

Пример:

```
a=[1,2,3,4,5]
print(a)
a.reverse()
print(a)
```

Результат:

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

Пример:

```
a=[1,2,3,4,5]
print(a)
for i in reversed(a):
    print(i,end=' ')
```

Результат:

```
5 4 3 2 1
```

Функция сортировки списка .sort

Пример:

```
a=[13,67,-8,56]
print(a)
a.sort()
print(a)
```

Результат:

```
[13, 67, -8, 56]
[-8, 13, 56, 67]
```

Пример:

```
a=['a','j','r','b']
print(a)
b=sorted(a)
print(b)
```

Результат:

```
['a', 'j', 'r', 'b']
['a', 'b', 'j', 'r']
```

Практическая работа

Дан массив из 10 случайных чисел в диапазоне от 0 до 9. Необходимо его отсортировать, повторяющиеся значения удалить, добавить недостающие цифры.

Примерный результат:

[7, 7, 8, 8, 6, 5, 3, 7, 2, 8] – исходный массив

[2, 3, 5, 6, 7, 7, 7, 8, 8, 8] – отсортированный массив

[2, 3, 5, 6, 7, 8] – массив после удаления лишних эл.

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] – массив после добавления недостающих цифр

Метод	Значение
append()	Добавляет элементы в конец списка
clear()	Удаляет все элементы в списке
copy()	Возвращает копию списка
count()	Возвращает число элементов с определенным значением
extend()	Добавляет элементы списка в конец текущего списка
index()	Возвращает индекс первого элемента с определенным значением
insert()	Добавляет элемент в определенную позицию
pop()	Удаляет элемент по индексу
remove()	Убирает элементы по значению
reverse()	Разворачивает порядок в списке
sort()	Сортирует список

Кортежи

```
my_tuple = (1, 2, 3, 4, 5)
a = my_tuple[0:3]# индексы от 0 до 3-1
print(a) # (1, 2, 3)

another_tuple = tuple() # объявление кортежа
abc = tuple([1, 2, 3]) # превращение списка в кортеж

abc_list = list(abc) #превращение кортежа в список
```

Вывод элементов кортежа

```
abc = tuple([1, 2, 3])
print(abc[0])
```

Вывод

1

Встроенные функции кортежей

len(tuple)	Возвращает длину кортежа.
max(tuple)	Возвращает максимальное значение в кортеже.
min(tuple)	Возвращает минимальное значение в кортеже.
sum(tuple)	Суммирует все числа в кортеже. Если в кортеже встречается строка, выдает ошибку.
sorted(tuple)	Сортирует кортеж.

Словари

Словари в Python — это неупорядоченные коллекции произвольных объектов, имеющих доступ к ним по ключу.

```
my_dict = {}
another_dict = dict()
my_other_dict = {"one":1, "two":2, "three":3}
print(my_other_dict)
```

Вывод:

```
{'one': 1, 'two': 2, 'three': 3}
```

Создание словаря с помощью функции zip()

```
name = ['Артур', 'Михаил', 'Игорь', 'Марина']
ages = [17, 38, 45, 40]
student = dict(zip(name, ages))
print(student)
```

Вывод:

```
{'Артур': 17, 'Михаил': 38, 'Игорь': 45, 'Марина': 40}
```

Вывод значений из словаря

```
my_other_dict = {"one":1, "two":2, "three":3}
print(my_other_dict["one"])
my_dict = {"name":"Mike", "address":"123 Happy Way"}
print(my_dict["name"])
```

Вывод:

```
1
Mike
```

Поиск ключей в словаре

```
my_dict = {"name":"Mike”}
print("name" in my_dict)
print("state" in my_dict)
Вывод:
True
False

my_dict = {"name":"Mike","2":123}
print(my_dict.keys())
Вывод: dict_keys(['name', '2'])
```