Algoritmos y Programación I (75.40 - 95.14) – Wachenchauzer – 1.er parcialito – 18/09/2017

Resolvé los siguientes problemas en forma clara y legible, respetando sangrías e incluyendo la documentación necesaria. Si te parece que los comentarios no son suficientemente explicativos, podés agregar una descripción de funcionamiento del código. Podés escribir tantas funciones auxiliares como creas necesarias.

1. Dado el siguiente código en Python

```
for x in range(-33, 99, 7):
    print("Valor al inicio de la iteracion:", x)
    x = (x // 2) - 3
    print("Valor al final de la iteracion:", x)
```

   a. Hacer una tabla con los valores de `x` iniciales y finales de cada iteración. En caso de que el ciclo no termine nunca, colocar una fila con "`... ...`".
   b. Transformar el ciclo anterior a un ciclo `while` que implemente el mismo comportamiento.

2. El estilo de escritura `snake_case` permite representar un conjunto de palabras separándolas por un guión bajo, mientras que el estilo `CamelCase` las representa sin separadores, utilizando letras mayúsculas para la primera letra de cada palabra.

   Se pide realizar una función que reciba una cadena escrita en `snake_case` y devuelva su representación en `CamelCase`. Ejemplos: `snake_a_camel("alan_turing") -> "AlanTuring"`, `snake_a_camel("hoy_es_el_parcial") -> "HoyEsElParcial"`.

3. Escribir una función que dada una matriz representada como una lista de listas de números (donde cada sublista representa una **fila**), devuelva una lista con los máximos de cada **columna**. Ejemplo:

```
maximos_columnas([[1, 2, 8, 4],
                  [6, 7, 3, 3],   ->   [6, 7, 8, 9]
                  [6, 5, 4, 9]])
```

¡Suerte! `:-)`

Algoritmos y Programación I (75.40 - 95.14) – Wachenchauzer – 1.er parcialito – 18/09/2017

Resolvé los siguientes problemas en forma clara y legible, respetando sangrías e incluyendo la documentación necesaria. Si te parece que los comentarios no son suficientemente explicativos, podés agregar una descripción de funcionamiento del código. Podés escribir tantas funciones auxiliares como creas necesarias.

1. Dado el siguiente código en Python

```
for x in range(-33, 99, 7):
    print("Valor al inicio de la iteracion:", x)
    x = (x // 2) - 3
    print("Valor al final de la iteracion:", x)
```

   a. Hacer una tabla con los valores de `x` iniciales y finales de cada iteración. En caso de que el ciclo no termine nunca, colocar una fila con "`... ...`".
   b. Transformar el ciclo anterior a un ciclo `while` que implemente el mismo comportamiento.

2. El estilo de escritura `snake_case` permite representar un conjunto de palabras separándolas por un guión bajo, mientras que el estilo `CamelCase` las representa sin separadores, utilizando letras mayúsculas para la primera letra de cada palabra.

   Se pide realizar una función que reciba una cadena escrita en `snake_case` y devuelva su representación en `CamelCase`. Ejemplos: `snake_a_camel("alan_turing") -> "AlanTuring"`, `snake_a_camel("hoy_es_el_parcial") -> "HoyEsElParcial"`.

3. Escribir una función que dada una matriz representada como una lista de listas de números (donde cada sublista representa una **fila**), devuelva una lista con los máximos de cada **columna**. Ejemplo:

```
maximos_columnas([[1, 2, 8, 4],
                  [6, 7, 3, 3],   ->   [6, 7, 8, 9]
                  [6, 5, 4, 9]])
```

¡Suerte! `:-)`