7 – Sorting Lists

Ordinare le liste è una delle operazioni più comuni in Python. Python offre diversi modi per **ordinare elementi** in modo semplice e flessibile, sia **in-place** (modificando la lista originale) sia creando **una nuova lista ordinata**.

Ordinamento in-place con `sort()`

Il metodo `sort()` ordina la lista **modificando direttamente l'originale**:

```
numeri = [30, 10, 20, 40]
numeri.sort()
print(numeri)  # [10, 20, 30, 40]
```

Ordine inverso

Puoi ordinare in ordine decrescente usando il parametro `reverse=True`:

```
numeri.sort(reverse=True)
print(numeri)  # [40, 30, 20, 10]
```

Ordinamento con una funzione chiave

Puoi specificare una funzione per ordinare secondo criteri personalizzati tramite il parametro `key`:

```
parole = ["gatto", "cane", "elefante", "topo"]

# ordinamento per lunghezza
parole.sort(key=len)
print(parole)  # ['cane', 'gatto', 'topo', 'elefante']
```

Creare una nuova lista ordinata con `sorted()`

`sorted()` restituisce **una nuova lista** ordinata senza modificare l'originale:

```
numeri = [30, 10, 20, 40]
ordinati = sorted(numeri)
print(ordinati)  # [10, 20, 30, 40]
print(numeri)    # [30, 10, 20, 40] -> originale intatto
```

Puoi usare anche `reverse=True` e `key` con `sorted()`:

```
ordinati = sorted(numeri, reverse=True)
```

Ordinamento di liste complesse

Se la lista contiene tuple o oggetti, puoi ordinare secondo uno specifico campo:

```
studenti = [("Anna", 24), ("Luca", 20), ("Marco", 22)]

# ordinamento per età
studenti.sort(key=lambda s: s[1])
print(studenti)  # [('Luca', 20), ('Marco', 22), ('Anna', 24)]
```

Riassunto

- `sort()` ordina la lista originale, `sorted()` crea una nuova lista ordinata.
- `reverse=True` permette l'ordinamento decrescente.
- `key` permette di ordinare secondo criteri personalizzati.
- Le liste di oggetti o tuple possono essere ordinate usando funzioni lambda o funzioni definite dall'utente.

Vuoi procedere con il **paragrafo 8 – Lambda Functions**?