Uso de banderas y bucles `while` en programación, aplicados a menús interactivos inspirados en videojuegos

1. ¿Qué es una bandera en programación?

Una bandera (o flag) es una variable booleana que puede tener solo dos valores: `True` (verdadero) o `False` (falso). Su nombre viene de las banderas reales que se usan para señalar algo: están alzadas o bajadas, encendidas o apagadas.

En programación, las banderas nos sirven para recordar si algo ya pasó o no, si una condición se cumple o no, o si debemos activar o desactivar alguna funcionalidad. Por ejemplo:

- ¿El jugador ya derrotó al jefe final? → `jefe_derrotado = True`
- ¿El cofre ya fue abierto? → `cofre_abierto = False`
- ¿El menú debe seguir funcionando? → `menu_activo = True`

Las banderas son especialmente útiles porque nos permiten controlar el flujo de nuestro programa de manera clara y sencilla.

2. Relación entre bucles `while` y banderas

Un bucle `while` ejecuta un bloque de código mientras una condición sea verdadera. Cuando combinamos `while` con banderas, podemos mantener un programa funcionando hasta que decidamos cambiarlo.

El patrón más común es:

```
continuar = True

while continuar:
    # Aquí va el código que se repite
    print("Menú principal")
    print("1. Opción 1")
    print("2. Salir")

    opcion = input("Elige una opción: ")

    if opcion == "2":
        continuar = False  # Cambiamos la bandera para salir
```

Cuando `continuar` se vuelve `False`, el bucle termina y el programa continúa con lo que venga después.

3. Ejemplos inspirados en videojuegos

Pokémon: Entrenador con el que solo puedes luchar una vez

```
ya_luche_con_brock = False
```

```
while True:
    print("Te encuentras con Brock")

    if ya_luche_con_brock == False:
        print("¡Brock quiere luchar!")
        input("Presiona ENTER para luchar...")
        print("¡Ganaste contra Brock!")
        ya_luche_con_brock = True
    else:
        print("Brock: Ya luchamos antes, ¡sigue tu camino!")

    continuar = input("¿Seguir explorando? (s/n): ")
    if continuar == "n":
        break
```

Zelda: Cofre que puede abrirse solo una vez

```
cofre_abierto = False

while True:
    print("Ves un cofre dorado")

    if cofre_abierto == False:
        abrir = input("¿Abrir cofre? (s/n): ")
        if abrir == "s":
            print("¡Encontraste una espada maestra!")
            cofre_abierto = True
    else:
        print("El cofre está vacío")

    salir = input("¿Salir de la habitación? (s/n): ")
    if salir == "s":
        break
```

Minecraft: Monstruos que aparecen solo si es de noche

```
es_de_noche = True

while True:
    print("Explorando el mundo...")

    if es_de_noche == True:
        print("¡Aparecen zombies y esqueletos!")
        print("Tienes que defenderte")
    else:
        print("Es de día, puedes explorar tranquilo")

    cambiar_hora = input("¿Cambiar hora del día? (s/n): ")
    if cambiar_hora == "s":
        es_de_noche = not es_de_noche  # Cambia de día a noche o viceversa
```

```
    salir = input("¿Dejar de jugar? (s/n): ")
    if salir == "s":
        break
```

Among Us: Jugador que puede moverse solo si está vivo

```
esta_vivo = True

while True:
    print("Estás en la nave espacial")

    if esta_vivo == True:
        print("Puedes moverte y hacer tareas")
        accion = input("¿Qué hacer? (tareas/reportar/salir): ")
        if accion == "reportar":
            print("¡Te eliminaron! Ya no puedes hacer tareas")
            esta_vivo = False
    else:
        print("Estás eliminado, solo puedes observar")

    if input("¿Seguir jugando? (s/n): ") == "n":
        break
```

4. Limpiar la pantalla en terminal

Para que nuestros menús se vean más limpios y profesionales, podemos limpiar la pantalla al inicio de cada iteración:

```
import os

# Esta línea limpia la pantalla
os.system('cls' if os.name == 'nt' else 'clear')
```

Esta función detecta si estamos en Windows (`'cls'`) o en Mac/Linux (`'clear'`) y ejecuta el comando correcto.

Ejemplo de uso en un menú:

```
import os
continuar = True

while continuar:
    # Limpiar pantalla al inicio
    os.system('cls' if os.name == 'nt' else 'clear')

    print("=== MENÚ PRINCIPAL ===")
    print("1. Ver inventario")
    print("2. Salir")

    opcion = input("Elige una opción: ")
```

```
    if opcion == "2":
        continuar = False
```

El `clear screen` es opcional, pero hace que la experiencia del usuario sea mucho mejor.

5. Ejercicio práctico: Menú tipo Pokémon

Instrucciones para el estudiante:

Crear un menú que funcione mientras el jugador no seleccione la opción "Salir".

Este menú debe tener las siguientes opciones:

- Ver equipo Pokémon (mostrar una lista con 3 Pokémon)
- Ver Pokédex (mostrar una lista con 3 Pokémon distintos)
- Guardar progreso (mostrar un mensaje que diga "Progreso guardado")
- Salir del juego

Si el estudiante entra a "Equipo Pokémon", debe poder ver los Pokémon y luego volver al menú principal. Lo mismo en cada sección. No se requiere modificar datos ni guardar nada, solo mostrar información y volver al menú.

Tips para resolverlo:

- Usar un bucle `while` con una bandera booleana.
- Usar `input()` para capturar la opción del usuario.
- Usar `elif` para mostrar el contenido correcto según la opción.
- Agregar una opción para "Volver al menú principal" en cada submenú (por ejemplo, presionar ENTER para continuar).
- Si quieres, puedes usar `clear screen` al inicio del bucle para que se vea más limpio.

6. Solución propuesta (no mostrar en clase de inmediato)

```
import os

# Bandera para controlar el menú principal
menu_activo = True

# Listas con datos de ejemplo
mi_equipo = ["Pikachu", "Charizard", "Blastoise"]
pokedex = ["Bulbasaur", "Squirtle", "Pidgey"]

while menu_activo:
    # Limpiar pantalla (opcional)
    os.system('cls' if os.name == 'nt' else 'clear')

    # Mostrar menú principal
    print("=== CENTRO POKÉMON ===")
    print("1. Ver equipo Pokémon")
    print("2. Ver Pokédex")
    print("3. Guardar progreso")
    print("4. Salir del juego")
```

```
    print("==================")

    # Capturar opción del usuario
    opcion = input("Elige una opción (1-4): ")

    # Procesar la opción elegida
    if opcion == "1":
        print("\n=== TU EQUIPO POKÉMON ===")
        for i, pokemon in enumerate(mi_equipo, 1):
            print(f"{i}. {pokemon}")
        print("==========================")
        input("\nPresiona ENTER para volver al menú...")

    elif opcion == "2":
        print("\n=== POKÉDEX ===")
        for i, pokemon in enumerate(pokedex, 1):
            print(f"{i}. {pokemon}")
        print("===============")
        input("\nPresiona ENTER para volver al menú...")

    elif opcion == "3":
        print("\n¡Progreso guardado!")
        print("Tu aventura ha sido guardada correctamente.")
        input("\nPresiona ENTER para volver al menú...")

    elif opcion == "4":
        print("\n¡Gracias por jugar!")
        print("¡Hasta la próxima, entrenador!")
        menu_activo = False  # Cambiar bandera para salir del bucle

    else:
        print("\nOpción no válida. Por favor elige un número del 1 al 4.")
        input("Presiona ENTER para continuar...")

print("El juego ha terminado.")
```

Explicación del código:

- `menu_activo = True`: Bandera que mantiene el menú funcionando
- `while menu_activo:` : El bucle continúa mientras la bandera sea True
- `os.system(...)`: Limpia la pantalla al inicio de cada iteración
- `if opcion == "4":` : Cuando el usuario elige salir, cambiamos `menu_activo = False`
- `input("Presiona ENTER...")`: Pausa el programa para que el usuario pueda leer antes de volver al menú
- Cada opción muestra su contenido y luego vuelve al menú principal