TIPI IN PYTHON
Python è dinamicamente tipato e questo permette di non determinare il tipo di una variabile quando la creiamo.
Quando facciamo un assegnamento del tipo :

A = 42

Succedono 3 cose :
- Creazione dell'oggetto che conterrà 42, se non esiste già;
- Creazione dell'etichetta, se non esiste già;
- Creazione del link che collega l'etichetta A al suo contenuto.

In python il tipo della variabile viene assegnato in base al contenuto dell'oggetto collegato all'etichetta A, e non quindi al suo contenuto.
Perciò, è importante dire che quando facciamo un assegnamento stiamo assegnando l'oggetto all'etichetta e non il contrario.

COSA SUCCEDE QUANDO GLI VIENE RIASSEGNATO UN VALORE :
In quel momento il nostro 42 viene ' garbage collected' finchè nessun'altro si riferirà a 42.
Gli interi piccoli non verranno mai rimossi perchè vengono utilizzati dal sistema mentre nel caso 'spam' verrà sicuramente cancellato.
Quindi il garbage collector interviene quando il reference counter = 0 e va a ripulire lo spazio di memoria.
Quindi 2 variabili che avranno lo stesso valore saranno linkati allo stesso spazio di memoria.

```
a = 42
b = a
a = 'spam'
```

Se stampo b sarà uguale a 42 perchè nel momento in cui vado a cambiare il contenuto di A, il 42 non viene eliminato dal garbage collector perchè esiste ancora B che gli punta.
Infatti 42 sarà puntato da 1 etichetta e verrà creato un nuovo spazio di memoria che conterrà 'spam' linkato ad a
E' sempre cosi?

```
a = [ 1, 2, 3]
b = a
b[1] = 'spam'
```

Quindi se stampo b otterrò -> [1, 'spam' , 3]
Se stampo a?
Se stampo a otterò anche qui -> [1, 'spam' , 3] perchè a è linkato allo spazio di memoria che contiene i link agli oggetti dell'array.
Quando creo b creo un etichetta che punta alla stessa cosa appena citata.
Quindi vado a modificare b, a non cambia puntatore ma è sempre attaccato alla stessa lista che però è mutata.

MODI DI VERIFICARE UGUAGLIANZA
- == -> Uguaglianza del contenuto
- Is -> Uguaglianza tra oggetti

```
l = [1,2,3]
m = [1,2,3]
n = l

l == m      # True
l is m      # false - sono due oggetti diversi
l == n      # True
l is n      # True
```

Ma...

x = 42
y = 42
x == y # True
x is y # True -> Questa cosa è strana perchè per il discorso appena fatto mi aspetto che siano due oggetti diversi
Questo perchè l'idea di fondo che i numeri sono oggetti si basa sul fatto che i numeri sono immutabili
quindi qualsiasi operazione che faccio sui numeri non modificano l'oggetto ma ne creano un altro.
Questa tipologia di oggetti quindi sono Singleton al contrario delle liste.

I piccoli interi e altre costanti sono cached per il motivo che abbiamo detto prima.
Infatti, se faccessimo :

sys.getrefcount(42) # restituisce 10
sys.getrefcount([1,2,3]) # restituisce 1

PASSARE ARGOMENTI A METODI
Gli argomenti di una funzione sono passati per valore tranne nel caso delle liste.