UNIVERSIDAD CATOLICA DEL MAULE

Tragamonedas

P3

Facultad Ciencias de la Ingeniería

Ingeniería Civil Informática

Programación

Hugo Araya Carrasco y Luis Ponce Rosales

Javier Fuentes

Miércoles 04 de Diciembre

Introducción:

El presente código, hecho en lenguaje Python, implementa una simulación de un juego de tragamonedas, generalmente encontrado en casinos, donde el jugador realiza apuestas con el objetivo de obtener premios basados en combinaciones específicas de símbolos. En este tipo de juegos, los resultados de las apuestas dependen de la aparición aleatoria de símbolos en las "ruedas" del tragamonedas, y las ganancias se calculan según las combinaciones obtenidas.

El problema que se resuelve con este código es la creación de un sistema que permita:

1. **Realizar apuestas:** El jugador puede ingresar una cantidad inicial de dinero y realizar apuestas de diferentes montos.
2. **Generación aleatoria de figuras:** El código simula las figuras, en este caso números, que aparecerían en un tragamonedas, utilizando probabilidades ya definidas para determinar qué símbolos aparecerán en cada ronda.
3. **Cálculo de premios:** El código implementa una lógica que evalúa las combinaciones de figuras obtenidas y calcula el premio correspondiente según una serie de reglas ya definidas.
4. **Control del saldo:** El saldo del jugador se actualiza en función de los premios obtenidos o las pérdidas, y se valida que las apuestas no superen el saldo disponible.

De esta manera, el código proporciona una experiencia de juego interactiva y controlada, en la que el jugador puede ver cómo su saldo cambia dependiendo de las combinaciones y recompensas obtenidas en cada ronda. Además, se implementa una restricción que asegura que el jugador no pueda realizar apuestas que superen su saldo, evitando posibles errores o desbalances en el flujo del juego.

Este sistema permite simular un tragamonedas de manera sencilla, pero efectiva, mostrando cómo las probabilidades influyen en los resultados y brindando una mecánica de juego justa.

Diseño:

Modelado del mundo real: En el mundo real, las tragamonedas funcionan generando una serie de símbolos de forma aleatoria, donde cada símbolo tiene una probabilidad determinada de aparecer. Los jugadores realizan apuestas y, dependiendo de la combinación de símbolos obtenidos, pueden ganar premios. En este código, se emula esta dinámica mediante los siguientes componentes:

1. **Probabilidades de figuras:** Se simula el proceso de selección de símbolos o "figuras" con probabilidades predefinidas (por ejemplo, 2% de obtener un 7, 3% de obtener un 6, etc.).

Aquí está la tabla con las probabilidades:

Figura	probabilidad
7	0,02
6	0,03
5	0,05
4	0,10
3	0,10
2	0,20
1	0,20
0	0,30

2. **Apuestas y saldo:** El jugador tiene un saldo que se reduce con cada apuesta y se incrementa si gana un premio.
3. **Premios:** El cálculo del premio se realiza en función de las combinaciones de figuras obtenidas por el jugador, con diferentes multiplicadores según la cantidad de figuras coincidentes.

Aquí está la tabla con los multiplicadores:

Configuración	Factor de ganancia
777	x 1000
666	x 250
555	x 150
444 ó 333	x 50
222 ó 111	x 30
000	x 20
Dos 5, dos 6 ó dos 7	x 10
Un 7	x 5
Un 5 ó 6	x 2

Descomposición en subproblemas:

Generación de figuras aleatorias: El primer subproblema es generar una figura aleatoria para cada una de las tres posiciones en la tragamonedas. Se utiliza una distribución de probabilidad para determinar cuál de las figuras se seleccionará en cada intento. Este subproblema es resuelto por la función *"generar_figura()"*, que se basa en la función *"random.choices()"* para elegir las figuras según las probabilidades.

Cálculo de premios: El siguiente subproblema es calcular el premio basado en las figuras obtenidas. Se debe verificar cuántas veces aparece cada figura y aplicar los multiplicadores establecidos. Este cálculo se realiza mediante la función *"calcular_apuesta()"*, que evalúa las combinaciones y calcula el premio correspondiente.

Gestión del saldo y apuestas: Otro subproblema importante es la gestión del saldo del jugador. El saldo inicial se ingresa al principio, y después de cada ronda, el saldo se ajusta y se muestra según si el jugador gana o pierde. Este proceso se maneja dentro del ciclo principal, que se repite mientras el saldo sea suficiente para realizar apuestas. Si el saldo es insuficiente, el juego termina.

Interacción con el usuario: El último subproblema es la interacción con el jugador, que debe ingresar su monto de apuesta y decidir si desea continuar jugando. La función principal "*tragamonedas()"* gestiona este flujo, mostrando las opciones de apuesta, la cantidad de dinero que el jugador tiene, y la posibilidad de continuar o terminar el juego.

Algoritmos y estructuras de datos

Algoritmo de selección de figuras: Se utiliza la función *"random.choices()"* para simular la aleatoriedad de la selección de figuras según las probabilidades definidas. Esta función selecciona elementos de una lista de acuerdo con una distribución de probabilidades, modelando el comportamiento aleatorio de las tragamonedas.

Algoritmo de cálculo de premio: Para determinar el premio, el código recorre las figuras obtenidas y cuenta cuántas veces aparece cada número usando *"figuras.count()"*. Luego, según las combinaciones de figuras, se aplica el multiplicador adecuado para calcular el premio.

Estructura de datos:

Diccionario *"numeros"*: Este diccionario le otorga a cada figura (0-7) su respectiva probabilidad de aparición. Esta estructura es útil para gestionar y actualizar las probabilidades de manera centralizada.

Lista *"figuras"*: Esta lista almacena las figuras obtenidas en cada ronda. Es usada para contar las ocurrencias de cada figura y calcular el premio.

Variables de control: El saldo y las apuestas son variables que gestionan el flujo del juego. El saldo del jugador se actualiza después de cada ronda, y las apuestas son verificadas antes de permitir su aceptación.

Solución General

A través de la combinación de estos subproblemas y la implementación de los algoritmos adecuados, se ha logrado una solución que simula el comportamiento de un tragamonedas. La solución abarca todos los aspectos importantes del juego, incluyendo la generación de figuras aleatorias, el cálculo de premios, la gestión del saldo y la interacción con el jugador.

Así, el código permite que el jugador pueda disfrutar de una simulación de tragamonedas realista en la que se respetan las probabilidades, se calculan premios de forma correcta y se mantiene el control sobre el saldo y las apuestas.

Detalles de Implementación del Código

La implementación de este código en Python 3 se centra en simular un juego de tragamonedas, donde el jugador tiene un saldo inicial y puede realizar apuestas con base en probabilidades predeterminadas para generar una figura. Dependiendo de la combinación de figuras obtenidas, el jugador gana o pierde dinero. A continuación, se detallan las distintas partes del código:

1. Generación de Figuras Aleatorias "*generar_figura()*":

Algoritmo: La función "*generar_figura*()" genera una figura aleatoria utilizando la función *"random.choices()".* Este algoritmo se basa en las probabilidades definidas en un diccionario llamado *"números"*, donde cada clave representa una figura (de 0 a 7) y el valor asociado es la probabilidad de que esa figura se elija.

Detalles Técnicos:

- Se usa el diccionario *"números"* para almacenar las probabilidades y las figuras posibles.
- Las figuras se almacenan en una lista figuras y las probabilidades en una lista probabilidades.
- La función *"random.choices()"* se utiliza con las listas de figuras y probabilidades para seleccionar una figura aleatoriamente según las probabilidades definidas. Esto garantiza que las probabilidades de cada figura se respeten al generar el resultado.
- La función devuelve una figura seleccionada aleatoriamente.

```python
#genera una figura respetando las probabilidades indicadas
def generar_figura():
    numeros = {
        7: 0.02,  # 2% probabilidad
        6: 0.03,  # 3% probabilidad
        5: 0.05,  # 5% probabilidad
        4: 0.10,  # 10% probabilidad
        3: 0.10,  # 10% probabilidad
        2: 0.20,  # 20% probabilidad
        1: 0.20,  # 20% probabilidad
        0: 0.30   # 30% probabilidad
    }
    figuras = list(numeros.keys()) #crea lista de los numeros
    probabilidades = list(numeros.values()) #crea lisa de las probabilidades
    return random.choices(figuras, probabilidades, k=1)[0] #elige una figura dependiendo de las probabilidades
```

2. Cálculo de Premio *"(calcular_apuesta()"*:

Algoritmo: La función *"calcular_apuesta()"* calcula el premio que recibe el jugador en función de las figuras obtenidas. Utiliza una serie de estructuras if para determinar la cantidad de veces que aparece cada figura (de 0 a 7) y aplicar el multiplicador correspondiente.

Detalles Técnicos: Se utiliza *"count()"* en la lista para contar cuántas veces aparece cada figura en el resultado de las tres figuras obtenidas.

En función de cuántas veces aparece cada figura, se calcula el premio correspondiente aplicando los multiplicadores predefinidos.

La función devuelve el valor del premio en función de la combinación de figuras obtenida.

```python
#Calcula el premio en base a las figuras/numeros obtenidas/os
def calcular_apuesta(figuras, apuesta):
    if figuras.count(7) == 3:
        return apuesta * 1000
    elif figuras.count(7) == 2:
        return apuesta * 10
    elif figuras.count(6) == 2:
        return apuesta * 10
    elif figuras.count(5) == 2:
        return apuesta * 10
    elif figuras.count(7) == 1:
        return apuesta * 5
    elif figuras.count(6) == 3:
        return apuesta * 250
    elif figuras.count(5) == 3:
        return apuesta * 150
    elif figuras.count(4) == 3:
        return apuesta * 50
    elif figuras.count(3) == 3:
        return apuesta * 50
    elif figuras.count(2) == 3:
        return apuesta * 30
    elif figuras.count(1) == 3:
        return apuesta * 30
    elif figuras.count(0) == 3:
        return apuesta * 20
    else:
        return 0
```

3. Función Principal del Juego *"tragamonedas()"*:

Algoritmo: La función *"tragamonedas()"* es la que gestiona el flujo principal del juego. Permite al jugador ingresar un saldo inicial, realizar apuestas y ver las figuras generadas y el resultado de la apuesta.

Resumen técnico:

- El saldo inicial del jugador es ingresado por medio de *"input()"*, y se almacena como una variable entera saldo.
- El jugador puede apostar $100, $500 o $1000, y se verifica que la apuesta sea válida y no exceda el saldo disponible.
- Se generan tres figuras utilizando la función *"generar_figura()"*, y se muestra el resultado al jugador.
- El premio se calcula con la función *"calcular_apuesta()"* y se actualiza el saldo en base de si el jugador gana o pierde.
- Si el saldo es insuficiente para realizar una apuesta, el juego termina.
- El jugador tiene la opción de continuar jugando o salir al final de cada ronda.

```python
#Función principal para simular el tragamonedas.
def tragamonedas():
    saldo = int(input("Ingrese el monto inicial: "))

    for _ in range(10000):
        if saldo < 100:
            print("Saldo insuficiente. Intente de nuevo.")
            break

        print(f"Saldo actual: ${saldo}")
        apuesta = int(input("Ingrese su apuesta ($100, $500, $1000): "))
        if apuesta not in [100, 500, 1000]:
            print("Apuesta no válida. Intente de nuevo.")
            continue
        if apuesta > saldo:
            print("Saldo insuficiente. Intente de nuevo.")
            continue

        # Generar las figuras
        figuras = [generar_figura() for _ in range(3)]
        print("Figuras obtenidas:", " - ".join(map(str, figuras))) #convierte cada numero en la lista figuras en una cadena de texto separandose con

        # Calcular el premio
        premio = calcular_apuesta(figuras, apuesta)
        if premio > 0:
            print(f"¡Ganaste! Premio: ${premio}")
            saldo += premio
        else:
            print(f"Perdiste la apuesta de ${apuesta}")
            saldo -= apuesta
```

```python
        # Verificar si el jugador desea continuar
        continuar = input("¿Desea continuar? (1=SI, 0=NO): ")
        if continuar != "1":
            break

    print(f"Saldo final: ${saldo}")
    print("Gracias por jugar.")
```

4. Estructura del Programa:

El programa está estructurado en funciones que se encargan de tareas específicas:

"generar_figura()": Genera una figura aleatoria basada en probabilidades.

"calcular_apuesta()": Calcula el premio en base a las figuras obtenidas y la apuesta.

"tragamonedas()": Controla el flujo principal del juego, incluyendo las interacciones con el jugador, el saldo y las apuestas.

Al final del archivo, el bloque *if __name__ == "__main__":* asegura que el juego se ejecute solo si el archivo se ejecuta directamente.

Conclusión

El código implementa un juego de tragamonedas donde se simulan figuras aleatorias con probabilidades predefinidas, y se calcula un premio dependiendo de las combinaciones de figuras obtenidas. El jugador interactúa con el juego a través de la consola, apostando y viendo si gana o pierde dinero. Las funciones están organizadas para separar claramente las responsabilidades y asegurar que el flujo del juego sea coherente y controlado.

Limitaciones del Programa:

A pesar de que el programa cumple con los requisitos básicos para simular un juego de tragamonedas con las reglas planteadas, existen ciertas limitaciones tanto sobre el enunciado del problema como a las decisiones tomadas durante la ejecución del código. Estas limitaciones son las siguientes:

Falta de validación de entrada avanzada:
Aunque se valida si la apuesta es de $100, $500 o $1000, el programa no implementa una verificación exhaustiva sobre otros posibles errores de entrada, como la introducción de caracteres no numéricos o valores fuera de los límites.

Estrategia de solución: Se podría mejorar agregándole una función que asegure que el usuario solo ingresa valores válidos y, en caso contrario, le solicite que ingrese un valor correcto.

Reinicio del saldo:
El programa no permite al jugador continuar con el juego en sesiones diferentes sin tener que reiniciar el saldo manualmente al iniciar de nuevo. Si el jugador decide dejar de jugar y regresar más tarde, tendría que ingresar nuevamente el monto inicial o el monto con el que quedó al finalizar la sesión anterior.

Estrategia de solución: Se podría implementar una opción para guardar el saldo en un archivo o base de datos y que el jugador pueda continuar desde el mismo punto en que lo dejó.

Interacción limitada:
El programa depende completamente de la interacción del usuario a través de la consola, lo que limita la accesibilidad y la experiencia del jugador. No se ofrece una interfaz gráfica, lo que haría el juego más atractivo y fácil de usar.

Estrategia de solución: Se podría desarrollar una interfaz gráfica utilizando una librería como Tkinter para mejorar la experiencia del usuario.

Falta de manejo de excepciones:
El programa no maneja correctamente situaciones excepcionales que puedan surgir durante la ejecución, como una entrada inesperada del usuario o un error de cálculo en la función *“random.choices”.*

Estrategia de solución: Se puede implementar un bloque try-except, por ejemplo, para manejar posibles errores en las entradas del usuario y garantizar que el programa no se detenga abruptamente debido a errores.

Límites de jugabilidad:
El número de intentos está limitado a 10000, lo que podría no ser adecuado para todos los usuarios. Esto también puede ser innecesario en un juego simple como este, ya que puede ser confuso.

Estrategia de solución: Se podría modificar para permitir que el jugador juegue hasta que decida terminar, sin necesidad de un límite artificial en el número de intentos.

Lógica del cálculo de premios:
La estructura del cálculo de premios es rígida y no permite una personalización sencilla. El hecho de que solo se verifiquen combinaciones de números exactos en las figuras generadas puede hacer que algunos resultados no sean tan interesantes o variados como podrían ser.

Estrategia de solución: Se podría agregar más reglas o combinaciones de premios para hacer el juego más dinámico y aumentar la variabilidad de los premios obtenidos.

A pesar de estas limitaciones, el programa cumple con la funcionalidad básica de un tragamonedas, pero su extensión y mejora son necesarias para proporcionar una experiencia de usuario más completa y flexible.

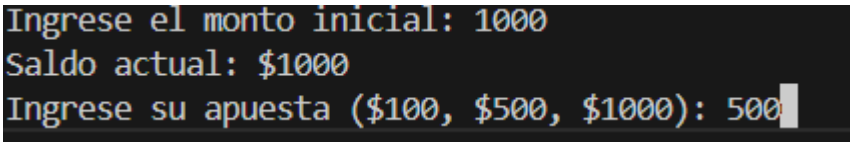
Manual de Usuario: Juego de Tragamonedas

Descripción del Programa: El programa simula un juego de tragamonedas en el que el usuario puede realizar apuestas y obtener premios basados en las figuras generadas en cada jugada. Cada jugada genera tres figuras, y dependiendo de las combinaciones, el jugador puede ganar un premio o perder la cantidad apostada.

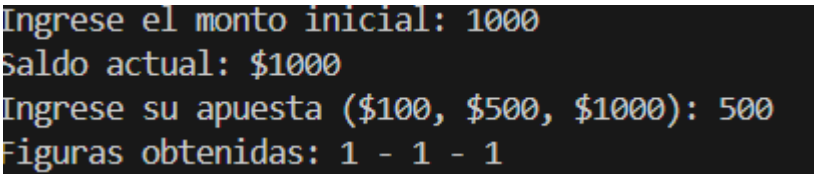
Instrucciones de Uso:

Inicio del Programa: Al ejecutar el programa, se te solicitará que ingreses un monto inicial de dinero. Este monto será tu saldo disponible para realizar apuestas durante el juego.
Ejemplo:

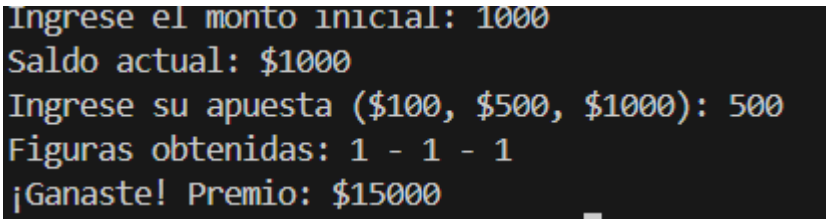
Ingreso de Apuesta: Después de ingresar tu saldo inicial, podrás realizar apuestas. Las apuestas permitidas son de $100, $500 o $1000. El programa te pedirá que selecciones una de estas opciones.
Ejemplo:



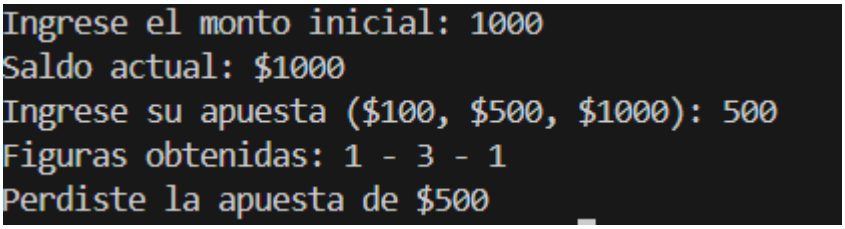
Generación de Figuras: El programa generará tres figuras al azar, que serán mostradas en pantalla. Las figuras pueden ser números del 0 al 7, y dependiendo de la combinación, se calculará el premio.
Ejemplo:



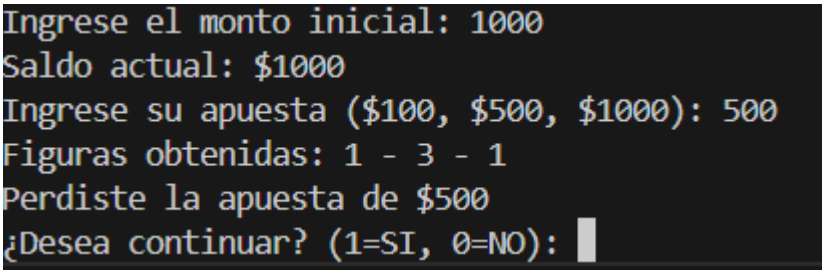
Cálculo de Premio: En base a las figuras obtenidas, el programa calculará el premio o la pérdida. Si obtienes una combinación ganadora, se te mostrará el premio y se sumará a tu saldo. Si no, se restará la cantidad apostada.
Ejemplo de mensaje de ganancia:



Ejemplo de mensaje de pérdida:

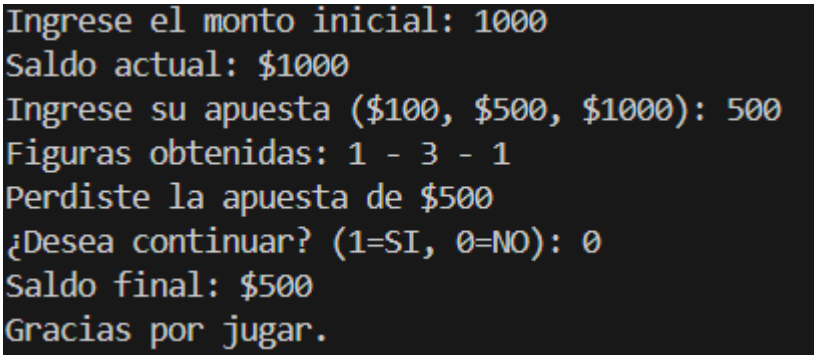


Continuar Jugando: Después de cada jugada, se te preguntará si deseas seguir jugando. Si decides continuar, el programa continuará generando figuras y calculando premios hasta que decidas salir o se te acabe el saldo.
Ejemplo:

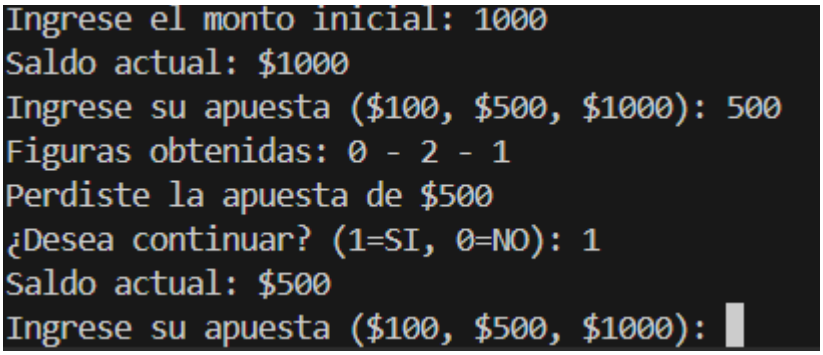


Si deseas terminar de jugar, ingresa 0 y el programa finalizará.

Finalización del Juego: El juego terminará cuando decidas no continuar o cuando tu saldo sea insuficiente para realizar una apuesta. Al final, se mostrará el saldo restante.
Ejemplo de mensaje final:



Si decides seguir jugando, te preguntará nuevamente cuando deseas apostar y así hasta que decidas dejar de jugar o se te acabe el saldo.



Reglas del Juego:

El juego genera tres figuras en cada jugada. Las figuras son números del 0 al 7, y cada uno tiene una probabilidad de ser seleccionado.

Las combinaciones ganadoras de las figuras y los premios son los siguientes:

Tres 7's: Premio de 1000 veces la apuesta
Tres 6's: Premio de 250 veces la apuesta
Tres 5's: Premio de 150 veces la apuesta
Tres 4's: Premio de 50 veces la apuesta
Tres 3's: Premio de 50 veces la apuesta
Tres 2's: Premio de 30 veces la apuesta
Tres 1's: Premio de 30 veces la apuesta
Tres 0's: Premio de 20 veces la apuesta
Dos 7's, dos 6´s o dos 5´s: Premio de 10 veces la apuesta
Un 1: Premio de 5 veces la apuesta

Consejos:

- Asegúrate de tener suficiente saldo para realizar apuestas. Si no tienes suficiente dinero, el juego terminará.
- Elige tus apuestas sabiamente, ya que mayores apuestas aumentan las ganancias, pero también el riesgo de perder.

Ejemplo Completo de Ejecución:

```
Ingrese el monto inicial: 1000
Saldo actual: $1000
Ingrese su apuesta ($100, $500, $1000): 500
Figuras obtenidas: 0 - 2 - 1
Perdiste la apuesta de $500
¿Desea continuar? (1=SI, 0=NO): 1
Saldo actual: $500
Ingrese su apuesta ($100, $500, $1000): 100
Figuras obtenidas: 1 - 1 - 2
Perdiste la apuesta de $100
¿Desea continuar? (1=SI, 0=NO): 0
Saldo final: $400
Gracias por jugar.
```

Notas Importantes:

- El programa está diseñado para ser ejecutado en la terminal o consola de Python.
- Asegúrate de tener instalada una versión de Python compatible (recomendado Python 3).
- El saldo mínimo para jugar es de $100, por lo que, si el saldo cae por debajo de esa cantidad, el juego se detendrá, ¡asegúrate de manejar bien tus apuestas!