

# Entrega Lección 4 Buenas Prácticas de Programación en Python

Realizado por Marcos Santana Pastor

En la siguiente captura se muestra el código de la primera tarea de la práctica

```
entrega.py > ...
1  #Buenas Practicas de Programación en Python
2  #Practica 4 realizada por Marcos Santana Pastor
3
4  import pdb
5  pdb.set_trace()
6
7  def max_list (L):
8      M = [max(L[i]) for i in range(0,len(L))]
9      return M
10
11  L = [[2,4,1],[1,2,3,4,5,6,7,8],[100,250,43]]
12  print(max_list(L))
```

Vamos a la terminal y ejecutamos:

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL
PS C:\Users\crist\Desktop\Master\Buenas Practicas\Leccion 4> python entrega.py
> c:\users\crist\desktop\master\buenas practicas\leccion 4\entrega.py(7)<module>()
-> def max_list (L):
(Pdb)
```

Colocamos el break point en la línea 8:

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL
PS C:\Users\crist\Desktop\Master\Buenas Practicas\Leccion 4> python entrega.py
> c:\users\crist\desktop\master\buenas practicas\leccion 4\entrega.py(7)<module>()
-> def max_list (L):
(Pdb) break 8
Breakpoint 1 at c:\users\crist\desktop\master\buenas practicas\leccion 4\entrega.py:8
(Pdb)
```

Hacemos que el programa se ejecute hasta la línea 8. Primero hará todo el código y cuando entre en la función será cuando pare en la línea 8.

Vemos cómo en ese primer momento M no tiene valor porque paró justo antes. Pero también vemos que está haciendo un bucle interno ya que es necesario que le demos a continuar 4 veces más para que termine de crear la lista M y devuelva el valor final.

Es decir, necesitamos varios intentos para ver terminar el programa, en concreto, la línea 8 se ejecuta 4 veces hasta que termina de dar un valor a M.

Entiendo que esto se debe a que internamente está inicializando el iterante y luego con cada iteración damos un nuevo valor a la lista.

Más abajo vemos el código del terminal.

```

> c:\users\cris\desktop\master\buena practicas\leccion 4\entrega.py(7)<module>()
-> def max_list (L):
(Pdb) break 8
Breakpoint 1 at c:\users\cris\desktop\master\buena practicas\leccion 4\entrega.py:8
(Pdb) continue
> c:\users\cris\desktop\master\buena practicas\leccion 4\entrega.py(8)max_list()
-> M = [max(L[i]) for i in range(0,len(L))]
(Pdb) p M
*** NameError: name 'M' is not defined
(Pdb) continue
> c:\users\cris\desktop\master\buena practicas\leccion 4\entrega.py(8)<listcomp>()
-> M = [max(L[i]) for i in range(0,len(L))]
(Pdb) p M
*** NameError: name 'M' is not defined
(Pdb) continue
> c:\users\cris\desktop\master\buena practicas\leccion 4\entrega.py(8)<listcomp>()
-> M = [max(L[i]) for i in range(0,len(L))]
(Pdb) p M
*** NameError: name 'M' is not defined
(Pdb) continue
> c:\users\cris\desktop\master\buena practicas\leccion 4\entrega.py(8)<listcomp>()
-> M = [max(L[i]) for i in range(0,len(L))]
(Pdb) p M
*** NameError: name 'M' is not defined
(Pdb) continue
> c:\users\cris\desktop\master\buena practicas\leccion 4\entrega.py(8)<listcomp>()
-> M = [max(L[i]) for i in range(0,len(L))]
(Pdb) p M
*** NameError: name 'M' is not defined
(Pdb) continue
[4, 8, 250]

```

Utilizando las opciones de depuración de Visual Studio Code

entrega.py > ...

```
1 #Buenas Practicas de Programación en Python
2 #Practica 4 realizada por Marcos Santana Pastor
3
4
5 #1 Parte
6 #import pdb
7 #pdb.set_trace()
8
9 def max_list (L):
10     M = [max(L[i]) for i in range(0,len(L))]
11     return M
12
13 L = [[2,4,1],[1,2,3,4,5,6,7,8],[100,250,43]]
14 print(max_list(L))
15
16 #2 Parte
17
18 def comprobar (x):
19
20     if x == 2:
21         return True
22     else:
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
es\lib\python\debugpy\adapter\...\debugpy\launcher' '55647'
y'
[4, 8, 250]
[3, 5, 13]
PS C:\Users\crist\Desktop\Master\Buenas Practicas\Leccion 4>
\Users\crist\AppData\Local\Programs\Python\Python38\python.exe
es\lib\python\debugpy\adapter\...\debugpy\launcher' '55828'
y'
[]
```

Continuamos el código-> se inicia el objeto iteración

Locals

```
> .0: <range_iterator object at...
> L: [[2, 4, 1], [1, 2, 3, 4, 5...
> Globals
```

INSPECCIÓN

```
1 #Buenas Practicas de Programación en Python
2 #Practica 4 realizada por Marcos Santana Pastor
3
4
5 #1 Parte
6 #import pdb
7 #pdb.set_trace()
8
9 def max_list (L):
10     M = [max(L[i]) for i in range(0,len(L))]
11     return M
```

Empieza a coger la primera lista de la lista-> i=0

```

VARIABLES
  Locals
    .0: <range_iterator object at...
    L: [[2, 4, 1], [1, 2, 3, 4, 5...
    i: 0
  Globals

entrega.py > max_list
1 #Buenas Practicas de Programación en Python
2 #Practica 4 realizada por Marcos Santana Pastor
3
4
5 #1 Parte
6 #import pdb
7 #pdb.set_trace()
8
9 def max_list (L):
10     M = [max(L[i]) for i in range(0,len(L))]
11     return M
12

```

Segunda lista de la lista i =1

```

VARIABLES
  Locals
    .0: <range_iterator object at...
    L: [[2, 4, 1], [1, 2, 3, 4, 5...
    i: 1
  Globals

entrega.py > max_list
1 #Buenas Practicas de Programación en Python
2 #Practica 4 realizada por Marcos Santana Pastor
3
4
5 #1 Parte
6 #import pdb
7 #pdb.set_trace()
8
9 def max_list (L):
10     M = [max(L[i]) for i in range(0,len(L))]
11     return M
12

```

Tercera lista de la lista i = 2

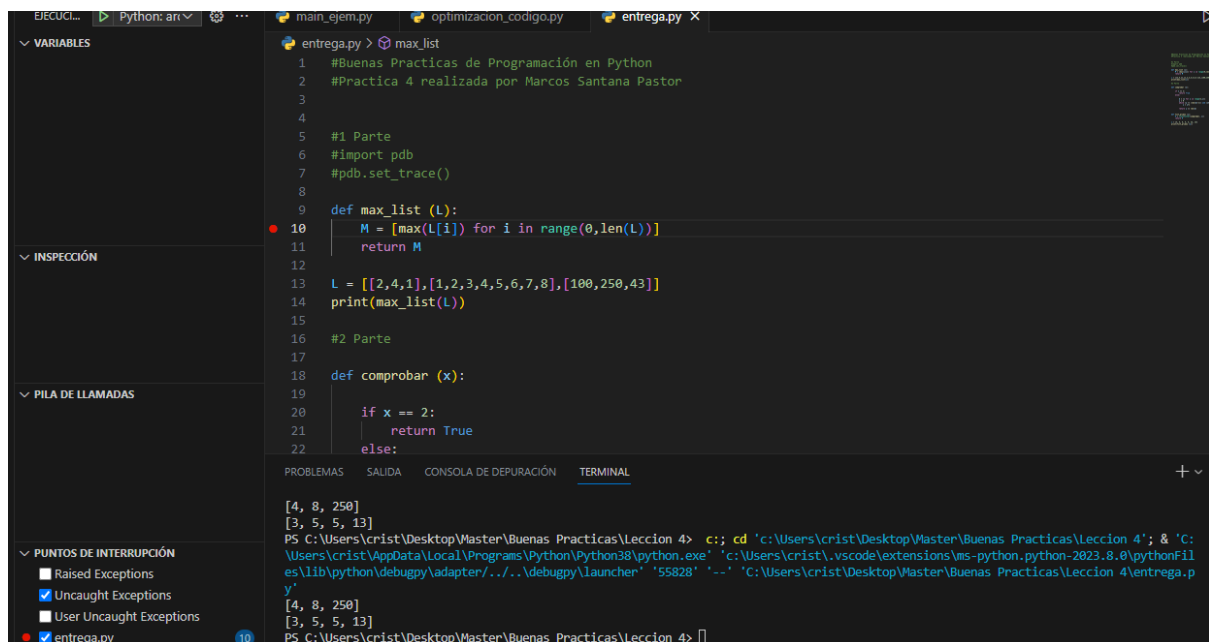
```

VARIABLES
  Locals
    .0: <range_iterator object at...
    L: [[2, 4, 1], [1, 2, 3, 4, 5...
    i: 2
  Globals

entrega.py > max_list
1 #Buenas Practicas de Programación en Python
2 #Practica 4 realizada por Marcos Santana Pastor
3
4
5 #1 Parte
6 #import pdb
7 #pdb.set_trace()
8
9 def max_list (L):
10     M = [max(L[i]) for i in range(0,len(L))]
11     return M
12

```

Y termina



```
1 #Buenas Practicas de Programación en Python
2 #Practica 4 realizada por Marcos Santana Pastor
3
4
5 #1 Parte
6 #import pdb
7 #pdb.set_trace()
8
9 def max_list (L):
10     M = [max(L[i] for i in range(0,len(L)))]
11     return M
12
13 L = [[2,4,1],[1,2,3,4,5,6,7,8],[100,250,43]]
14 print(max_list(L))
15
16 #2 Parte
17
18 def comprobar (x):
19
20     if x == 2:
21         return True
22     else:
23
24         m = [i for i in range(2,x)]
25         j = 0
26         while (j <= (len(m)-1)) and (x%m[j] != 0):
27             j = j+1
28
29         return j == len(m)
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
[4, 8, 250]
[3, 5, 5, 13]
PS C:\Users\crist\Desktop\Master\Buenas Practicas\Leccion 4> c:: cd 'c:\Users\crist\Desktop\Master\Buenas Practicas\Leccion 4'; & 'C:\Users\crist\AppData\Local\Programs\Python\Python38\python.exe' 'c:\Users\crist\.vscode\extensions\ms-python.python-2023.8.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '55828' '--' 'c:\Users\crist\Desktop\Master\Buenas Practicas\Leccion 4\entrega.py'
[4, 8, 250]
[3, 5, 5, 13]
PS C:\Users\crist\Desktop\Master\Buenas Practicas\Leccion 4> []
```

Para el segundo codigo simplemente creo una funcion que compruebe si un numero es primo.

```
16 #2 Parte
17
18 def comprobar (x):
19
20     if x == 2:
21         return True
22     else:
23
24         m = [i for i in range(2,x)]
25         j = 0
26         while (j <= (len(m)-1)) and (x%m[j] != 0):
27             j = j+1
28
29         return j == len(m)
30
```

Y es la función que utilizo en filter

```
def filt_primos (L):
    P = list(filter(comprobar, L))
    return P

L = [3, 4, 8, 5, 5, 22, 13]
print(filt_primos (L))
```

Finalmente da lo esperado.



The screenshot shows a VS Code terminal window with the 'TERMINAL' tab selected. The terminal displays the output of a Python script, which is a list: `[4, 8, 250]`. Below the output, there is a command prompt showing the current directory as `C:\Users\cris\Desktop\Master\Buenas Practicas\Leccion 4`. A tooltip is visible over the command prompt, indicating the command to open a new window: `Abrir carpeta en ventana nueva (ctrl + clic)`. The terminal also shows the command being executed: `c::; cd 'C:\Users\cris\Desktop\Master\Buenas Practicas\Leccion 4'; & 'C:\Users\cris\AppData\Local\Programs\Python\Python38\python.exe' 'c:\Users\cris\.vscode\extensions\ms-python.python-2023.8.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '55828' '--' 'C:\Users\cris\Desktop\Master\Buenas Practicas\Leccion 4\entrega.py'`. The status bar at the bottom indicates the file is at line 37, column 23, with 4 spaces, using UTF-8 encoding, CRLF line endings, and is a Python 3.8.10 64-bit file.

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  + v ...
[4, 8, 250]
Abrir carpeta en ventana nueva (ctrl + clic) as Practicas\Leccion 4> c::; cd 'C:\Users\cris\Desktop\Master\Buenas Practicas\Leccion 4'; & 'C:\Users\cris\AppData\Local\Programs\Python\Python38\python.exe' 'c:\Users\cris\.vscode\extensions\ms-python.python-2023.8.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '55828' '--' 'C:\Users\cris\Desktop\Master\Buenas Practicas\Leccion 4\entrega.py'
[4, 8, 250]
[3, 5, 5, 13]
PS C:\Users\cris\Desktop\Master\Buenas Practicas\Leccion 4>

Lín. 37, col. 23  Espacios: 4  UTF-8  CRLF  Python  3.8.10 64-bit
```