PROYECTO LÓGICA: Katas de Python

1. Escribe una función que reciba una cadena de texto como parámetro y devuelva un diccionario con las frecuencias de cada letra en la cadena. Los espacios no deben ser considerados.
2. Dada una lista de números, obtén una nueva lista con el doble de cada valor. Usa la función `map()`
3. Escribe una función que tome una lista de palabras y una palabra objetivo como parámetros. La función debe devolver una lista con todas las palabras de la lista original que contengan la palabra objetivo.
4. Genera una función que calcule la diferencia entre los valores de dos listas. Usa la función `map()`
5. Ecribe una función que tome una lista de números como parámetro y un valor opcional nota_aprobado, que por defecto es 5. La función debe calcular la media de los números en la lista y determinar si la media es mayor o igual que nota aprobado. Si es así, el estado será "aprobado", de lo contrario, será "suspenso". La función debe devolver una tupla que contenga la media y el estado.
6. Escribe una función que calcule el factorial de un número de manera recursiva.
7. Genera una función que convierta una lista de tuplas a una lista de strings. Usa la función `map()`
8. Escribe un programa que pida al usuario dos números e intente dividirlos. Si el usuario ingresa un valor no numérico o intenta dividir por cero, maneja esas excepciones de manera adecuada. Asegúrate de mostrar un mensaje indicando si la división fue exitosa o no.
9. Escribe una función que tome una lista de nombres de mascotas como parámetro y devuelva una nueva lista excluyendo ciertas mascotas prohibidas en España. La lista de mascotas a excluir es ["Mapache", "Tigre", "Serpiente Pitón", "Cocodrilo", "Oso"].Usa la función `filter()`
10. Escribe una función que reciba una lista de números y calcule su promedio. Si la lista está vacía, lanza una excepción personalizada y maneja el error adecuadamente.
11. Escribe un programa que pida al usuario que introduzca su edad. Si el usuario ingresa un valor no numérico o un valor fuera del rango esperado (por ejemplo, menor que 0 o mayor que 120), maneja las excepciones adecuadamente.
12. Genera una función que al recibir una frase devuelva una lista con la longitud de cada palabra. Usa la función `map()`
13. Genera una función la cual, para un conjunto de caracteres, devuelva una lista de tuplas con cada letra en mayúsculas y minúsculas. Las letras no pueden estar repetidas .Usa la función `map()`
14. Crea una función que retorne las palabras de una lista de palabras que comience con una letra en especifico. Usa la función `filter()`
15. Crea una función `lambda` que sume 3 a cada número de una lista dada.
16. Escribe una función que tome una cadena de texto y un número entero n como parámetros y devuelva una lista de todas las palabras que sean más largas que n. Usa la función `filter()`
17. Crea una función que tome una lista de dígitos y devuelva el número correspondiente. Por ejemplo, [5,7,2] corresponde al número quinientos setenta y dos (572). Usa la función `reduce()`
18. Escribe un programa en Python que cree una lista de diccionarios que contenga información de estudiantes (nombre, edad, calificación) y use la función filter para extraer a los estudiantes con una calificación mayor o igual a 90. Usa la función `filter()`
19. Crea una función `lambda` que filtre los números impares de una lista dada.
20. Para una lista con elementos tipo integer y string obtén una nueva lista sólo con los valores int. Usa la función `filter()`
21. Crea una función que calcule el cubo de un número dado mediante una función `lambda`
22. Dada una lista numérica, obtén el producto total de los valores de dicha lista.Usa la función `reduce()`.
23. Concatena una lista de palabras.Usa la función `reduce()`.
24. Calcula la diferencia total en los valores de una lista. Usa la función `reduce()`.
25. Crea una función que cuente el número de caracteres en una cadena de texto dada.

26. Crea una función `lambda` que calcule el resto de la división entre dos números dados.
27. Crea una función que calcule el promedio de una lista de números.
28. Crea una función que busque y devuelva el primer elemento duplicado en una lista dada.
29. Crea una función que convierta una variable en una cadena de texto y enmascare todos los caracteres con el carácter '#', excepto los últimos cuatro.
30. Crea una función que determine si dos palabras son anagramas, es decir, si están formadas por las mismas letras pero en diferente orden.
31. Crea una función que solicite al usuario ingresar una lista de nombres y luego solicite un nombre para buscar en esa lista. Si el nombre está en la lista, se imprime un mensaje indicando que fue encontrado, de lo contrario, se lanza una excepción.
32. Crea una función que tome un nombre completo y una lista de empleados, busque el nombre completo en la lista y devuelve el puesto del empleado si está en la lista, de lo contrario, devuelve un mensaje indicando que la persona no trabaja aquí.
33. Crea una función `lambda` que sume elementos correspondientes de dos listas dadas.
34. Crea la clase `Arbol`, define un árbol genérico con un tronco y ramas como atributos. Los métodos disponibles son: `crecer_tronco`, `nueva_rama`, `crecer_ramas`, `quitar_rama` e `info_arbol`. El objetivo es implementar estos métodos para manipular la estructura del árbol.

    Código a seguir:

    1. Inicializar un árbol con un tronco de longitud 1 y una lista vacía de ramas.
    2. Implementar el método `crecer_tronco` para aumentar la longitud del tronco en una unidad.
    3. Implementar el método `nueva_rama` para agregar una nueva rama de longitud 1 a la lista de ramas.
    4. Implementar el método `crecer_ramas` para aumentar en una unidad la longitud de todas las ramas existentes.
    5. Implementar el método `quitar_rama` para eliminar una rama en una posición específica.

    6. Implementar el método
    `info_arbol` para devolver información sobre la longitud del tronco, el número de ramas y las longitudes de las mismas.

    Caso de uso:

    1. Crear un árbol.
    2. Hacer crecer el tronco del árbol una unidad.
    3. Añadir una nueva rama al árbol.
    4. Hacer crecer todas las ramas del árbol una unidad.
    5. Añadir dos nuevas ramas al árbol.
    6. Retirar la rama situada en la posición 2.
    7. Obtener información sobre el árbol.
36. Crea la clase `UsuarioBanco`,representa a un usuario de un banco con su nombre, saldo y si tiene o no cuenta corriente. Proporciona métodos para realizar operaciones como retirar dinero, transferir dinero desde otro usuario y agregar dinero al saldo.

    Código a seguir:

    1. Inicializar un usuario con su nombre, saldo y si tiene o no cuenta corriente mediante `True` y `False`.
    2. Implementar el método `retirar_dinero` para retirar dinero del saldo del usuario. Lanzará un error en caso de no poder hacerse.
    3. Implementar el método `transferir_dinero` para realizar una transferencia desde otro usuario al usuario actual. Lanzará un error en caso de no poder hacerse.
    4. Implementar el método `agregar_dinero` para agregar dinero al saldo del usuario.

    Caso de uso:

    1. Crear dos usuarios: "Alicia" con saldo inicial de 100 y "Bob" con saldo inicial de 50, ambos con cuenta corriente.

2. Agregar 20 unidades de saldo de "Bob".
3. Hacer una transferencia de 80 unidades desde "Bob" a "Alicia".
4. Retirar 50 unidades de saldo a "Alicia".

37. Crea una función llamada `procesar_texto` que procesa un texto según la opción especificada: `contar_palabras`, `reemplazar_palabras`, `eliminar_palabra`. Estas opciones son otras funciones que tenemos que definir primero y llamar dentro de la función `procesar_texto`.

    Código a seguir:

    1. Crear una función `contar_palabras` para contar el número de veces que aparece cada palabra en el texto. Tiene que devolver un diccionario.
    2. Crear una función `reemplazar_palabras` para remplazar una `palabra_original` del texto por una `palabra_nueva`. Tiene que devolver el texto con el remplazo de palabras.
    3. Crear una función `eliminar_palabra` para eliminar una palabra del texto. Tiene que devolver el texto con la palabra eliminada.
    4. Crear la función `procesar_texto` que tome un texto, una opción(entre "contar", "reemplazar", "eliminar") y un número de argumentos variable según la opción indicada.

    Caso de uso:

        Comprueba el funcionamiento completo de la función `procesar_texto`

38. Genera un programa que nos diga si es de noche, de día o tarde según la hora proporcionada por el usuario.
39. Escribe un programa que determine qué calificación en texto tiene un alumno en base a su calificación numérica. Las reglas de calificación son:

    - 0 - 69 insuficiente

    - 70 - 79 bien

    - 80 - 89 muy bien

    - 90 - 100 excelente

40. Escribe una función que tome dos parámetros: `figura` (una cadena que puede ser `"rectangulo"`, `"circulo"` o `"triangulo"`) y `datos` (una tupla con los datos necesarios para calcular el área de la figura).
41. En este ejercicio, se te pedirá que escribas un programa en Python que utilice condicionales para determinar el monto final de una compra en una tienda en línea, después de aplicar un descuento. El programa debe hacer lo siguiente:

    1. Solicita al usuario que ingrese el precio original de un artículo.

    2. Pregunta al usuario si tiene un cupón de descuento (respuesta sí o no).

    3. Si el usuario responde que sí, solicita que ingrese el valor del cupón de descuento.

    4. Aplica el descuento al precio original del artículo, siempre y cuando el valor del cupón sea válido (es decir, mayor a cero). Por ejemplo, descuento de 15€.

    5. Muestra el precio final de la compra, teniendo en cuenta el descuento aplicado o sin él.

    6. Recuerda utilizar estructuras de control de flujo como if, elif y else para llevar a cabo estas acciones en tu programa de Python.