Python Arrays

In programming, an array is a collection of elements of the same type.
Arrays are popular in most programming languages like Java, C/C++, JavaScript and so on. However, in Python, they are not that common. When people talk about Python arrays, more often than not, they are talking about Python lists.

Python Lists Vs array Module as Arrays

We can treat lists as arrays. However, we cannot constrain the type of elements stored in a list. For example:

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

If you create arrays using the `array` module, all elements of the array must be of the same numeric type.

```
import array as arr


a = arr.array('d', [1, 3.5, "Hello"])   // Error
```

How to create arrays?

As you might have guessed from the above example, we need to import `array` module to create arrays. For example:

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

Here, we created an array of `float` type. The letter `'d'` is a type code. This determines the type of the array during creation.

Commonly used type codes:

Code	C Type	Python Type	Min bytes
`'b'`	signed char	int	1
`'B'`	unsigned char	int	1

`'u'`	Py_UNICODE	Unicode	2
`'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 about different C types in this article. We will use two type codes in this entire article: `'i'` for integers and `'d'` for floats.
Note: The `'u'` type code for Unicode characters is deprecated since version 3.3. Avoid using it when possible.

How to access array elements?

We use indices to access elements of an array:

```
1. import array as arr
2. a = arr.array('i', [2, 4, 6, 8])
3.
4. print("First element:", a[0])
5. print("Second element:", a[1])
6. print("Last element:", a[-1])
```

Remember, the index starts from 0 (not 1) similar to lists.

How to slice arrays?

We can access a range of items in an array by using the slicing operator `:`.

```
1. import array as arr
2.
```

```
numbers_list = [2, 5, 62, 5, 42, 52, 48, 5]
numbers_array = arr.array('i', numbers_list)

print(numbers_array[2:5]) # 3rd to 5th
print(numbers_array[:-5]) # beginning to 4th
print(numbers_array[5:])  # 6th to end
print(numbers_array[:])   # beginning to end
```

When you run the program, the output will be:

```
array('i', [62, 5, 42])
array('i', [2, 5, 62])
array('i', [52, 48, 5])
array('i', [2, 5, 62, 5, 42, 52, 48, 5])
```

How to change or add elements?

Arrays are mutable; their elements can be changed in a similar way like lists.

```
import array as arr

numbers = arr.array('i', [1, 2, 3, 5, 7, 10])

# changing first element
numbers[0] = 0
print(numbers)     # Output: array('i', [0, 2, 3, 5, 7, 10])

# changing 3rd to 5th element
numbers[2:5] = arr.array('i', [4, 6, 8])
print(numbers)     # Output: array('i', [0, 2, 4, 6, 8, 10])
```

We can add one item to a list using the `append()` method, or add several items using `extend()` method.

```
import array as arr

numbers = arr.array('i', [1, 2, 3])

numbers.append(4)
print(numbers)     # Output: array('i', [1, 2, 3, 4])

# extend() appends iterable to the end of the array
numbers.extend([5, 6, 7])
print(numbers)     # Output: array('i', [1, 2, 3, 4, 5, 6, 7])
```

We can concatenate two arrays using `+` operator.

```
import array as arr

```

```
odd = arr.array('i', [1, 3, 5])
even = arr.array('i', [2, 4, 6])

numbers = arr.array('i')   # creating empty array of integer
numbers = odd + even

print(numbers)
```

How to remove/delete elements?

We can delete one or more items from an array using Python's del statement.

```
import array as arr

number = arr.array('i', [1, 2, 3, 3, 4])

del number[2] # removing third element
print(number) # Output: array('i', [1, 2, 3, 4])

del number # deleting entire array
print(number) # Error: array is not defined
```

We can use the `remove()` method to remove the given item, and `pop()` method to remove an item at the given index.

```
import array as arr

numbers = arr.array('i', [10, 11, 12, 12, 13])

numbers.remove(12)
print(numbers)   # Output: array('i', [10, 11, 12, 13])

print(numbers.pop(2))   # Output: 12
print(numbers)   # Output: array('i', [10, 11, 13])
```

1. Basic example

Here is a simple example of an array containing 5 integers

```
>>> from array import *
>>> my_array = array('i', [1,2,3,4,5])
>>> for i in my_array:
... print(i)
...
1
2
3
4
5
```

So this way we can create a simple python array and print it.

2. Access individual elements through indexes

Individual elements can be accessed through indexes. Here is an example :

```
>>> my_array[1]

2

>>> my_array[2]

3

>>> my_array[0]

1
```

Remember that indexes start from zero.

3. Append any value to the array using append() method

Here is an example :

```
>>> my_array.append(6)
>>> my_array
array('i', [1, 2, 3, 4, 5, 6])
```

So we see that the value ‘6’ was appended to the existing array values.

```
a=arr.array('d', [1.1 , 2.1 ,3.1] )
```

```
a.append(3.4)
print(a)
```

Output –
array(‘d’, [1.1, 2.1, 3.1, 3.4])

a=arr.array('d', [1.1 , 2.1 ,3.1])

a.extend([4.5,6.3,6.8])

print(a)

Output –

array(‘d’, [1.1, 2.1, 3.1, 4.5, 6.3, 6.8])

4. Insert value in an array using insert() method

We can use the insert() method to insert a value at any index of the array. Here is an example :

```
>>> my_array.insert(0,0)
>>> my_array
array('i', [0, 1, 2, 3, 4, 5, 6])
```

In the above example, using insert() method, the value 0 was inserted at index 0. Note that the first argument is the index while second argument is the value.

5. Extend python array using extend() method

A python array can be extended with more than one value using extend() method. Here is an example :

```
>>> my_extnd_array = array('i', [7,8,9,10])


>>> my_array.extend(my_extnd_array)
>>> my_array
array('i', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```

So we see that the array my_array was extended with values from my_extnd_array.

6. Add items from list into array using fromlist() method

Here is an example:

```
>>> c=[11,12,13]


>>> my_array.fromlist(c)
>>> my_array
array('i', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13])
```

So we see that the values 11,12 and 13 were added from list ‘c’ to ‘my_array’.

7. Remove any array element using remove() method

Here is an example :

```
>>> my_array.remove(13)
>>> my_array
array('i', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
```

So we see that the element 13 was removed from the array.

8. Remove last array element using pop() method

Here is an example :

```
>>> my_array.pop()
12
>>> my_array
array('i', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11])
```

So we see that the last element 12 was popped out of array.

9. Fetch any element through its index using index() method

Here is an example :

```
>>> my_array.index(5)
5
```

So we see that the value at index 5 was fetched through this method.

10. Reverse a python array using reverse() method

Here is an example :

```
>>> my_array.reverse()
>>> my_array
array('i', [11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
```

So we see that the complete array got reversed.

11. Get array buffer information through buffer_info() method

This method provides you the array buffer start address in memory and number of elements in array. Here is an example:

```
>>> my_array.buffer_info()
(33881712, 12)
```

So we see that buffer start address and number of elements were provided in output.

12. Check for number of occurrences of an element using count() method

Here is an example :

```
>>> my_array.count(11)
1
```

So we see that the element 11 occurred only once in the array.

13. Convert array to string using tostring() method

Here is an example :

```
>>> my_char_array = array('c', ['g','e','e','k'])

>>> my_char_array
```

```
array('c', 'geek')


>>> my_char_array.tostring()
'geek'
```

So we see that the character array was converted to string using this method.

14. Convert array to a python list with same elements using tolist() method

Here is an example :

```
>>> c = my_array.tolist()
>>> c
[11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

So we see that list ‘c’ was created by using tolist() method on my_array.

15. Append a string to char array using fromstring() method

Here is an example :

```
>>> my_char_array.fromstring("stuff")
>>> my_char_array
array('c', 'geekstuff')
```