En Python existen tres estructuras de datos muy comunes que a veces se confunden: **listas**, **tuplas** y **conjuntos**:

1.Lista (`list`)

- **Sintaxis**: `[1, 2, 3]`
- **Ordenada**: mantiene el orden en que se insertan los elementos.
- **Mutable**: se pueden modificar (añadir, quitar o cambiar elementos).
- **Permite duplicados**: `[1, 1, 2]` es válido.
- **Uso típico**: colecciones de datos que pueden cambiar.ç
- **Los elementos pueden ser de distintos tipos**. Es decir podemos tener una lista y que uno de sus elementos sea un diccionario y otro elemento sea una cadena y otro un conjunto.

Ejemplo:

```
mi_lista = [1, 2, 3]
mi_lista.append(4)      # [1, 2, 3, 4]
mi_lista[0] = 10        # [10, 2, 3, 4]
```

2.Tupla (`tuple`)

- **Sintaxis**: `(1, 2, 3)`
- **Ordenada**: también conserva el orden de los elementos.
- **Inmutable**: no se puede modificar una vez creada (no puedes añadir, quitar ni cambiar elementos).
- **Permite duplicados**.
- **Uso típico**: cuando quieres que los datos no cambien (ej. coordenadas, configuraciones).

Ejemplo:

```
mi_tupla = (1, 2, 3)
print(mi_tupla[0])      # 1
# mi_tupla[0] = 10 → ERROR (no se puede modificar)
```

3. Conjunto (`set`)

- **Sintaxis**: `{1, 2, 3}`
- **No ordenado**: no garantiza el orden de los elementos.

- **Mutable**: se pueden añadir o eliminar elementos, pero **no se pueden cambiar** directamente.
- **No permite duplicados**: si metes dos valores iguales, se queda solo uno.
- **Uso típico**: cuando quieres almacenar elementos únicos y trabajar con operaciones matemáticas de conjuntos (unión, intersección, diferencia).

Ejemplo:

```
mi_conjunto = {1, 2, 3}
mi_conjunto.add(2)      # {1, 2, 3} → no se duplica
mi_conjunto.add(4)      # {1, 2, 3, 4}
```

Resumen

- **Lista**: ordenada, mutable, permite duplicados.
- **Tupla**: ordenada, inmutable, permite duplicados.
- **Conjunto**: no ordenado, mutable, sin duplicados.