Justo Mota Márquez

Lección 4: Buenas Prácticas de Programación en Python

Apartado 1:

1. Creo una matriz de números enteros (en mi caso he usado la del ejemplo):

```
matrix = [[2, 4, 1], [1, 2, 3, 4, 5, 6, 7, 8], [100, 250, 43]] # Creo una matriz de números
```

2. Creo una compresión de listas en la que voy comprobando en cada lista que contiene la matriz, si el valor de cada iteración de la lista es igual al valor máximo de esta misma. Si el valor de la iteración es igual al valor máximo de la lista, le concatenamos a este número los caracteres ANSI correspondientes al texto en negrita, en el caso de que no sea igual el valor máximo que al valor de la iteración dejamos el número tal y como está:

```
numberMaxList = [['\033[1m' + str(j) + '\033[0m' if j == max(i) else str(j) for j in i] for i in matrix] #
```

3. Recorro con bucles la matriz que contiene la matriz inicial modificada con los valores máximos de cada lista en negrita, y voy pintando cada valor de esta misma:

```
print("1. Números mayores de cada lista de la matriz señalados en negrita: :")
print("[", end="") # Pinto el corchete inicial de la matriz
for lis in numberMaxList: # Recorro los valores de la matriz
    print("[", end="") # Pinto el corchete del inicio de cada lista que contiene la matriz
    for element in lis: # Recorro cada elemento de las listas de la matriz
        if element == lis[len(lis)-1]: # Si el valor del elemento es igual al valor de la última posición de la lista
            print(element, end="") # Pinto el valor del elemento
        else: # Si el valor del elemento no es igual al valor de la última posición de la lista
            print(element + ", ", end="") # Pinto el valor del elemento con una coma al final
    if lis == numberMaxList[len(numberMaxList)-1]: # Si el valor de lis es igual al valor de la última posición de la matriz
        print("]", end="") # Pinto el corchete de separación entre lista y lista
    else: # Si el valor de lis no es igual al valor de la última posición de la matriz
        print("], ", end="") # Pinto el corchete de separación entre lista y lista con una coma al final
print("]") # Pinto el corchete final de la matriz
```

4. El programa muestra por pantalla:

```
PS C:\Users\Justo\Desktop\proyectos-python\leccion4_bppp_Justo_Mota_Marquez> python .\__main__
1. Números mayores de cada lista de la matriz señalados en negrita: :
[[2, 4, 1], [1, 2, 3, 4, 5, 6, 7, 8], [100, 250, 43]]
```

5. Conclusión:
   La compresión de listas me ha hecho reducir líneas de código y a la vez aumentar la eficiencia del programa al guardar una lista filtrada directamente y no tener que hacer bucles anidados en mas de una línea.

Apartado 2:

1. Especifíco al programa la línea en la que quiero que comience a depurar, con la función “pdb.set_trace()”:

```
7       pdb.set_trace()
8       numberMaxList = [['\033[1m' + str(j) + '\033[0m' if j == max(i) else str(j) for j in i] for i in matrix]
9
```

```
PS C:\Users\Justo\Desktop\proyectos-python\leccion4_bppp_Justo_Mota_Marquez> python .\__main__.py
> c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(8)<module>()
-> numberMaxList = [['\033[1m' + str(j) + '\033[0m' if j == max(i) else str(j) for j in i] for i in matrix] # Creo una comprensión de listas en la que en los valores mayores de cada lista, concateno en secuencia de escape ANS
I los carácteres necesarios para escribir en negrita
(Pdb)
```

2. Creo un punto de parada en la línea anterior en donde empleo la compresión de listas y otro en la línea posterior que no esté en blanco, con el comando pdb “break (número de línea)”:

```
(Pdb) break 8
Breakpoint 1 at c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py:8
(Pdb)
```

```
(Pdb) break 10
Breakpoint 2 at c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py:10
```

3. Compruebo que exista el punto de parada:

```
(Pdb) b
Num Type         Disp Enb   Where
1   breakpoint   keep yes   at c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py:8
        breakpoint already hit 21 times
2   breakpoint   keep yes   at c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py:10
        breakpoint already hit 1 time
```

4. Compruebo el valor de la variable que contiene la compresión, con el comando “p (nombre de la variable)”:

```
(Pdb) p numberMaxList
[['2', '\x1b[1m4\x1b[0m', '1'], ['1', '2', '3', '4', '5', '6', '7', '\x1b[1m8\x1b[0m'], ['100', '\x1b[1m250\x1b[0m', '43']]
```

5. Ejecuto linea el programa hasta el final:

```
(Pdb) next
> c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(13)<module>()
-> print("[", end="") # Pinto el corchete del inicio de cada lista que contiene la matriz
(Pdb) next
[> c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(14)<module>()
-> for element in lis: # Recorro cada elemento de las listas de la matriz
(Pdb) next
> c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(15)<module>()
-> if element == lis[len(lis)-1]: # Si el valor del elemento es igual al valor de la última posición de la lista
(Pdb) next
> c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(18)<module>()
-> print(element + ", ", end="") # Pinto el valor del elemento con una coma al final
(Pdb) next
1, > c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(14)<module>()
-> for element in lis: # Recorro cada elemento de las listas de la matriz
(Pdb) next
> c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(15)<module>()
-> if element == lis[len(lis)-1]: # Si el valor del elemento es igual al valor de la última posición de la lista
(Pdb) next
> c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(18)<module>()
-> print(element + ", ", end="") # Pinto el valor del elemento con una coma al final
(Pdb) next
2, > c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(14)<module>()
-> for element in lis: # Recorro cada elemento de las listas de la matriz
(Pdb) next
> c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(15)<module>()
-> if element == lis[len(lis)-1]: # Si el valor del elemento es igual al valor de la última posición de la lista
(Pdb) next
> c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(18)<module>()
-> print(element + ", ", end="") # Pinto el valor del elemento con una coma al final
(Pdb) next
3, > c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(14)<module>()
-> for element in lis: # Recorro cada elemento de las listas de la matriz
(Pdb) next
> c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(15)<module>()
-> if element == lis[len(lis)-1]: # Si el valor del elemento es igual al valor de la última posición de la lista
(Pdb) next
> c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(18)<module>()
-> print(element + ", ", end="") # Pinto el valor del elemento con una coma al final
(Pdb) next
4, > c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(14)<module>()
-> for element in lis: # Recorro cada elemento de las listas de la matriz
(Pdb) next
> c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(15)<module>()
-> if element == lis[len(lis)-1]: # Si el valor del elemento es igual al valor de la última posición de la lista
(Pdb) next
> c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(18)<module>()
-> print(element + ", ", end="") # Pinto el valor del elemento con una coma al final
(Pdb) next
5, > c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(14)<module>()
-> for element in lis: # Recorro cada elemento de las listas de la matriz
(Pdb) next
> c:\users\justo\desktop\proyectos-python\leccion4_bppp_justo_mota_marquez\__main__.py(15)<module>()
-> if element == lis[len(lis)-1]: # Si el valor del elemento es igual al valor de la última posición de la lista
(Pdb) next
```

6. Conclusión:

   La depuración me ha servido para ver que esta pasando en mi código en cada línea e ir viendo los valores que van adoptando las variables

Apartado 3:

1. Creo una función que devuelva “True” si el número que le pasamos es primo y “False” si no lo es:

```python
def is_number_prime(num): # Función que devuelve si el número que le pasamos a la función es primo
    for n in range(2, num): # Creo un bucle que valla desde el numero 2 hasta el número que le pasamos a la función menos uno
        if num % n == 0: # Si algún número entre el 2 y el número-1 dividido entre el numero el modulo es 0, quiere decir que el número no es primo
            return False
    return True # Si el numero entre los valores que adopta el bucle no da su módulo 0, quiere decir que es primo
```

2. Creo una lista con valores desde el 1 al 20:

```python
numberList = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20] # Creo una lista con valores entre el 1 y el 20
```

3. Creo una lista filtrada que me guarda los números primos que hay entre el 1 y el 20, con la función “filter(función, lista)”:

```python
listPrimeNumber = list(filter(is_number_prime, numberList))
```

4. El programa muestra por pantalla:

```
2. Números primos entre el 1 y el 20:
[1, 2, 3, 5, 7, 11, 13, 17, 19]
```

5. Conclusión:
   El uso de la función “filter” agiliza el proceso de filtración de listas para eliminar los datos que no necesitamos, ahorrando en código y tiempo.