3 – List Unpacking

Il **list unpacking** è una caratteristica potente e molto elegante di Python che permette di estrarre uno o più elementi da una lista (o da qualsiasi sequenza) e assegnarli direttamente a variabili in un’unica istruzione.
Risulta particolarmente utile quando si lavora con strutture dati di dimensione nota oppure quando si vogliono separare parte degli elementi dal resto.

Unpacking di base

Se la lista contiene lo stesso numero di elementi delle variabili, l'unpacking è diretto:

```
numeri = [10, 20, 30]

a, b, c = numeri

print(a, b, c)   # 10 20 30
```

È un'alternativa più elegante rispetto a:

```
a = numeri[0]
b = numeri[1]
c = numeri[2]
```

Unpacking con il resto degli elementi

Python permette di raccogliere un sottoinsieme della lista in un'unica variabile usando l'operatore `*`.

```
numeri = [1, 2, 3, 4, 5]

a, b, *c = numeri
print(a)   # 1
print(b)   # 2
print(c)   # [3, 4, 5]
```

Puoi posizionare la variabile con `*` anche al centro o alla fine:

```
*a, b = numeri      # a = [1,2,3,4], b = 5
a, *b, c = numeri   # b = [2,3,4]
```

Ignorare alcuni valori

Spesso non interessano tutti gli elementi della lista.

In questi casi, per convenzione, si usa `_` come variabile "segnaposto".

```
x, _, y = [10, 20, 30]
print(x, y)   # 10 30
```

Unpacking di liste annidate

Puoi combinare l’unpacking con strutture più complesse:

```
persona = ["Mario", "Rossi", [1990, 5, 12]]

nome, cognome, (anno, mese, giorno) = persona
```

Unpacking in funzioni

Molto utile anche per passare elementi di una lista come parametri di una funzione:

```
def somma(a, b, c):
    return a + b + c

numeri = [1, 2, 3]
print(somma(*numeri))   # 6
```

Quando usare l’unpacking

- Quando vuoi estrarre rapidamente valori da una lista.
- Quando vuoi separare “testa” e “resto” di una sequenza.
- Quando vuoi evitare assegnazioni verbose.
- Perfetto per funzioni che accettano numero variabile di parametri.

Vuoi procedere con il **paragrafo 4 – Looping over Lists**?