Conceptos básicos y avanzados de ordenación en Python

Conceptos básicos de ordenación

```
# Introducción a la ordenación en Python
numeros = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
print(sorted(numeros))  # Imprime la lista ordenada
```

Invocando .sort y sorted

```
# Uso del método sort() y la función sorted()
numeros = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
numeros.sort()  # Ordena la lista en su lugar
print(numeros)

# Uso de sorted() para ordenar sin modificar la lista original
numeros = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
print(sorted(numeros))
```

Parámetro opcional reverse

```
# Uso del parámetro reverse para ordenar en orden inverso
numeros = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
print(sorted(numeros, reverse=True))  # Ordena en orden descendente
```

Parámetro opcional key

```
# Uso del parámetro key para ordenar por la longitud de las palabras
palabras = ["manzana", "banana", "cereza", "date"]
print(sorted(palabras, key=len))  # Ordena por longitud de las palabras
```

```
# Ordenar una lista de diccionarios por un valor específico
estudiantes = [
    {"nombre": "Juan", "edad": 22},
    {"nombre": "Ana", "edad": 21},
    {"nombre": "Luis", "edad": 24}
]
print(sorted(estudiantes, key=lambda x: x["edad"]))
```

Ordenación de las claves de un diccionario

```
# Ordenar las claves de un diccionario
notas = {"Juan": 88, "Ana": 92, "Luis": 95, "Maria": 85}
claves_ordenadas = sorted(notas.keys())
print(claves_ordenadas)
```

Romper empates

```
# Ordenar una lista de tuplas, rompiendo empates
# Ordenar por el primer elemento, luego por el segundo
tuplas = [(1, 'b'), (1, 'a'), (2, 'c'), (2, 'b')]
print(sorted(tuplas, key=lambda x: (x[0], x[1])))
```