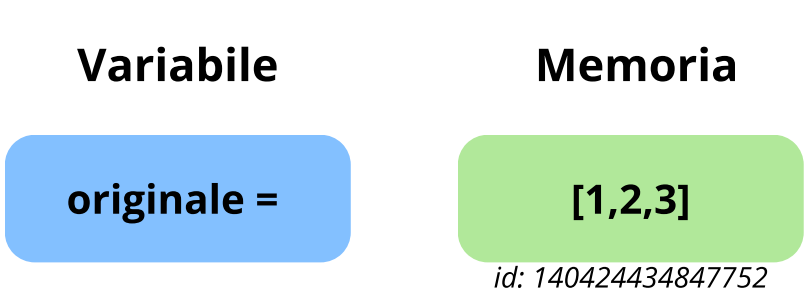
Fase 1: Creazione della lista originale

originale = [1,2,3]
Crea una lista e assegna un riferimento alla variabile

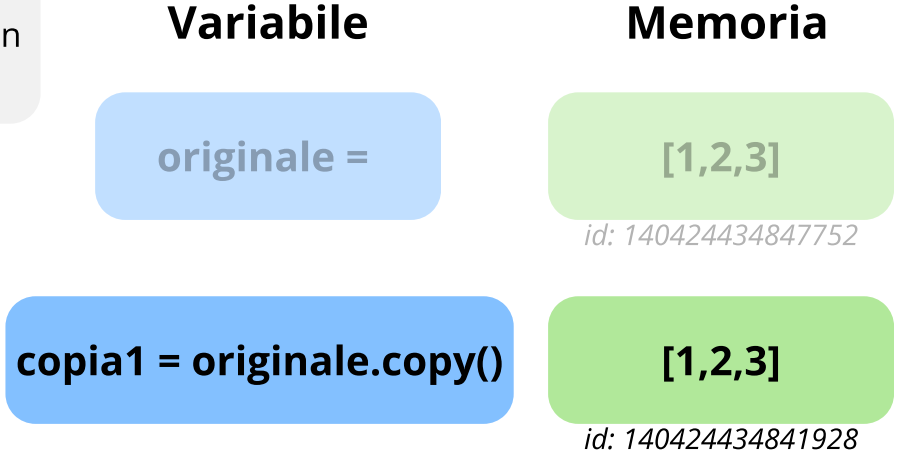


Una variabile in Python è un riferimento a un oggetto in memoria. Quando creiamo una lista, questa viene allocata in memoria e la variabile contiene solo un riferimento (*puntatore)* a quella posizione.

Nota: Le liste in Python sono oggetti mutabili. Il loro contenuto può essere modificato dopo la crezione.

Fase 2: Creazione di copia1 con metodo copy()

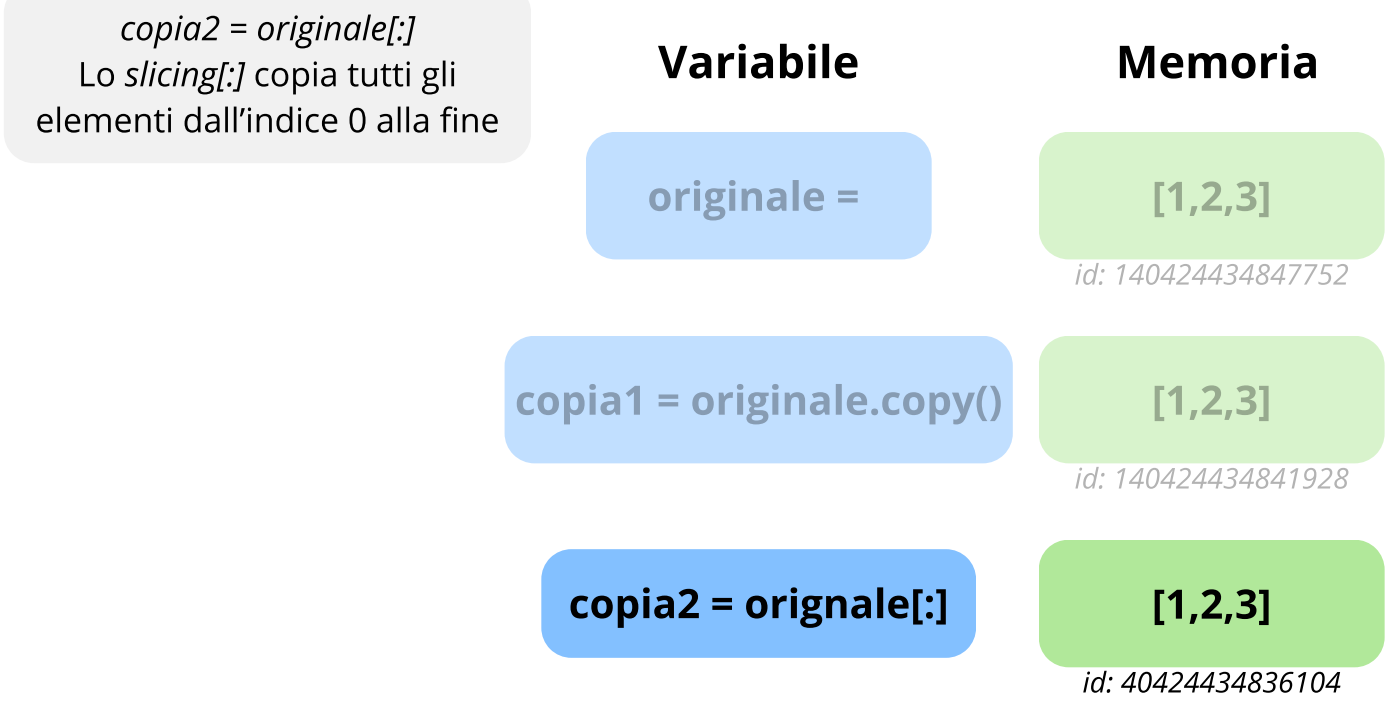
copia1 = originale.copy()
.copy() crea una nuova lista con gli stessi elementi originale



Il metodo *copy()* crea una nuova lista in memoria con gli stessi elementi dell'originale. Gli **ID** diversi confermano che *originale* e *copia1* sono oggetti distinti.

Fase 3: Creazione di copia2 con metodo slicing [:]

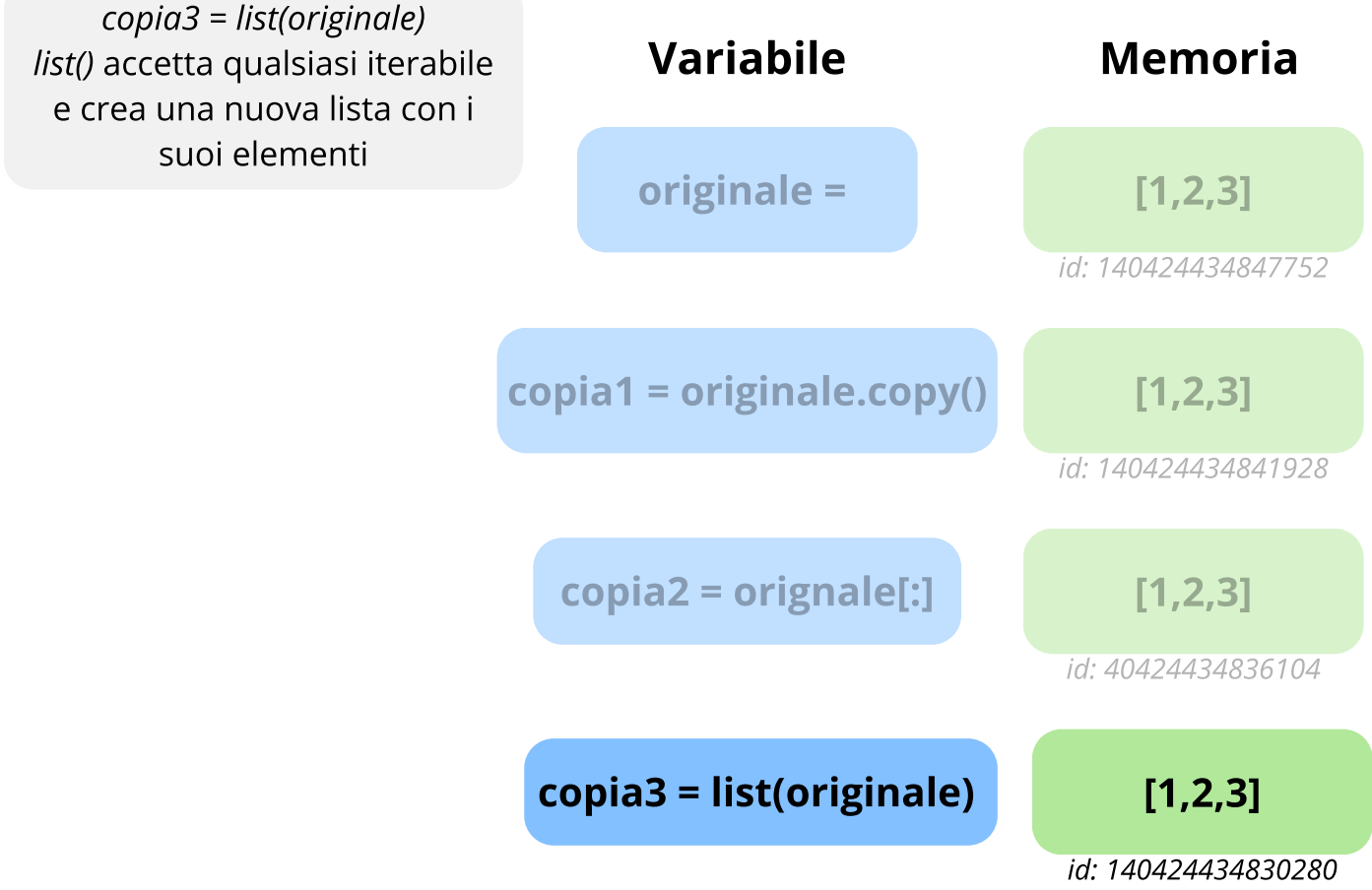
copia2 = originale[:]
Lo *slicing[:]* copia tutti gli elementi dall'indice 0 alla fine



Lo *slicing[:]* è un altro metodo per creare una copia della lista. Come *copy(),* crea un nuovo oggetto lista indipendente dall'originale con gli stessi valori.

Fase 4: Creazione di copia3 con costruttore list()

copia3 = list(originale)
list() accetta qualsiasi iterabile e crea una nuova lista con i suoi elementi



Il costruttore list() crea una copia indipendente della lista. A differenza di .copy() che è un metodo specifico delle liste, list() può trasformare qualsiasi oggetto iterabile (come tuple, set, stringhe, generatori) in una nuova lista

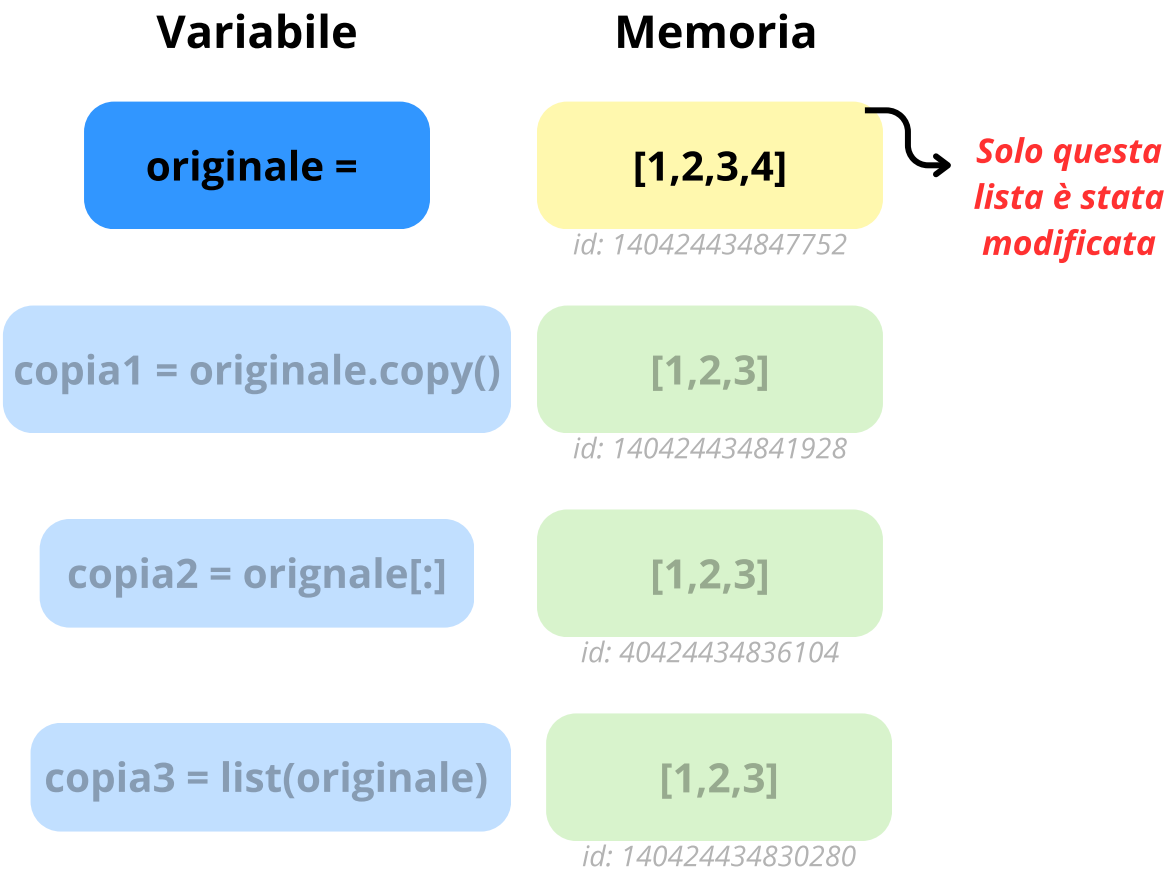
Confronto tra i metodi di copia:

Metodo	Sitnassi	Casi d'uso/Note
.copy()	*nomelista.copy()*	Esplicito, solo Python 3+
Slicing *[:]*	*nomelista[:]*	Idiomatico, tutte le versioni
list()	*list(nomelista)*	Utile per convertire iterabili

Nota importante: Tutti e tre i metodi creano copie superficiali (shallow copies). Questo è sufficiente per liste con elementi immutabili (numeri, stringhe, tuple), ma può causare problemi con elementi mutabili annidati (altre liste, dizionari).

Fase 5: Effetto delle modifiche sulla lista originale

originale.append(4)
Abbiamo modificato la lista originale aggiungendo l'elemento 4.



La modifica della lista originale con ***apppend()*** influisce solo su quella lista. Le altre copie rimangono invariate, dimostrando che tutte le tecniche di copia creano oggetti indipendenti della lista ***originale***