Desafío - Estructuras de datos y funciones (II)

En este desafío validaremos nuestros conocimientos de funciones para la codificación de un programa en Python.

Lee todo el documento antes de comenzar el desarrollo **individual**, para asegurarte de tener el máximo de puntaje y enfocar bien los esfuerzos.

Descripción

La empresa de desarrollo de software **DESARROLLA** se encuentra actualmente trabajando en muchos proyectos distintos. Es tanta la demanda que te solicita trabajar en 3 soluciones que tienen pendientes. Para ello, te entregarán los requerimientos de cada tarea y deberás implementar una función que entregue la solución a cada problema.

Requerimientos

1. Filtrado relevante. (**3 Puntos**)

La empresa tiene un contrato con una tienda de tecnología en la cual quieren implementar un filtrado por precio. Para ello se solicita generar el archivo `filtro.py` con la solución al problema. Dada una muestra de los productos que actualmente se encuentran disponibles en la tienda (un diccionario), se solicita implementar una función que permita entregar lo siguiente:

- Un diccionario con los productos que cumplen una cierta condición dado un umbral
- La función debe permitir mostrar los valores mayor que o menor que un umbral (siempre estrictos).
- Por defecto la función debe siempre mostrar los valores mayor que el umbral a menos que se indique lo contrario.

Para desarrollar la funcionalidad se le entrega a usted un diccionario de prueba para verificar sus resultados.

```
precios = {'Notebook': 700000,
           'Teclado': 25000,
           'Mouse': 12000,
           'Monitor': 250000,
           'Escritorio': 135000,
           'Tarjeta de Video': 1500000}
```

Se espera ejecutar el programa de la siguiente manera:

- Si se especifica un argumento, este debe ser el umbral y por defecto debe calcular los que son estrictamente mayores al umbral.

```
python filtro.py 30000
```

```
Los productos mayores al umbral son: Notebook, Monitor, Escritorio,
Tarjeta de Video
```

- En caso de que se ingresen dos valores, el primero seguirá siendo el umbral, mientras el segundo podrá tomar los valores mayor o menor. Por ejemplo, el siguiente código calculará los que son estrictamente menores.

```
python filtro.py 30000 menor
```

```
Los productos menores al umbral son: Teclado, Mouse
```

En caso que otro elemento se utilice se debe devolver lo siguiente:

```
python filtro.py 30000 otro
```

```
Lo sentimos, no es una operación válida
```

 **TIP**: El método `.join()` podría ser de utilidad para este problema:

```
', '.join(['uno','dos','tres','cuatro']
```

Donde el string `', '` funcionará como el separador de los elementos de la lista dentro del método join. Es decir, el output será el siguiente:

```
uno, dos, tres, cuatro
```

2. Alertas telemáticas. (**3 Puntos**)

Otra solución que se encuentra pendiente es la encargada por una empresa de flotas que debe medir mediante telemetría las velocidades de cada una de sus correas transportadoras. Una de sus políticas es distribuir su energía de manera eficiente, por lo que, para poder entregar energía a las correas más lentas, es necesario ralentizar las más rápidas, para asegurar una correcta distribución de la energía disponible. Para ello, se requiere levantar una alerta de la posición de las correas transportadoras que están por sobre el promedio.

- Para ello se pide determinar una funcionalidad que calcule el promedio de una lista de velocidades. El servidor donde se pretende instalar esta funcionalidad no cuenta con mucha capacidad por lo que se pide no depender de librerías externas.
- Listar las posiciones de todas las correas transportadoras que están sobre el promedio.
- Implementar la solución mediante una función en un archivo llamado `velocidad.py`.

Se entrega la siguiente lista con una muestra de velocidades para probar las funcionalidades.

```
velocidad = [25, 12, 19, 16, 11, 11, 24, 1,
 14, 14, 16, 10, 6, 23, 13, 25, 4, 19,
 14, 20, 18, 9, 18, 4, 18, 1, 3, 4, 2,
 14, 23, 19, 23, 9, 18, 20, 22, 14, 1,
 10, 5, 23, 3, 5, 9, 5, 3, 12, 20, 5,
 11, 10, 18, 10, 14, 5, 23, 20, 23, 21]
```

La salida que se espera en este caso es la siguiente:

```
python velocidad.py
```

```
[0, 2, 3, 6, 8, 9, 10, 13, 15, 17, 18, 19, 20, 22, 24, 29, 30, 31, 32,
34, 35, 36, 37, 41, 48, 52, 54, 56, 57, 58, 59]
```

3. Apoyo matemático. (**4 Puntos**)

Otra área en la que la empresa presta soporte es a las ONG. En un programa de ayuda escolar que tiene la esta ONG se está enseñando a programar algunas operaciones avanzadas a niños con alto potencial pero de escasos recursos. Se quiere enseñar dos operaciones conocidas como el factorial y la productoria y se requiere que usted prepare una script que implemente esto.

El factorial se define de la siguiente forma:

$$n! = n * n-1 * n - 2 *\ldots* 2 * 1$$

En un ejemplo práctico, el factorial de 5 (5!) se calculará como:

$$5! = 5 * 4 * 3 * 2 * 1 = 120$$

Por otro lado la productoria se define como la multiplicación de elementos.

$$A = [3,6,4,2,8]$$

$$\prod A_i = 3 * 6 * 4 * 2 * 8$$

Para resolver este programa se solicita lo siguiente:

- Crear un script llamado `ong.py` que contenga las siguientes funciones:
  - Una función que calcule el factorial.
  - Una función que calcule la productoria.
  - Una función que permita controlar los cálculos. Esta función se debe invocar de la siguiente manera:

```
calcular(fact_1 = 5, prod_1 = [3,6,4,2,8], fact_2 = 6)
```

Se ingresarán un valor numérico como argumento con el nombre `fact_i` cuando se requiera calcular un factorial, y una lista como argumento `prod_i` cuando se requiera calcular una productoria. Cabe destacar que la función debe permitir ingresar estos argumentos en cualquier orden y en cualquier cantidad. El resultado de la función se debe imprimir en pantalla de la siguiente forma:

```
El factorial de 5 es 120
La productoria de [4, 6, 7, 4, 3] es 2016
El factorial de 6 es 720
```

 **NOTA**: Esta función no será ejecutada desde el terminal sino que desde el mismo script utilizando diferentes combinaciones de requerimientos de factorial y productoria.

 **TIP**: El operador `in` podría ser de utilidad acá ya que es capaz de detectar trozos de string. Por ejemplo:

```
'gato' in 'hay un gato acá'
```

```
True
```

 ¡Mucho éxito!

Consideraciones y recomendaciones

- Comprime el desarrollo de los 3 requerimientos en un archivo .zip y luego sube tru respuesta en el LMS.