Paradigmas de Programación: Práctica 1

Vamos a crear desarrollar una serie de funciones utilizando programación funcional en Python. Todas las implementaciones deben respetar los nombres marcados en negrita, así como la entrada y salida especificada. Para cada función, se deben implementar un mínimo de 5 ejemplos aplicándolas. Se valorará cada apartado en función del uso adecuado de las características de **programación funcional** vistas en clase.

Parte 1 (4 puntos):

- *(1 punto)* Función **palíndromos**, que recibirá una lista de strings y nos devolverá una lista de True/False indicando cuales eran (o no) palíndromos. Para ello crearemos otra función **es_palindromo** que, a partir de un string, nos devuelva un booleano indicando si el string es palíndromo.
- *(1 punto)* Función **cuadrados_sumados** que, a partir de un numero natural n devuelva la suma de los cuadrados de todos los números entre 1 y n
- *(1 punto)* Función **factorial** que, tomando un numero como entrada, nos devuelva el factorial de ese número
- *(1 punto)* Tenemos una lista de estudiantes, donde cada estudiante tiene su nombre y una lista de calificaciones. Queremos filtrar a los estudiantes que tienen un promedio de calificaciones mayor o igual a 6 (nota mínima para aprobar). Como ejemplo podemos utilizar la siguiente lista:
    - estudiantes = [
        {"nombre": "Juan", "calificaciones": [7, 8, 6, 5]},
        {"nombre": "Ana", "calificaciones": [9, 6, 8]},
        {"nombre": "Pedro", "calificaciones": [4, 5, 3]},
        {"nombre": "Maria", "calificaciones": [10, 9, 9]},
        {"nombre": "Luis", "calificaciones": [3, 2, 5]}
      ]

Parte 2 (6 puntos):
Script “haveibeenpwned”, que empleará la API “pwnedpasswords” de la plataforma Have I Been Pwned (https://haveibeenpwned.com/API/v3#PwnedPasswords). En específico, usaremos las siguientes funcionalidades y endpoints de la misma:

- Searching by range:
    - Mediante este endpoint, podremos verificar si una contraseña dada aparece en filtraciones de datos conocidas y almacenadas por Have I Been Pwned. Acepta contraseñas hasheadas en SHA1 y NTLM; a la llamada se le especifica únicamente el prefijo del hash, y devuelve los sufijos de los hashes cuyos prefijos coincidan con el especificado, así como un recuento de las veces que han sido filtrados.
    - Request: `GET https://api.pwnedpasswords.com/range/{first 5 hash chars}` y `GET https://api.pwnedpasswords.com/range/{prefijo}?mode=ntlm`
- Getting all breached sites in the system
    - Mediante este endpoint podremos verificar si existen filtraciones o colecciones de datos relacionadas con un dominio especificado.
    - Request: `GET https://haveibeenpwned.com/api/v3/breaches?Domain={dom}`

El objetivo de este ejercicio es, mediante programación funcional, realizar un script multimodo que permita al usuario:

1. Elegir el modo de funcionamiento (comprobación de contraseñas / comprobación de dominios)
    a. Para el modo de contraseñas, especificar si se quiere consultar formato Windows (NTLM) o estándar (SHA1). A continuación, solicitar contraseña.
    b. Para el modo de dominios, solicitar el dominio.
2. Mostrar el resultado pertinente por pantalla, dependiendo del modo escogido.

Será preciso que se empleen callbacks para la selección del tipo de hash, así como gestión de casuísticas y errores relacionados con las requests.

Observaciones:

- En ningún ejercicio hace falta gestionar las excepciones que ocurran por pasar tipos incorrectos como parámetro
- No se puntuará ningún ejercicio (aunque funcione correctamente) que no aplique los conceptos vistos sobre el paradigma de programación funcional
- El script de Python debe contener al menos cinco llamadas a cada una de las funciones para comprobar su correcto funcionamiento
- Además del código de las funciones en un archivo .py, será necesaria una explicación sobre el funcionamiento de cada función desarrollada en un pdf