Depuración y reglas de optimización de código.

Autor: Victor Simo Lozano

Actividad 1

Realizar un programa en el que se devuelva el mayor de los numeros de tres listas dentro de una lista. Para ello se ha de emplear la **"compresión de listas".**
Por último, mediante el uso del depurador de python **pdb** se depurará el código de modo que se pueda obtener conclusiones respecto a la asignación de variables en la compresión de listas.

SOLUCION

1.-
En base a la lista ejemplo de la actividad, realizamos la comprensión de listas para extraer el numero mayor de cada sublista mediante la función de python **"max(i)".**

Para el debug, se emplea el modulo pdb del core de python, con este, mediante el uso de la cmd y comandos se podrá navegar entre los puntos de interrupción marcado como "**pdb.set_trace()**"

In [1]:
```
import pdb
pdb.set_trace()

# Declaración de la lista de la actividad.
lista = [[2, 4, 1], [1,2,3,4,5,6,7,8], [100,250,43]]

# Compresion de listas para almacenar en la lista "lista_maximos" el máximo de cada sublista donde:
# - "x" hace referencia a cada sublista de "lista"
# - "lista" será recorrida con el bucle "for x in lista"
# - de cada iteración del bucle se obtiene el máximo valor de la sublista "x" mediante "max(x)"
```

```
pdb.set_trace()
lista_maximos = [max(x) for x in lista]
pdb.set_trace()

# Output
print(f'El valor máximo de la primera sublista es {lista_maximos[0]},'
      f' de la segunda es {lista_maximos[1]} y de la tercera {lista_maximos[2]}.')
```

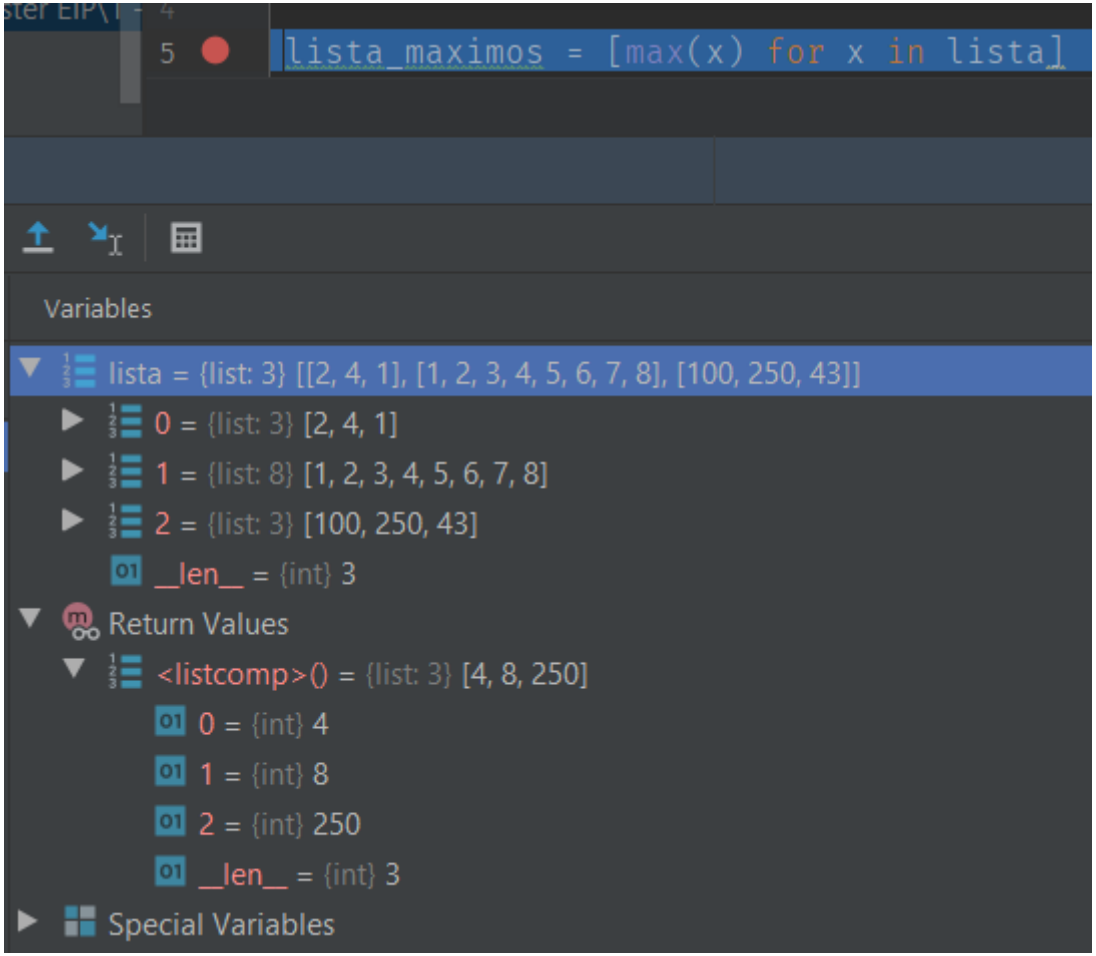
```
--Return--
None
> c:\users\vicsi\appdata\local\temp\ipykernel_9612\4067047672.py(2)<cell line: 2>()

ipdb> c
--Return--
None
> c:\users\vicsi\appdata\local\temp\ipykernel_9612\4067047672.py(12)<cell line: 12>()

ipdb> list
      7 # Compresion de listas para almacenar en la lista "lista_maximos" el máximo de cada sublista donde:
      8 # - "x" hace referencia a cada sublista de "lista"
      9 # - "lista" será recorrida con el bucle "for x in lista"
     10 # - de cada iteración del bucle se obtiene el máximo valor de la sublista "x" mediante "max(x)"
     11
---> 12 pdb.set_trace()
     13 lista_maximos = [max(x) for x in lista]
     14 pdb.set_trace()
     15
     16 # Output
     17 print(f'El valor máximo de la primera sublista es {lista_maximos[0]},'

ipdb> p lista
[[2, 4, 1], [1, 2, 3, 4, 5, 6, 7, 8], [100, 250, 43]]
ipdb> p lista_maximos
*** NameError: name 'lista_maximos' is not defined
ipdb> c
--Return--
None
> c:\users\vicsi\appdata\local\temp\ipykernel_9612\4067047672.py(14)<cell line: 14>()

ipdb> list
      9 # - "lista" será recorrida con el bucle "for x in lista"
     10 # - de cada iteración del bucle se obtiene el máximo valor de la sublista "x" mediante "max(x)"
     11
     12 pdb.set_trace()
     13 lista_maximos = [max(x) for x in lista]
---> 14 pdb.set_trace()
     15
     16 # Output
     17 print(f'El valor máximo de la primera sublista es {lista_maximos[0]},'
     18       f' de la segunda es {lista_maximos[1]} y de la tercera {lista_maximos[2]}.')

ipdb> p lista_maximos
```

```
[4, 8, 250]
ipdb> p x
*** NameError: name 'x' is not defined
ipdb> c
El valor máximo de la primera sublista es 4, de la segunda es 8 y de la tercera 250.
```

Conclusión:

*Al ejecutar el primer breakpoint en la linea 12, esta definida la variable **lista**, pero no obtenemos ningún valor para **lista_maximos**. Para el segundo breakpoint en la linea 14, una vez ejecutado el codigo para la compresion de lista, obtenemos el resultado esperado. Es decir, se observa como la ejecución del codigo es linea a linea.*

*En la imagen adjunta se puede contemplar con mejor detalle el "debugueado" con el propio debug del IDE Pycharm en el que se observa como en una unica linea de codigo se puede observa la variable **lista** y el resultado **lista_maximos** como "listcomp" lo cual nos muestra la máxima eficiencia del codigo.*



Nota:

Se ha hecho uso del debug del ide de Pycharm para poder observar con mejor detalle el funcionamiento del programa. Esta solución se ha decidio dado que durante el "debugueado" con pdb, no se ha podido crear nuevos break points con ***b linea*** *y para pasar de intruccion a instrucción con el comando* ***n*** *se ha visto que la ejecución saltaba al propio codigo fuente de pdb sin volver al codigo principal y no se ha podido ver en detalle la evolución del mismo.*