Ordenando Arreglos: El Método .sort()

Materia: Algoritmos y Programación

Prof. D.Sc. BARSEKH-ONJI Aboud

Facultad de Ingeniería
Universidad Anáhuac México

7 de noviembre de 2025

Agenda

1 Fundamentos de `.sort()`

2 Ordenamiento Avanzado con `key`

3 Sort en Arreglos 2D (Matrices)

4 Práctica

¿Qué es Ordenar?

Ordenar una Lista

'Ordenar' es el proceso algorítmico de reorganizar los elementos de una lista (o arreglo) para que sigan un criterio específico, comúnmente numérico (de menor a mayor) o alfabético.

¿Qué es Ordenar?

Diferencia Clave: `.sort()` vs. `sorted()`

Python tiene dos formas de ordenar, y es crucial entender la diferencia:

- `mi_lista.sort()` (Método):
  - **Modifica la lista original** (la ordena 'in-place').
  - No devuelve nada (retorna `None`).
- `nueva_lista = sorted(mi_lista)` (Función):
  - **Devuelve una nueva lista** ordenada.
  - La lista original no sufre ningún cambio.

Hoy nos enfocaremos en el método `.sort()` de las listas.

Sintaxis Básica: `mi_lista.sort()`

Comportamiento por Defecto

Por defecto, `.sort()` ordena la lista de forma **ascendente** (de menor a mayor, o alfabéticamente).

Sintaxis Básica: `mi_lista.sort()`

Ejemplo 1: Números

```
numeros = [5, 1, 10, -2, 8]

# Ordenamos la lista
numeros.sort()

print(numeros)
# Imprime: [-2, 1, 5, 8, 10]

```

Ejemplo 2: Strings

```
palabras = ['Zebra', 'Manzana', '
    Auto']

# Ordenamos alfabeticamente
palabras.sort()

print(palabras)
# Imprime: ['Auto', 'Manzana', '
    Zebra']

```

Parámetro 1: `reverse`

Sintaxis

```
mi_lista.sort(reverse=False)
```

Orden Descendente

El método `.sort()` acepta un parámetro opcional llamado `reverse`.

- Por defecto, `reverse` es `False` (orden ascendente).
- Si lo establecemos a `True`, la lista se ordenará de forma **descendente** (de mayor a menor).

Parámetro 1: reverse

Ejemplo de Uso

```
numeros = [5, 1, 10, -2, 8]

# Ordenamos en reversa (descendente)
numeros.sort(reverse=True)

print(numeros)
# Imprime: [10, 8, 5, 1, -2]

```

Agenda

1 Fundamentos de .sort()

2 Ordenamiento Avanzado con key

3 Sort en Arreglos 2D (Matrices)

4 Práctica

Parámetro 2: key

Sintaxis

```
mi_lista.sort(key=None, reverse=False)
```

El Parámetro Más Poderoso: key

El parámetro key nos permite cambiar **cómo** Python compara los elementos.

- La key (llave) debe ser una **función**.
- Python aplicará esta función a **cada elemento** de la lista *antes* de compararlos.
- El ordenamiento se basará en el **resultado** de esa función, no en el elemento original.

Parámetro 2: key

Analogía: Ordenar Libros

Imagina que tienes una pila de libros. ¿Cómo los ordenas?

- **Sin `key` (por defecto):** Los ordenas por título (alfabéticamente).
- **Con `key=obtener_num_paginas`:** Los ordenas por su número de páginas (de menor a mayor).
- **Con `key=obtener_apellido_autor`:** Los ordenas alfabéticamente por el apellido del autor.

La `key` es la 'llave' o 'criterio' que usamos para decidir el orden.

Ejemplo de key: Ordenar por Longitud

Usando la Función len()

Queremos ordenar una lista de palabras, no alfabéticamente, sino por su **longitud** (de la más corta a la más larga).
Podemos usar la función incorporada `len()` (que devuelve la longitud de un string) como nuestra `key`.

Ejemplo de key: Ordenar por Longitud

Código de Ejemplo

```
palabras = ['Gato', 'Mariposa', 'Pez', 'Elefante']

# Le decimos a .sort() que use la funcion 'len'
# como el criterio de ordenamiento.
palabras.sort(key=len)

print(palabras)
# Imprime: ['Pez', 'Gato', 'Mariposa', 'Elefante']
# (3, 4, 8, 8) <-- OJO: 'Mariposa' y 'Elefante'
# tienen la misma longitud (8), por lo que
# mantienen su orden relativo original (orden estable).

```

Ejemplo de key: Ignorar Mayúsculas

El Problema del Orden ASCII

Por defecto, Python ordena las letras mayúsculas **antes** que las minúsculas (orden ASCII). Esto lleva a resultados confusos:

```
palabras = ['Zebra', 'manzana', 'Auto', 'elefante']

palabras.sort() # Orden por defecto (ASCII)
print(palabras)
# Imprime: ['Auto', 'Zebra', 'elefante', 'manzana']
# (A, Z, e, m) <-- ¡Incorrecto!

```

Ejemplo de key: Ignorar Mayúsculas

La Solución: key=str.lower

Podemos pasar el método `str.lower` como la `key`. Esto le dice a Python que, *solo para propósitos de comparación*, trate a todas las palabras como si estuvieran en minúsculas.

```
palabras = ['Zebra', 'manzana', 'Auto', 'elefante']

# Orden correcto, ignorando mayusculas
palabras.sort(key=str.lower)

print(palabras)
# Imprime: ['Auto', 'elefante', 'manzana', 'Zebra']
# (Compara: 'auto', 'elefante', 'manzana', 'zebra')

```

Agenda

1 Fundamentos de .sort()

2 Ordenamiento Avanzado con key

3 Sort en Arreglos 2D (Matrices)

4 Práctica

Agenda

1 Fundamentos de .sort()

2 Ordenamiento Avanzado con key

3 Sort en Arreglos 2D (Matrices)

4 Práctica

Tarea: Ordenando Datos Complejos I

El Problema

Tenemos una lista de estudiantes. Cada estudiante es, a su vez, una lista que contiene su nombre, su calificación y su número de faltas:
`[Nombre, Calificación, Faltas]`

Tarea: Ordenando Datos Complejos I

Estructura de Datos

```
estudiantes = [
    ['Ana', 85, 2],
    ['Luis', 92, 0],
    ['Carla', 92, 3], # Carla y Luis tienen la misma nota
    ['Bruno', 76, 1]
]

```

Tarea: Ordenando Datos Complejos I

Examples

Tu Tarea

- **Parte 1:** Ordena la lista de estudiantes por **calificación** (de menor a mayor).
- **Parte 2:** Ordena la lista de estudiantes por **número de faltas** (de menor a mayor).

Tarea: Pista (Definir una Función key)

No podemos usar len o str.lower

Necesitamos crear nuestra **propia función** que le diga a `.sort()` qué índice de la sub-lista debe mirar.

Usamos key = lambda

key lambda nos ayuda a ordenar la matriz de acuerdo a una columna específica. `lambda columna: columna[indice]` donde `indice` es la columna que queremos usar para ordenar.

Tarea: Pista (Definir una Función key)

Código de Ejemplo (Parte 1: Por Calificación)

```
estudiantes = [
    ['Ana', 85, 2], ['Luis', 92, 0],
    ['Carla', 92, 3], ['Bruno', 76, 1]
]

estudiantes.sort(key=lambda columna: columna[1])

print('Ordenados por calificacion (con lambda):')
print(estudiantes)

```

Desafío de Programación: El Censo del Distrito

Eres un trabajador del censo y necesitas registrar a los habitantes de un distrito.

- Crear una lista vacía llamada `censo`.
- Iniciar un bucle infinito (`while True`) que pida al usuario: Nombre (string), Edad (int), Código Postal (int)
- **Condición de salida:** Si el usuario escribe `"salir"` en el nombre, el bucle debe romperse (`break`).
- **Almacenamiento:** Guardar cada ciudadano como una lista (ej. `['Ana', 30, 10300]`) dentro de la lista principal `censo`.
- **Reporte Final:** Al terminar el bucle, ordenar el `censo` por **edad**, de la persona **más vieja a la más joven**.
- Imprimir la lista `censo` ya ordenada.