



Nivel

Nombre curso

No. Guía

Universitario

Técnicas de programación

Taller #1

Taller listas, pilas, colas (Implemente los siguientes ejercicios utilizando persistencia con archivos)

Ejercicio #1: Aplicación de Aprendizaje de Vocabulario en Inglés

Desarrolla una aplicación de consola en **Python** utilizando **listas** para gestionar un sistema de aprendizaje de vocabulario en inglés. El programa permitirá a los usuarios almacenar, buscar, editar y eliminar palabras en inglés junto con su significado en español. Además, implementará un sistema de evaluación en el que las palabras serán preguntadas al usuario, registrando cuántas veces han sido respondidas correcta o incorrectamente.

Requisitos del Programa:

1. Gestión de Vocabulario:

- Permitir **agregar** nuevas palabras en inglés con su significado en español.
- Permitir **buscar** una palabra en la lista y mostrar su significado.
- Permitir **editar** una palabra y su significado.
- Permitir **eliminar** una palabra manualmente.

2. Modo de Evaluación:

- Preguntar al usuario la traducción de palabras en inglés.
- Si el usuario responde correctamente, se incrementa el contador de aciertos consecutivos.
- Si el usuario responde incorrectamente, el contador se reinicia y se incrementa la cantidad de errores de la palabra.
- Las palabras con **más errores registrados** serán preguntadas primero.
- Una palabra será **eliminada automáticamente** cuando haya sido respondida correctamente **5 veces seguidas**.

3. Interfaz en Consola:

- Mostrar un menú con opciones para gestionar palabras y entrar al modo de evaluación.
- Utilizar mensajes claros para la interacción con el usuario.

Ejemplo de Funcionalidad Esperada:

--- Menú Principal ---

1. Agregar nueva palabra
2. Buscar palabra
3. Editar palabra
4. Eliminar palabra
5. Iniciar evaluación
6. Salir

Seleccione una opción: Si el usuario selecciona **5 (Iniciar evaluación)**, se le preguntará:

Traduce al español: "apple"

> Manzana  (Correcto)

Luego de **5 respuestas correctas consecutivas**, el sistema mostrará:

¡Felicidades! Has aprendido la palabra "apple", se eliminará del listado.

Si responde mal:

 Incorrecto. Respuesta correcta: "manzana"

Y el contador de aciertos para esa palabra se reinicia.

Ejercicio #2: Sistema de Gestión de Pedidos en un Restaurante

Desarrolla una aplicación de consola en **Python** utilizando **colas** para gestionar los pedidos en un restaurante. El programa debe simular la atención de clientes, permitiendo que los pedidos se registren en orden de llegada y sean atendidos de manera **FIFO (First In, First Out)**.

Requisitos del Programa:

1. Gestión de platos del menú:

- Antes de poder gestionar los pedidos, se debe disponer de un CRUD para gestionar los diferentes platos del menú, indicando el nombre del plato y su precio.

2. Gestión de Pedidos:

- Permitir que los clientes **realicen pedidos**, ingresando su nombre y el plato solicitado (Debe estar registrado previamente).
- Los pedidos se **almacenarán en una cola** en orden de llegada.
- Permitir **mostrar todos los pedidos pendientes**.

3. Atención de Pedidos:

- Permitir que el sistema **atienda un pedido** (es decir, eliminarlo de la cola).
- Mostrar el nombre del cliente y su pedido cuando este sea atendido.

4. Interfaz en Consola:

- Mostrar un menú con opciones para agregar pedidos, ver la lista de pedidos y atenderlos.
- Utilizar mensajes claros para la interacción con el usuario.

Ejemplo de Funcionalidad Esperada:

--- Menú Principal ---

1. Agregar nuevo pedido
2. Mostrar pedidos pendientes
3. Atender pedido
4. Ganancias
5. Salir

Si el usuario selecciona la opción 1

Ingrese el nombre del cliente: Juan

Ingrese el platillo solicitado: Hamburguesa


Pedido agregado exitosamente.

Si el usuario selecciona **2 (Mostrar pedidos pendientes)**, el sistema muestra:

--- Pedidos Pendientes ---

1. Juan - Hamburguesa
2. María - Pizza
3. Pedro - Ensalada

Si selecciona **3 (Atender pedido)**, el sistema mostrará:

Atendiendo pedido de Juan - Hamburguesa 

Y al mostrar la lista nuevamente, el primer pedido habrá desaparecido.

Si se selecciona la **opcion 4**, se debe retornar el total de los ingresos generados a partir de los pedidos que se han atendido.

Ejercicio #3: Juego de 21 con Pilas

Desarrolla una aplicación en **Python** utilizando **pilas** para simular el **juego de 21 (Blackjack)** entre **N jugadores**. El juego se basa en una **pila de cartas**, donde las cartas son apiladas en orden aleatorio. Cada jugador puede pedir cartas hasta acercarse lo más posible a 21 sin pasarse. Si un jugador **sobrepasa 21, pierde inmediatamente**. En cualquier momento, un jugador puede decidir **no tomar más cartas**. Al final, el jugador con la **suma más cercana a 21** gana.

Si hay **empate**, los jugadores empatados jugarán una ronda extra con la **pila de cartas restante** hasta que haya un ganador.

Reglas y Reglas de Implementación:

1. Gestión de la Pila de Cartas:

- Se utilizarán las **cartas de póker** (2 al 10, J = 10, Q = 10, K = 10, A = 11 o 1).
- Las cartas estarán **barajadas** y almacenadas en una **pila** (LIFO: Last In, First Out).
- Los jugadores **sacan** cartas de la pila cuando las piden.

2. Flujo del Juego:

- Se le pregunta al jugador si quiere jugar o si quiere ver el histórico de ganadores
- Si se indica que se quiere jugar se solicita el número de jugadores y sus nombres.
- Cada jugador toma turnos para **pedir cartas o plantarse**.
- Si un jugador supera **21 puntos**, queda eliminado.
- Cuando todos los jugadores se han plantado, el jugador con el puntaje más cercano a 21 gana.
- En caso de **empate**, los jugadores empatados juegan otra ronda con las cartas restantes en la pila.
- El jugador que gano con su respectivo puntaje debe ser persistido en un archivo con el fin de tener el histórico de todos los ganadores

3. Interfaz en Consola:

- Mostrar mensajes claros para el flujo del juego.
- Mostrar la pila de cartas restante después de cada turno.

Ejemplo de Funcionalidad Esperada:

Bienvenido al juego de 21 🎲

Ingrese el número de jugadores: 3

Jugador 1, ingrese su nombre: Ana

Jugador 2, ingrese su nombre: Luis

Jugador 3, ingrese su nombre: Pedro

--- Turno de Ana ---

Carta obtenida: K (10 puntos)

Carta obtenida: 5 (Total: 15)

¿Quieres otra carta? (s/n): s

Carta obtenida: 3 (Total: 18)

¿Quieres otra carta? (s/n): n

Ana se planta con 18 puntos.

--- Turno de Luis ---

Carta obtenida: J (10 puntos)

Carta obtenida: 8 (Total: 18)

¿Quieres otra carta? (s/n): s

Carta obtenida: 5 (Total: 23) ❌ ¡Perdiste!

--- Turno de Pedro ---

Carta obtenida: 9 (9 puntos)

Carta obtenida: 7 (Total: 16)

¿Quieres otra carta? (s/n): s

Carta obtenida: 4 (Total: 20)

¿Quieres otra carta? (s/n): n

Pedro se planta con 20 puntos.

🎉 ¡Pedro gana con 20 puntos! 🎉

Si hubiera habido un empate, los jugadores empatados jugarían **una ronda más** con la pila de cartas restante.

Finalmente deben considerar lo siguiente:

Se debe documentar el código (Clases, funciones, etc.) según lo visto en clase:

- ✅ **Uso correcto de listas, pilas y colas.**
- ✅ **Buena interacción con el usuario en la consola.**
- ✅ **Código limpio, estructurado y documentado.**