Tecnicatura Universitaria en Programación
Programación II

UNIDAD N° 1: Lenguaje Python - Estabilidad en ordenamientos

Índice

Ordenamiento complejos de Listas

Supongamos que tenemos que ordenar por más de una condición una lista. Para ello necesitamos aplicar ordenamientos secuenciales a la lista hasta obtener el resultado deseado.
Para ello utilizaremos la función **sorted( )** y el método **sort( )** y funciones **lambda**

Cuando Python realiza un proceso de ordenamiento con **sorted( )** o **sort( )**, realiza un ordenamiento estable porque ordena la lista con el algoritmo TimSort, un algoritmo de ordenamiento muy eficiente y estable.
Un algoritmo de ordenación es estable si garantiza que no se cambia el orden relativo que mantienen inicialmente los elementos que se consideran iguales. Esto es útil para realizar ordenaciones en múltiples fases.
Esto significa que si dos elementos tienen el mismo valor o valor intermedio (clave), está garantizado que se mantendrán en el mismo orden relativo entre ellos.

Supongamos que necesitamos ordenar la siguiente lista primero por longitud del nombre de la marca y dentro de la misma longitud alfabéticamente. Para obtener el resultado deseado debemos ordenar en 2 fases. Primero ordenamos alfabéticamente y luego ordenamos por longitud obtenemos el resultado deseado.

```
12  marcas = ['gmc','audi','kia','FORD','BMW','AUDI',]
13
14  marcas.sort(key=lambda x: x.upper())
15  marcas.sort(key=lambda x: len(x))
16  print(marcas)
17
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS D:\TUP\Programacion II\Practica 2\TUP-Python-Practica2> python app.py
['BMW', 'gmc', 'kia', 'audi', 'AUDI', 'FORD']
```

Observe la diferencia de ordenar por una u otra condición únicamente.

```
12  marcas = ['gmc','audi','kia','FORD','BMW','AUDI',]
13
14  marcas.sort(key=lambda x: len(x))
15  print(marcas)
16
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS D:\TUP\Programacion II\Practica 2\TUP-Python-Practica2> python app.py
['gmc', 'kia', 'BMW', 'audi', 'FORD', 'AUDI']
```

```
12  marcas = ['gmc','audi','kia','FORD','BMW','AUDI',]
13
14  marcas.sort(key=lambda x: x.upper())
15  print(marcas)
16
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS D:\TUP\Programacion II\Practica 2\TUP-Python-Practica2> python app.py
['audi', 'AUDI', 'BMW', 'FORD', 'gmc', 'kia']
```

Supongamos que necesitamos ordenar la siguiente lista de números primero de mayor a menor y luego todos los pares primero y luego los impares, de manera que me queden primero los números pares de mayor a menor y luego los impares de mayor a menor. Puedo hacerlo ordenando en 2 fases.

```
19  numeros = [2,1,10,5,7,6,3]
20  lista_ord = sorted(numeros, reverse=True)
21  lista_ord_par = sorted(lista_ord, key=lambda x: x%2 != 0)
22  print(lista_ord_par)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS D:\TUP\Programacion II\Practica 2\TUP-Python-Practica2> python app.py
[10, 6, 2, 7, 5, 3, 1]
```

Bibliografía

https://es.wikipedia.org/wiki/Algoritmo_de_ordenamiento
https://en.wikipedia.org/wiki/Timsort

Versiones

Fecha	Versión
05/09/2023	1.0

Autores

María Mercedes Valoni