PUG\_BPP\_A4

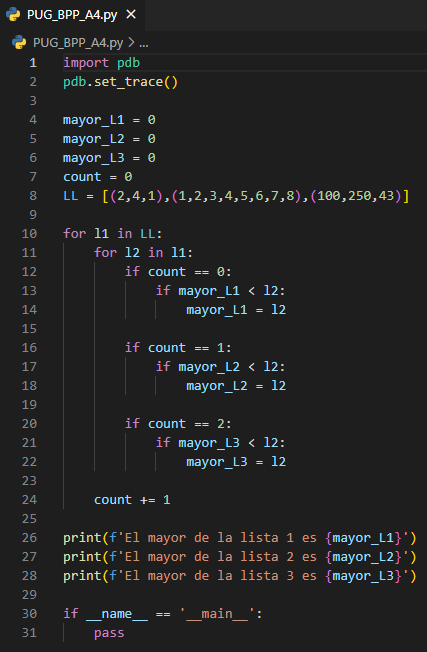
Por

Pablo Ujados García

# Actividad 1

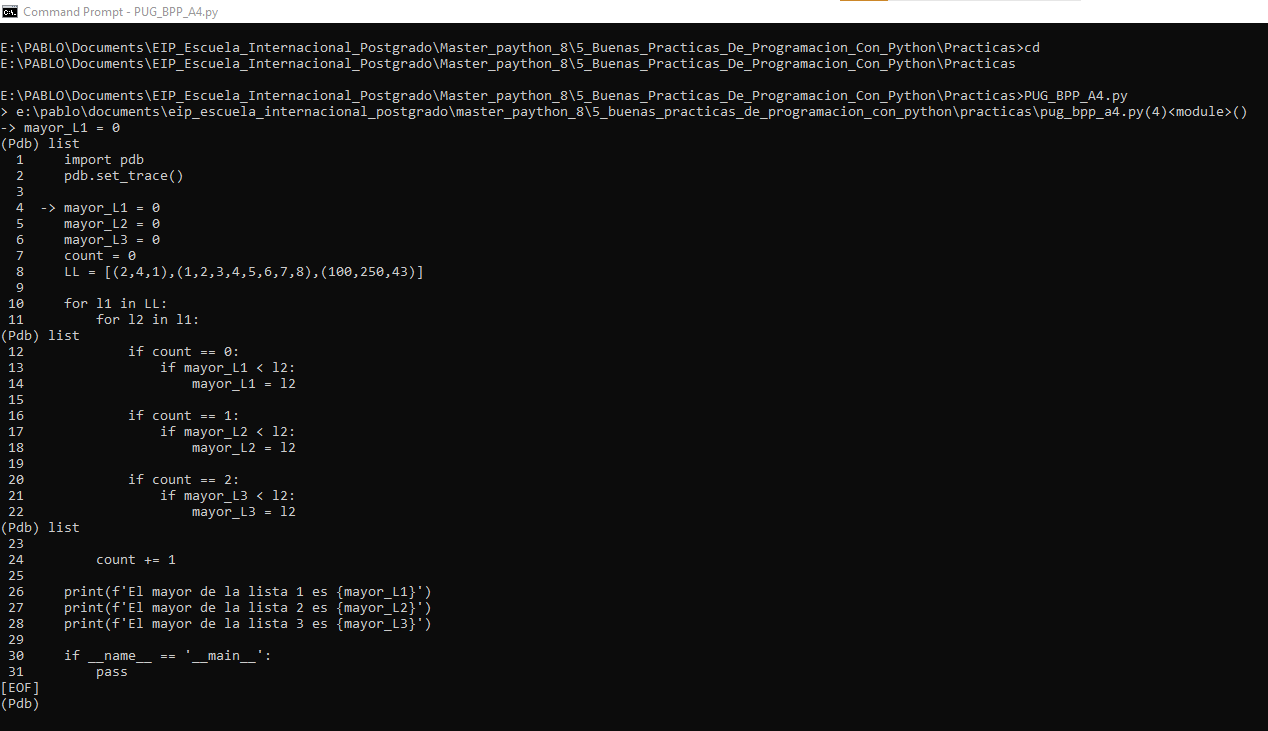
Dada la siguiente lista “[2, 4, 1], [1,2,3,4,5,6,7,8], [100,250,43]]” muestra el mayor de cada lista, depura el programa con PCB y saca conclusiones

Programa que hemos desarrollado

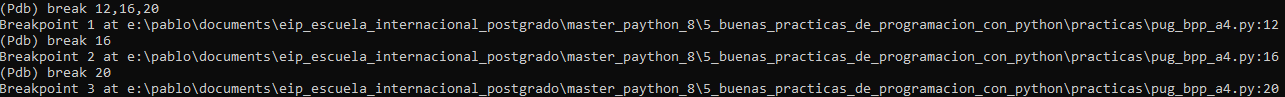


Se ha importado PDB y se ha utilizado la función set\_trace() para poder depurar el programa por consola

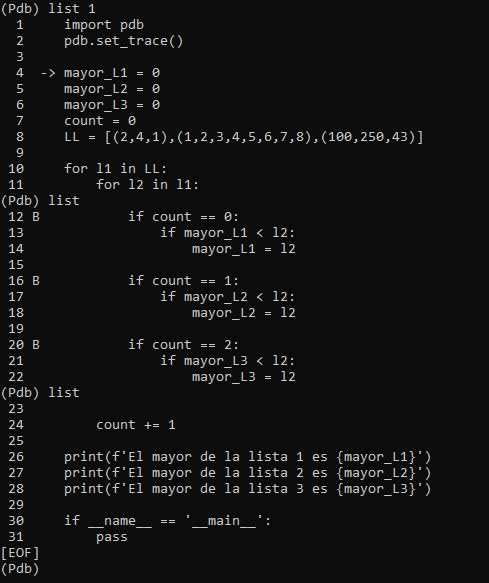
Activamos consola de comando y nos ubicamos en la carpeta de del desarrollo y activamos programa y hacemos el listado del programa



Ponemos los puntos de interrupción en las líneas 12,16 y 20



Listamos de nuevo para verificar los puntos de interrupción



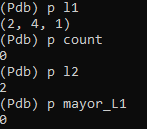
Hacemos una inspección inicial de todas las variables que necesitamos seguir, viendo que tiene los valores iniciales y correctos.

Count = contador de vuelta por la que va sobre la lista de listas

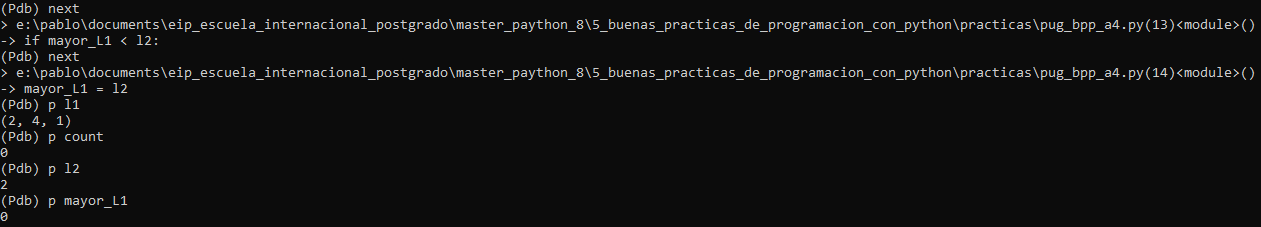
L1 = sub-lista actual de la lista

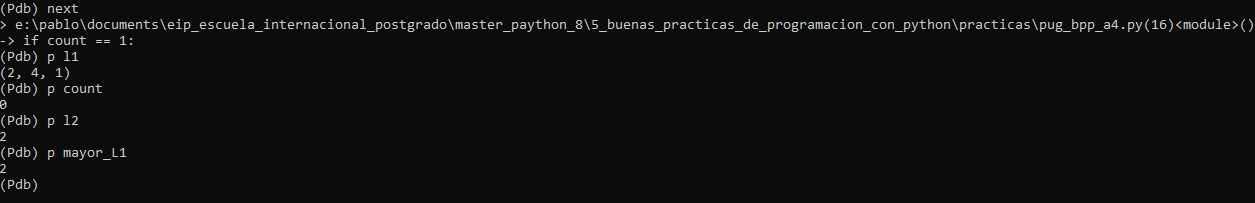
L2 = número cogido de la sub-lista actual

Mayor\_LX = variable en la que se guardara el mayor de cada sub-lista.



Avanzamos a la siguiente línea de ejecución viendo , viendo como entra en la instrucción de comparación de “mayor\_l1” menor que “l2” , al darse esta condición entra y asigna a “mayor\_l1” el valor de “l2”



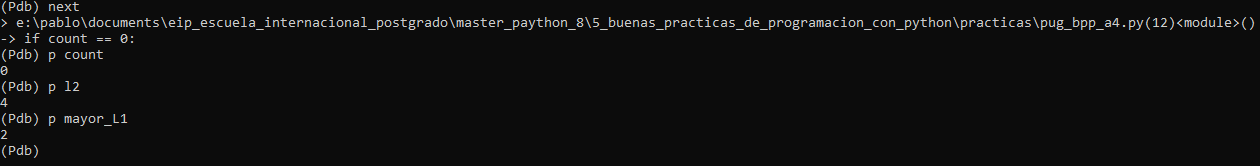


Quedando las variables de esta manera

L2 = 2

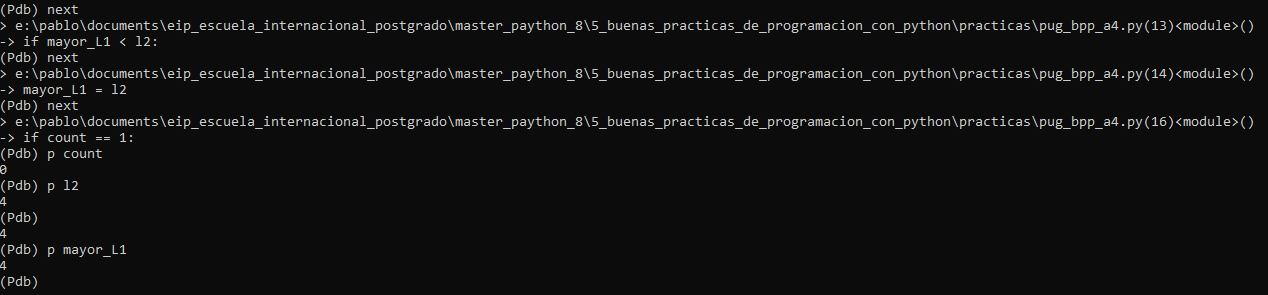
Mayor\_L1 = 2

Continuamos avanzando línea a línea la ejecución, vemos que en la siguiente vuelta del bucle no encontramos que “L2” a cogido el valor 4 y que “mayor\_L1” tiene 2

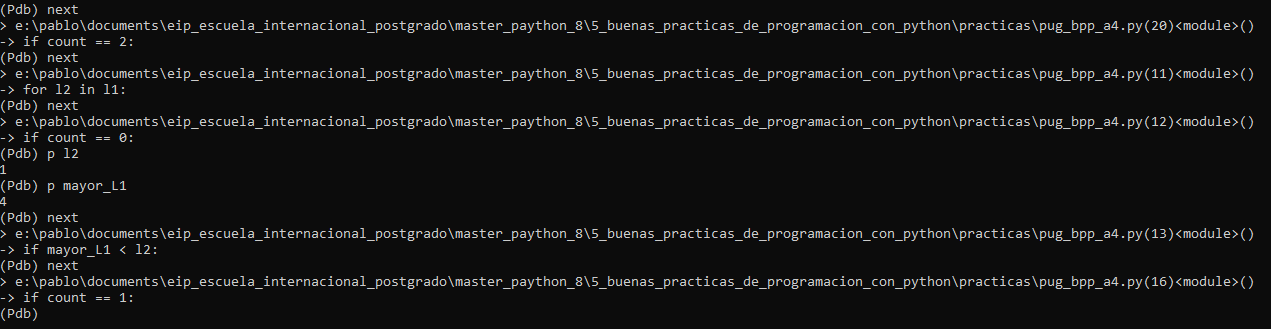


Dado este caso vuelve a entrar en la condición dado que “count” = 0 y “l2” es mayor que “mayor\_l1”

Y asigna el valor de “l2” a “mayor\_l1”

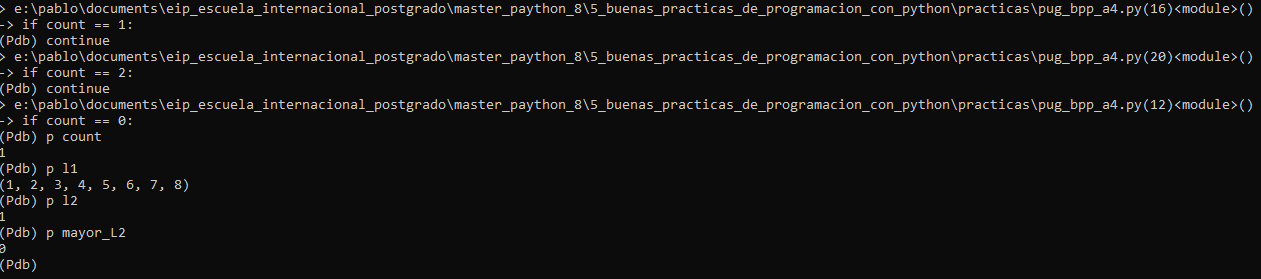


Seguimos avanzando línea a línea la ejecución , hasta encontrarlo en el ultimo numero, en esta ocasión tenemos que “count” = 0 con lo cual eso cumple pero “l2” = 2 y “mayor\_L1” = 4, con lo cual no es mayor que “l2”.

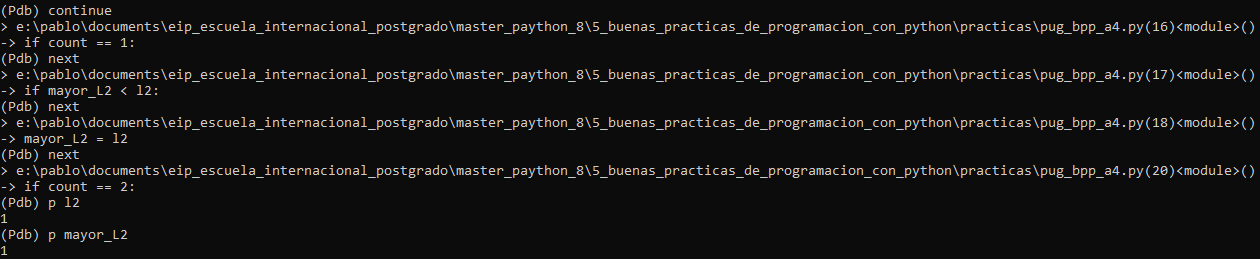


De esta manera, la variable no se asigna, teniendo ya el mayor de la lista, saliendo del sub-bucle y avanzando en el bucle principal a la siguiente sub-lista.

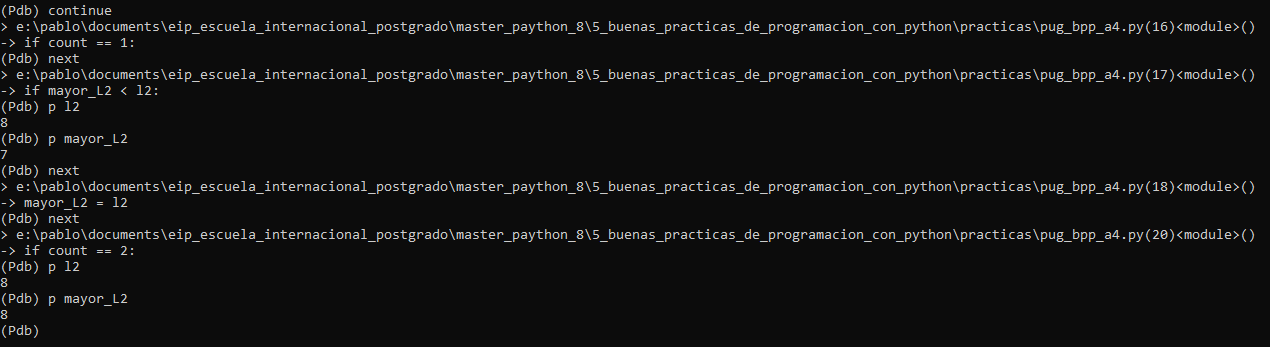
A partir de aquí se ha puesto la primera iteración, para ver el inicio de los datos y el final de la iteración



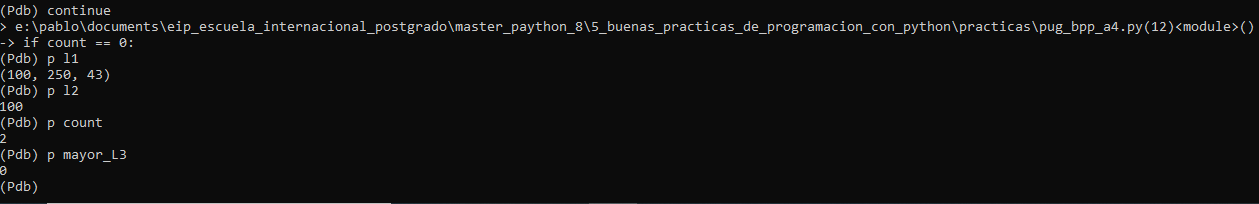
Vemos como avanzando línea a línea ocurre exactamente el mismo proceso que la anterior vez, pero teniendo en cuenta que ahora “count” = 1, y “l2” se compara con “mayor\_l2”



De esta manera llegamos a la última iteración, dando la casualidad de que el ultimo número es el mayor, con lo cual, cumpliendo condición entra en los diferentes condicionales y “mayor\_l2” termina con valor 8

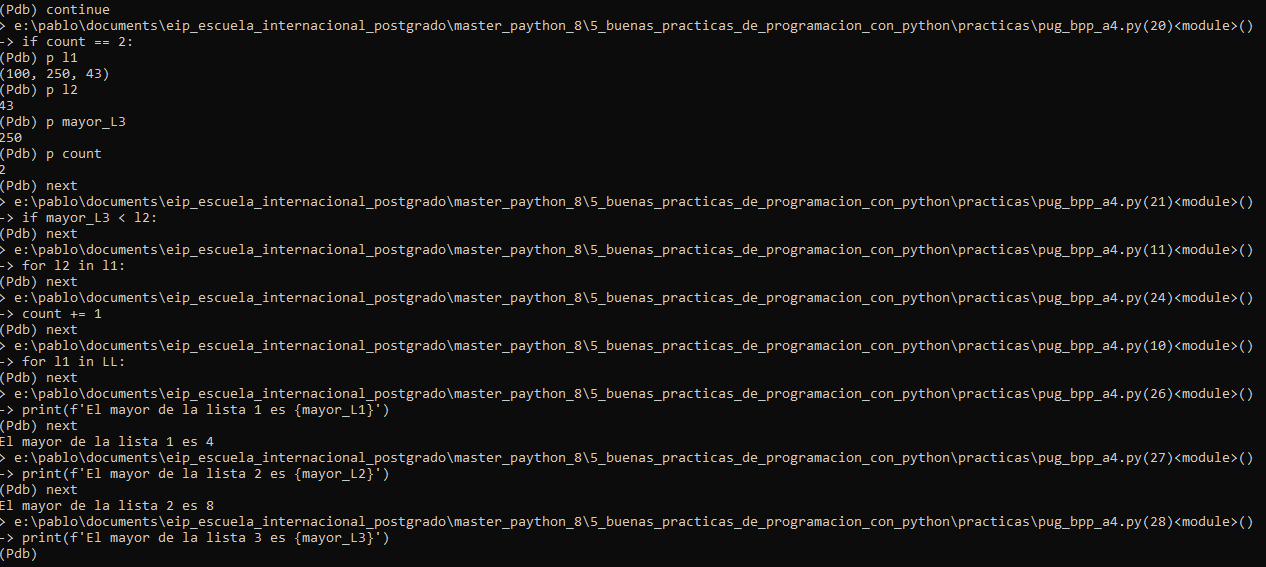


Por ultimo pasamos a la ultimo sub-lista, “count” pasa a tener valor 2, “l2” = 100 y “mayor\_l3” = 0



Dándose la misma casuística que las anteriores, el ejecución va entrado en el condicional con condición “count” = 2 y va viendo que se cumple la condición, hasta encontrarse con la utlima iteración en la cual “l2” = 43 y “mayor\_l3” = 250. Siendo “mayor\_L3” mayor que “l2” no se cumple condición y no se asigna el valor de “l2” a “mayor\_l3”, dado que este ya tiene el mayor de la lista.

El cursor sigue avanzando y vemos como sale del sub-bucle y también es la ultima iteración de bucle principal, saliendo de este también y mostrando por ultimo las correspondientes línea al usuario diciéndole cual es el mayor de cada lista



Conclusiones, hay que tener mucho cuidado tanto con los condicionales que se aplican y los accesos a lista, dado que un mal acceso o una mala condición no pueden llevar a un error de flujo del programa o una lógica errónea.

A nivel del depurado de consola de Python, me ha resultado más fácil de lo que creia, también es cierto que es un código pequeño y habria que ver cómo se maneja esto en un código de producción.