**Trabajo Práctico 7: Diccionarios**

1. Desarrolle un programa que almacene datos de canciones en formato MP3: Artista, Título, Duración (en segundos), Tamaño del fichero (en KB). Un programa debe pedir los datos de una canción al usuario y después mostrarlos en pantalla. Debe interrumpirse la carga cuando el artista ingresado sea vacío.
2. Escriba un programa que ingrese (por teclado) los datos de diez personas (nombre, edad, genero, dirección, teléfono), los almacene en un diccionario y los muestre. Al realizar dicha muestra, destacar la persona más joven en edad.
3. Hacer un programa que registre 10 alumnos y guarde: nombre, nombre de la asignatura y 4 notas. Calcular y mostrar el promedio de las notas.
4. Realice un programa que pida datos de personas: nombre, día de nacimiento, mes de nacimiento, y año de nacimiento. Después deberá repetir lo siguiente: preguntar un número de mes y mostrar en pantalla los datos de las personas que cumplan los años durante ese mes. Terminará de repetirse cuando se teclee vacío en el nombre. Persista y recupere la información en cada ejecución en un archivo llamado *cumpleaños.json*.
5. Se debe gestionar los datos de stock de una tienda de comestibles, la información a recoger será: nombre del producto, precio, cantidad en stock. La tienda dispone de 10 productos distintos. El programa debe ser capaz de:

* Dar de alta un producto nuevo.
* Buscar un producto por su nombre.
* Modificar el stock y precio de un producto dado.

1. Juego de adivinanza de frutas: Crea un diccionario con nombres de frutas como claves y sus descripciones como valores. El programa debe elegir aleatoriamente una fruta y mostrar su descripción. Luego, el usuario debe adivinar qué fruta es. Si el usuario adivina correctamente, el programa debe mostrar un mensaje de felicitación.
2. Contador de palabras: Escribe un programa que lea un archivo de texto y cuente la cantidad de veces que aparece cada palabra en el archivo. Utiliza un diccionario para almacenar las palabras y sus frecuencias. Al final, muestra las 10 palabras más comunes en el archivo.
3. Sistema de calificaciones: Escribe un programa que permita a los profesores registrar las calificaciones de sus alumnos y les permita calcular la nota final. Crea un diccionario para cada alumno, con su legajo como clave y una lista de notas como valor. Luego, el programa debe permitir al usuario ingresar las notas para cada alumno y calcular su nota final, basándose en un sistema de pesos predeterminado.
4. Supongamos que un coleccionista de figuras Funko Pop de Rick y Morty tiene un carrito de compras con las siguientes figuras y sus respectivas cantidades:

carrito = {

"Rick Sanchez" : 2,

"Morty Smith" : 3,

"Summer Smith" : 1,

"Mr. Meeseeks" : 4

}

Y además, el coleccionista conoce los precios unitarios (en dólares) de estas figuras:

precios = {

"Rick Sanchez" : 15,

"Morty Smith" : 12,

"Summer Smith" : 10,

"Mr. Meeseeks" : 20

}

Cree una función llamada *precioTotal* que calculará el monto total de la compra en función de estos datos. Cuando se llame a la función *precioTotal(carrito, precios)* con los diccionarios proporcionados, se debe obtener el monto total de la compra basado en la multiplicación de la cantidad de cada figura por su precio unitario en dólares.

1. Se desea implementar la funcionalidad del juego «piedra, papel o tijera». Para ello, se tiene una variable del tipo diccionario, en la que se almacena la regla de negocio de quién le gana a quién (la clave es el elemento, el valor a qué elemento le gana):

*opcionesGanadoras = {*

*"piedra" : "tijera",*

*"papel" : "piedra",*

*"tijera" : "papel"*

*}*

En una segunda variable, se almacenan un lote (lista) de jugadas que deben ser procesadas, distinguiendo a dos jugadores, en los que se especifica como valor su nombre y la opción seleccionada en esa jugada:

*jugadas = [ {*

*"jugador1" : {"nombre": "Oliver", "opcion": "piedra"},*

*"jugador2" : {"nombre": "Gabriela", "opcion": "papel"}*

*},*

*{*

*"jugador1" : {"nombre": "Oliver", "opcion": "tijera"},*

*"jugador2" : {"nombre": "Gabriela", "opcion": "papel"}*

*}*

*]*

Con relación a dicha información, se deberán elaborar las siguientes funciones:

* **resultado(opcion1, opcion2, opcionesGanadoras):** Recibe como parámetro dos opciones (“piedra”, “papel” o “tijera”) junto con la variable de opcionesGanadoras. Deberá retornar *None* si hay un empate; 1 si gana la opción 1; y el valor 2 si gana la opción 2. Por ejemplo, dado los siguientes llamados, la función deberá retornar:

*resultado ('papel', 'papel', opcionesGanadoras) ==> None*

*resultado ('tijera', 'papel', opcionesGanadoras) ==> 1*

*resultado ('tijera', 'piedra', opcionesGanadoras) ==> 2*

* **analizarJugada(jugada, opcionesGanadoras):** Recibe una jugada como parámetros junto con las opciones ganadoras. Dentro de la función deberá utilizar la función resultado (definida en el ítem anterior) y retornar la leyenda: “Empate”, en caso de igualdad; o bien, “Gana {nombre del jugador}”. Utilizando las variables anteriores como ejemplo, debería retornar:

*analizarJugada({'jugador1': {'nombre': 'Oliver', 'opcion': 'piedra'}, 'jugador2': {'nombre': 'Gabriela', 'opcion': 'papel'}}, opcionesGanadoras) ==> Gana Gabriela*

*analizarJugada({'jugador1': {'nombre': 'Oliver', 'opcion': 'tijera'}, 'jugador2': {'nombre': 'Gabriela', 'opcion': 'papel'}}, opcionesGanadoras) ==> Gana Oliver*

1. Se tienen dos variables del tipo diccionario, en una de ella se almacena la información de los artículos y la cantidad que tiene una persona en un carrito de compras:

*carrito = {*

*"lapiceras" : 12,*

*"borrador" : 1,*

*"carpeta" : 2*

*}*

En una segunda variable, se almacenan el stock (cantidad de artículos disponibles) de cada uno de los artículos:

*stock = {*

*"lapiceras" : 13,*

*"borrador" : 10,*

*"carpeta" : 3*

*}*

Con relación a dicha información, se deberán elaborar las siguientes funciones:

* **hayStock(articulo, cantidad, stock**): Recibe un artículo y verifica si hay stock disponible (retorna True en caso de que exista; False en caso contrario).

*hayStock("borrador", 1, stock) => True*

*hayStock("borrador", 13, stock) => False*

* **procesarPedido(carrito, stock):** Recibe los artículos solicitados en carrito y realiza el descuento de stock correspondiente. Debe retornar como resultado el stock actualizado. No afectar la variable recibida como parámetro. Utilizando las variables anteriores como ejemplo, debería retornar:

*{'lapiceras': 1, 'borrador': 9, 'carpeta': 1}*

1. En una tienda que confecciona banderas de países de tela, se tiene en su programa dos variables para cumplir su objetivo, en una de ella se almacena la información de colores de las telas que se tienen a disposición dentro de un conjunto:

*colores = { "azul", "blanco", "rojo"}*

Y, en una segunda variable, los colores necesarios para confeccionar las banderas de los diferentes países:

*banderas = {*

*"Argentina" : { "blanco", "celeste" },*

*"Francia" : { "azul", "blanco", "rojo"},*

*"Polonia" : { "blanco", "rojo"}*

*}*

En función de dicha información, deberá elaborar las siguientes funciones (utilice dentro de las mismas operaciones de conjuntos para lograr el resultado):

* **banderasPosibles(colores, banderas)**, que retorne una lista con el nombre de las banderas que son posibles de realizar con los colores indicados. Siguiendo los ejemplos brindados, debería retornar lo siguiente:

*['Francia', 'Polonia']*

* **coloresFaltantes (colores, banderas)**, que retorne un diccionario donde por cada bandera existente indique la lista de colores faltantes para realizarla. Si no falta ningún color, debería mostrarse la lista vacía. En relación a los ejemplos brindados, este debería ser el resultado del llamado de la función:

*{*

*'Argentina': ['celeste'],*

*'Francia': [],*

*'Polonia': []*

*}*

1. En un sistema de recetas de cocinas, se tiene dos variables. En una de ellas, se almacena en un conjunto la información de los ingredientes que tiene una persona en su casa a disposición:

*ingredientes = { "huevo", "aceite", "papas"}*

En una segunda variable, se encuentran parametrizadas las recetas existentes indicando que ingredientes se requieren para cada preparación:

*recetas = {*

*"Papas fritas" : { "aceite", "papas" },*

*"Huevo frito" : { "huevo", "aceite" },*

*"Pure de papas" : { "papas", "manteca" }*

*}*

Conociendo dicha información, se deberán elaborar las siguientes funciones (que deben incluir dentro de la misma operaciones de conjunto en su resolución):

* **recetasPosibles (ingredientes, recetas)**, que retorne una lista con el nombre de las recetas que son posibles de realizar con los ingredientes indicados. Resultado esperado con los ejemplos dados:

*['Papas fritas', 'Huevo frito']*

* **ingredientesFaltantes (ingredientes, recetas)**, que retorne un diccionario donde, por cada receta existente, indique la lista de ingredientes faltantes para realizar la receta. Si no falta ningún ingrediente, debería mostrarse la lista vacía. Resultado esperado con los ejemplos dados:

*{'Papas fritas': [], 'Huevo frito': [], 'Pure de papas': ['manteca']}*