Python - Copiar matrizes

Os tipos de sequência integrados do Python, ou seja, lista, tupla e string, são coleções indexadas de itens. No entanto, ao contrário dos arrays em C/C++, Java etc., eles não são homogêneos, no sentido de que os elementos nesses tipos de coleção podem ser de tipos diferentes. O módulo array do Python ajuda você a criar objetos semelhantes aos arrays Java. Neste capítulo, discutimos como copiar um objeto array para outro.

Os arrays Python podem ser do tipo string, inteiro ou flutuante. O construtor da classe array é usado da seguinte forma -

```
import array
obj = array.array(typecode[, initializer])
```

O typecode pode ser uma constante de caractere que representa o tipo de dados.

Podemos atribuir um array a outro pelo operador de atribuição.

```
a = arr.array('i', [1, 2, 3, 4, 5])
b=a.copy()
```

No entanto, tal atribuição não cria um novo array na memória. Em Python, uma variável é apenas um rótulo ou referência ao objeto na memória. Portanto, a é a referência a um array e b também. Verifique o id() de a e b. O mesmo valor de id confirma que a atribuição simples não cria uma cópia

```
import array as arr
a = arr.array('i', [1, 2, 3, 4, 5])
b=a
print (id(a), id(b))
```

Ele produzirá a seguinte **saída** -

```
2771967068656 2771967068656
```

Como "a" e "b" referem-se ao mesmo objeto array, qualquer alteração em "a" também será refletida em "b" -

```
a[2]=10
print (a,b)
```

Ele produzirá a seguinte **saída** -

```
array('i', [1, 2, 10, 4, 5]) array('i', [1, 2, 10, 4, 5])
```

Para criar outra cópia física de um array, usamos outro módulo da biblioteca Python, denominado copy e usamos a função deepcopy() no módulo. Uma cópia profunda constrói um novo objeto composto e, em seguida, insere recursivamente nele cópias dos objetos encontrados no original.

```
import array, copy
a = arr.array('i', [1, 2, 3, 4, 5])
import copy
b = copy.deepcopy(a)
```

Agora verifique o id() de "a" e "b". Você descobrirá que os IDs são diferentes.

```
print (id(a), id(b))
```

Ele produzirá a seguinte **saída** -

```
2771967069936 2771967068976
```

Isso prova que é criado um novo objeto "b" que é uma cópia real de "a". Se alterarmos um elemento em “a”, ele não será refletido em “b”.

```
a[2]=10
print (a,b)
```

Ele produzirá a seguinte **saída** -

```
array('i', [1, 2, 10, 4, 5]) array('i', [1, 2, 3, 4, 5])
```