Python - Copiar Dicionários

Como uma variável em Python é apenas um rótulo ou referência a um objeto na memória, um simples operador de atribuição não criará uma cópia do objeto.

Exemplo 1

Neste exemplo, temos um dicionário "d1" e o atribuímos a outra variável "d2". Se "d1" for atualizado, as alterações também serão refletidas em "d2".

```
d1 = {"a":11, "b":22, "c":33}
d2 = d1
print ("id:", id(d1), "dict: ",d1)
print ("id:", id(d2), "dict: ",d2)

d1["b"] = 100
print ("id:", id(d1), "dict: ",d1)
print ("id:", id(d2), "dict: ",d2)
```

Saída

```
id: 2215278891200 dict: {'a': 11, 'b': 22, 'c': 33}
id: 2215278891200 dict: {'a': 11, 'b': 22, 'c': 33}
id: 2215278891200 dict: {'a': 11, 'b': 100, 'c': 33}
id: 2215278891200 dict: {'a': 11, 'b': 100, 'c': 33}
```

Para evitar isso e fazer uma cópia superficial de um dicionário, use o método copy() em vez de atribuição.

Exemplo 2

```
d1 = {"a":11, "b":22, "c":33}
d2 = d1.copy()
print ("id:", id(d1), "dict: ",d1)
print ("id:", id(d2), "dict: ",d2)
d1["b"] = 100
print ("id:", id(d1), "dict: ",d1)
print ("id:", id(d2), "dict: ",d2)
```

Saída

Quando “d1” for atualizado, “d2” não mudará agora porque “d2” é a cópia do objeto de dicionário, não apenas uma referência.

```
id: 1586671734976 dict: {'a': 11, 'b': 22, 'c': 33}
id: 1586673973632 dict: {'a': 11, 'b': 22, 'c': 33}
id: 1586671734976 dict: {'a': 11, 'b': 100, 'c': 33}
id: 1586673973632 dict: {'a': 11, 'b': 22, 'c': 33}
```