 shallow copy یعنی کپی سطحی و deep copy یعنی کپی عمیق.

نکته ای که همین اول کار باید بدونید اینکه این بحث فقط در مورد آبجکت های mutable (قابل تغییر) مثل list, dict, set وجود داره و آبجکت هایی که immutable هستند مثل tuple ها این بحث درموردشون وجود نداره.

زمانی که شما سعی در کپی گرفتن از یک آبجکت mutable رو دارید اون چیزی که شما انتظار دارید با چیزی که پایتون قراره انجام بده متفاوت هستش. به مثال زیر دقت کنید:

a = [1, 2, 3, 4]

b = a

من در خط اول یک لیست دارم. برای کپی گرفتن از این لیست ساده ترین روشی که به ذهنمون میرسه اینکه اون لیست به یک متغیر دیگه منسوب کنیم، دقیقا مثل خط دوم. اما اینکار یک کپی از لیست نمیگیره و فقط یک اشاره گر جدید به اون لیست ایجاد میکنه.

اگر هم بخواید مطمئن بشید میتونید id های متغیرها رو چاپ کنید:

print( id(a) )

print( id(b) )

کد بالا رو که اجرا کنید میبینید که یک عدد رو نشون میده. به این میگیم shallow copy در پایتون. حالا من سعی میکنیم که لیست b رو تغییر بدم:

a = [1, 2, 3, 4]

b = a

b[0] = 11

print(a)

print(b)

در کد بالا من لیست b رو تغییر دادم اما لیست a هم تغییر خواهد کرد. چرا؟ چون b فقط یک اشاره گر بود به لیست a:

[11, 2, 3, 4]

[11, 2, 3, 4]

برای حل این مشکل ما سه تا متد داریم: list(), dict(), set()

b = list(a)

b[0] = 11

این سه تا متد deep copy میگیرن. حالا اگه لیست b رو تغییر بدین میبینید که لیست a تغییری نکرده.

[1, 2, 3, 4] #a

[11, 2, 3, 4] #b

اما این سه تا متد در زمانی که شما آبجکت های تو در تو دارید به درستی کار نخواهند کرد.

a = [1, 2, 3, 4, [5, 6]]

b = list(a)

b[4][0] = 55

print(a) # [1, 2, 3, 4, [55, 6]]

print(b) # [1, 2, 3, 4, [55, 6]]

همونطور که میبینید زمانی که لیست هامون تو در تو هستند باز هم هردو لیست تغییر میکنن. برای حل کامل مشکل میتونید از ماژول copy استفاده کنید:

import copy

a = [1, 2, 3, 4]

b = copy.copy(a) # shallow copy

c = copy.deepcopy(a) # deep copy

در این ماژول دوتا متد داریم که برامون هم shallow copy و هم deep copy میگیرن.