

11 – List Comprehensions

Le **list comprehension** sono una delle caratteristiche più eleganti e potenti di Python. Permettono di creare **nuove liste** in modo compatto, leggibile e spesso più efficiente rispetto all'uso di cicli `for` tradizionali.

Si possono usare anche **condizioni** e **trasformazioni** sugli elementi.

Sintassi base

```
[espressione for elemento in iterabile]
```

- `espressione` : cosa vuoi inserire nella nuova lista.
- `elemento` : variabile temporanea che rappresenta l'elemento corrente.
- `iterabile` : lista, tupla, stringa o qualsiasi oggetto iterabile.

Esempio semplice

Creare una lista dei quadrati dei numeri da 1 a 5:

```
quadrati = [x**2 for x in range(1, 6)]  
print(quadrati)  # [1, 4, 9, 16, 25]
```

Con condizione

Puoi aggiungere un `if` per filtrare gli elementi:

```
numeri = [1, 2, 3, 4, 5, 6]  
pari = [x for x in numeri if x % 2 == 0]  
print(pari) # [2, 4, 6]
```

Con espressione complessa

Puoi applicare trasformazioni più articolate:

```
nomi = ["Anna", "Luca", "Marco"]  
nomi_upper = [nome.upper() for nome in nomi]  
print(nomi_upper)  # ['ANNA', 'LUCA', 'MARCO']
```

List comprehension annidate

Per liste multidimensionali:

```
matrice = [[1, 2, 3], [4, 5, 6]]  
piatta = [num for riga in matrice for num in riga]  
print(piatta) # [1, 2, 3, 4, 5, 6]
```

Vantaggi

- Codice compatto e leggibile.
- Permette di filtrare e trasformare in un'unica espressione.
- Più efficiente in termini di tempo rispetto ai cicli `for` tradizionali per liste grandi.

Nota

List comprehension sono pensate per creare **liste**. Per generare altri tipi di sequenze, come generatori, si usano **generator expressions** (vedi paragrafo 21).

Vuoi procedere con il **paragrafo 12 – Zip Function?**