1. **Tuplas:**
   Las tuplas son secuencias inmutables de elementos. Se definen utilizando paréntesis ( ). Se utilizan cuando necesitas un conjunto de elementos que no cambiarán a lo largo del tiempo.
   Ejemplo de definición:

```
mi_tupla = (1, 2, 3, 4, 5)
```

   O así

```
mi_tupla = 1, 2, 3, 4, 5
```

   o también desde otro tipo de estructuras como las listas, se pueden generar tuplas:

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

tupla_desde_lista = tuple(lista)
```

   Métodos principales:
   - **index(valor):** Devuelve el índice de la primera aparición del valor especificado.
   - **count(valor):** Devuelve el número de veces que aparece un valor en la tupla.

   Recorrido:

```
mi_tupla = (1, 2, 3, 4, 5)
for elemento in mi_tupla:
    print(elemento)

'''
En este caso, enumerate(mi_tupla) devuelve un iterable que produce tuplas de
la forma (índice, valor) para cada elemento en mi_tupla, y luego el bucle for
desempaqueta estas tuplas en las variables indice y elemento,
respectivamente.
'''
mi_tupla = (1, 2, 3, 4, 5)
for indice, elemento in enumerate(mi_tupla):
    print("El elemento en el índice", indice, "es:", elemento)
```

2. **Listas:**
   Las listas son secuencias mutables de elementos. Se definen utilizando corchetes [ ]. Se utilizan para almacenar colecciones de elementos que pueden cambiar a lo largo del tiempo.

   Ejemplo de definición: `mi_lista = [1, 2, 3, 4, 5]`
   Métodos principales:
   - **append(elemento):** Agrega un elemento al final de la lista.
   - **insert(posición, elemento):** Inserta un elemento en la posición especificada.
   - **remove(valor):** Elimina la primera aparición del valor especificado.
   - **pop([índice]):** Elimina y devuelve el elemento en la posición especificada (o el último si no se especifica ninguna posición).
   - **index(valor):** Devuelve el índice de la primera aparición del valor especificado.

- **count(valor):** Devuelve el número de veces que aparece un valor en la lista.
- **sort():** Ordena la lista de forma ascendente.
- **reverse():** Invierte el orden de los elementos en la lista.

3. **Pilas (Stacks):**
   Una pila es una estructura de datos de tipo LIFO (Last In, First Out), lo que significa que el último elemento añadido es el primero en ser eliminado.
   Puedes implementar pilas utilizando una lista en Python.
   Métodos principales:
   - **append(elemento):** Añade un elemento a la pila (equivale a push).
   - **pop():** Elimina y devuelve el elemento en la cima de la pila (equivale a pop).

4. **Diccionarios:**
   Los diccionarios son colecciones de pares clave-valor, donde cada clave está asociada a un valor. Se definen utilizando llaves { }.
   Se utilizan para representar relaciones entre datos donde cada elemento tiene una etiqueta única.
   Ejemplo de definición: `mi_diccionario = {'clave1': valor1, 'clave2': valor2}`
   Métodos principales:
   - **keys():** Devuelve una vista de todas las claves en el diccionario.
   - **values():** Devuelve una vista de todos los valores en el diccionario.
   - **items():** Devuelve una vista de tuplas que contienen las claves y los valores de cada par en el diccionario.
   - **get(clave, valor_predeterminado):** Devuelve el valor asociado con la clave especificada, o un valor predeterminado si la clave no está presente.
   - **update(diccionario):** Actualiza el diccionario con los pares clave-valor de otro diccionario.
   - **pop(clave[, valor_predeterminado]):** Elimina la clave especificada y devuelve su valor, o devuelve un valor predeterminado si la clave no está presente.