INDICE

Operatore di assegnamento

Secondo la documentazione ufficiale:
Assignment statements are used to (re)bind names to values and to modify attributes or items of mutable objects.

in altre parole funziona cosi':

I. l'espressione sul lato destro dell' = viene valutata;
II. l'oggetto corrispondente all'espressione viene creato/modificato da qualche parte, nella memoria del calcolatore;
III. il nome sul lato sinistro e' assegnato all'oggetto del punto precedente.

In [1]:
```
# creo una lista (a destra dell =) e gli associo il nome a
a = [1,2,3]
```

Operatore ***is***

Da quanto detto sopra a proposito dell'operatore di assegnamento, deriva che possiamo attribuire nomi diversi allo stesso oggetto e quindi e' necessario disporre di strumenti che ci permettano di evidenziare tale occorrenza.

Ogni oggetto che istanziamo ha un suo id univoco (che nel caso della versione di Python utilizzata e' l'indirizzo fisico della porzione di memoria occupata dall'oggetto stesso).
L'operatore is fa un controllo di uguaglianza tra l'id di due oggetti:

In [2]:
```
a = [1,2,3]
#nomi diversi ma lo stesso oggetto
b = a
#verifica
a is b
```

Out[2]:
```
True
```

In [3]:
```
#oggetti uguali ma distinti
a = [1,2,3]
b = [1,2,3]
#test di uguaglianza
print(a==b)
#test di coincidenza dell'oggetto
print(a is b)
```

```
True
False
```

id degli oggetti

Come detto sopra, ogni oggetto che istanziamo ha un suo id univoco. La funzione builtin id restituisce l'id di un oggetto:

In [4]:
```
a=[1,2,3]
b=a
#stesso oggetto, stesso id
print(id(a))
print(id(b))
```

```
46320936
46320936
```

In [5]:
```
a=[1,2,3]
b=[1,2,3]
#oggetti diversi (anche se con lo stesso contenuto), id diversi
print(id(a))
print(id(b))
```

```
46320776
46320456
```

Dymamic typing

vedi presentazione disponibile al seguente link presentazione sul dynamic typing