Ayuda para el maratón de retos 2013

Este documento explica brevemente cómo se deben de presentar los problemas de la maratón para su (pre) calificación. La idea de la maratón es que ustedes implementen un catálogo de funciones, son varios algoritmos, cada uno representado por una función.

Tutorial del formato de su proyecto

A continuación explico cómo deben de organizar el código de su proyecto:

1. Creen un directorio (folder) con el numero de su grupo, por ejemplo **grupo1**
2. Adentro de ese directorio creen un archivo que se llame **__init__.py**, este archivo DEBE QUEDARSE VACÍO.
3. Al mismo nivel de **__init__.py** creen otro archivo que se llame **solutions.py** este será su script principal.
4. **solutions.py** contendrá cada una de sus soluciones a los problemas, CADA SOLUCIÓN A UN PROBLEMA SERÁ UNA FUNCIÓN DIFERENTE**.**

Explicación breve del formato de las funciones

Las funciones se ven como sigue:

```
# Problema 1: Dado un entero positivo, determine si es múltiplo de 3
def prob_1(n):
        if n % 3 == 0:
                return True
        else:
                return False
```

Dense cuenta de la estructura de la función, responde al enunciado … Esta función recibe un entero **n,** luego presenta un algoritmo para determinar si **n** es múltiplo de 3. Si es múltiplo de 3, la función responde **True** (puesto que **n** si es múltiplo de tres); en caso contrario, la función responde **False** (puesto que **n** no es múltiplo de tres)

¿Tiene toda la lógica del mundo va?

Ahora veamos el ejemplo del problema 5, este dice: “Dados dos vectores (o listas) encuentre el vector (o lista) resultante del producto cruz entre ambos vectores”. Entonces, primero que nada sabemos que tenemos que escribir una función con este encabezado:

```
 # Problema 5: Dadas dos vectores … calcular su producto cruz
def prob_5(lista1, lista2):
```

Acá estamos indicando que nuestra función recibe como entrada dos listas, las cuales vamos a multiplicar como vectores. Lo que nuestra función debe **regresar**

es otra lista, conteniendo el resultado de la multiplicación. Entonces, la parte final de nuestra función se vería así:

```
return listaMultiplicada
```

¿Cómo probamos nuestras funciones?

Hagamos el ejercicio del problema 1 y el problema 6. El problema 1 dice que tenemos que encontrar si un número dado es par o impar, entonces al correr esta función:

```
prob_1(5) # Esperamos ver False de resultado
```

Esperaríamos ver de resultado **False**, pero si corremos esta función:

```
prob_1(2) # Esperamos ver True de resultado
```

Esperaríamos ver de resultado **True**

Ahora veamos el problema 6, este problema dice que recibimos de entrada una lista de enteros y queremos obtener la lista ordenada de salida, entonces probamos nuestro algoritmo así:

```
prob_5([6, 3, 9, 1]) # Esperamos ver de resultado  [1, 3, 6, 9]
```

Esperaríamos ver de resultado **[1, 3, 6, 9]**,

Si sus funciones siempre regresan los resultados esperados, podemos concluir que están bastante bien. Si ustedes pueden demostrar que sus funciones SIEMPRE regresan el resultado esperado, su ejercicio dentro de la maratón está resuelto correctamente.

¿Cómo le hago para saber cómo va mi equipo?

Si siguen estas reglas, pueden enviarme un correo con un zip de su archivo **grupoN** (donde N es su número de grupo), y yo les regreso un correo con su evaluación automática.

Por favor empiecen su proyecto en cuanto antes, así se resuelven todas las dudas antes de la entrega. ¡Sigan las instrucciones!