Array in Python

Arrays are a fundamental part of all programming languages, it is a collection of elements of a single data type, for example, integer arrays, string arrays. However, in Pythong, there is no original array data structure. So we use Python lists instead of arrays.

Arrays are a fundamental part of all programming languages, it is a collection of elements of a single data type, for example, integer arrays, string arrays. Unlike arrays, each list can store elements with any data type and do everything the array can do. We can store integers, decimals, strings in the same list. Therefore, working with the list is quite flexible. However, in Python, there is no original array data structure. So we use Python lists instead of arrays.

An array of numeric values ??is supported in Python by the array module.

Note: If you want to create an array really in Python, you need to use the array data structure of NumPy. To solve math problems, the NumPy array will be more efficient.

List and array modules in Python

You can manipulate the list as an array but cannot cast the element stored in the list. For example:

```
a = [1, 3.5, "Hello"]
```

If you create an array using an array module, all array elements must have the same number type.

```
import array as arr
a = arr.array ('d', [1, 3.5, "Hello"]) // Run this code to report an error
```

How to create array in Python?

From the above examples you can guess, we need to enter the array module to create arrays. For example:

```
import array as arr
a = arr.array ('d', [ 1.1 , 3.5 , 4.5 ]) print ( a )
```

The code on creating the array has type float. The word 'd' is type code, which determines the type of array in the creation process. Here are the common style codes:

TypeCython Type Type Minimum size in bytes `'b'` signed char int 1 `'B'` unsigned char int 1 `'u'` Py_UNICODE 2 Unicode character `'h'` signed short int 2 `'H'` unsigned short int 2 `'i'` signed int int 2 `'I'` unsigned int int 2 `'l'` signed long int 4 `'L'` unsigned long int 4 `'f'` float float 4 `'d'` double float 8

We will not discuss the different C data types in this article. We will use the code 'i' for integers and 'd' for decimals in the whole lesson.

Note: The code 'u' for Unicode characters is no longer accepted from Python version 3.3. Avoid using it when possible.

How to access array elements?

We use index to access array elements. Index also starts at 0, similar to the Python list.

```
import  array as  arr a =  arr . array ( 'i' , [ 2 , 4 , 6 , 8 ]) print ( "Ph
?n t? ??u tiên:" ,  a [ 0 ]) print ( "Ph?n t? th?
 2:" ,  a [ 1 ]) print ( "Ph?n t? cu?i cùng:" ,  a [- 1 ])
```

Run the program above:

```
  First element: 2
  Element 2: 4
  Last element: 8
```

You can access a range of elements in the array, using the slice operator:.

```
import  array as  arr numbers_list = [ 5 , 85 , 65 , 1 5 , 95 , 52 , 36 , 2 5 ]
?n t? th? 3 ??n 5 print ( numbers_array [:- 5 ]) # Ph?n t? ??u tiên ??
n 4 print ( numbers_array [ 5 :]) # Ph?n t? th? 6 ??n h?
t print ( numbers_array [:]) # Ph?n t? ??u tiên ??n cu?i cùng
```

When you run the above code, you will get the output:

```
  array ('i', [65, 15, 95])
  array ('i', [5, 85, 65])
  array ('i', [52, 36, 25])
  array ('i', [5, 85, 65, 15, 95, 52, 36, 25])
```

Change, add element in Python array

Arrays can be changed, its elements can be changed in the same way as lists.

```
import  array as  arr numbers =  arr . array ( 'i' , [ 1 , 1, 2 , 5 , 7 , 9 ]) #
??i ph?n t? ??
u tiên  numbers [ 0 ] = 0 print ( numbers ) # Output: array('i', [0, 1, 2, 5, 7,
?n t? th? 3 ??n th?
 5  numbers [ 2 : 5 ] =  arr . array ( 'i' , [ 4 , 6 , 8 ]) print ( numbers ) # O
```

Do you add an item to the list using append () or add some items using extend ():

```
import  array as  arr numbers =  arr . array ( 'i' , [3, 5, 7 ])  numbers . appe
?i vào cu?i m?
ng  numbers . extend ([ 5 , 6 , 7 ]) print ( numbers ) # Output: array('i', [3, 5
```

Two arrays can also be recombined into one by the + operator:

```
import  array as  arr mang_le =  arr . array ( 'i' , [ 3 , 5 , 7 ])  mang_chan =
?o m?ng tr?
ng  numbers =  mang_le + mang_chan  # Code by quantrimang.com print ( numbers ) #
```

Delete element of array in Python

To delete one or more elements of the array we use the del command.

```
import  array as  arr number =  arr . array ( 'i' , [ 1 , 3 , 3 , 5 , 7 ]) del  n
?n t? th?
 3 print ( number ) # Output: array('i', [1, 3, 5, 7]) del  number # xóa toàn b
? m?ng print ( number ) # Error: array 'number' is not defined
```

You can use remove () to delete the given item or pop () to delete the item with the given index:

```
import  array as  arr numbers =  arr . array ( 'i' , [ 1 , 1 , 3 , 5 , 9 ])  numb
```

You can learn more about the Python array methods here.

When to use array?

List is more flexible than array, they can store elements with many different data types, including strings. List is also faster than array, so why use an array? If you have to perform array and matrix calculations, you should use the NumPy library. Unless you really need arrays (array modules may need to communicate with the C code), don't use them.

Previous post: Pakage in Python