



Escuela De Ingeniería Y Ciencias

Pensamiento Computacional para la Ingeniería TC1028

Casos de Prueba: Proyecto- Juego Cognitivo

Luis Espinoza Navarrete A01668021

```

+ Proyecto git:(master) x python3 main.py
Ingrese el nombre del jugador 1: Luis
Ingrese el nombre del jugador 2: Manuel
      1      2      3      4      5      6
1  - - - - - -
2  - - - - - -
3  - - - - - -
4  - - - - - -
5  - - - - - -
6  - - - - - -
Turno del jugador Luis
Ingrese la fila y columna de la primera carta
Ingrese la fila: 3
Ingrese la columna: 3
      1      2      3      4      5      6
1  - - - - - -
2  - - - - - -
3  - - 16 - - - -
4  - - - - - -
5  - - - - - -
6  - - - - - -
Ingrese la fila y columna de la segunda carta
Ingrese la fila: 4
Ingrese la columna: 5
      1      2      3      4      5      6
1  - - - - - -
2  - - - - - -
3  - - 16 - - - -
4  - - - - 18 - -
5  - - - - - -
6  - - - - - -
El jugador Luis no ha encontrado una par
Turno del jugador Manuel
Ingrese la fila y columna de la primera carta
Ingrese la fila: _

```

El juego inicia solicitando ambos nombres de jugadores que serán mostrados según sus turnos y a continuación el jugador en este caso “Luis” podrá indicar el número y fila deseado.

### Caso de Prueba 1:

¿Qué pasa si ingresa un carácter no numérico?

```
Turno del jugador Manuel

Ingrese la fila y columna de la primera carta
Ingrese la fila: caracter_no_numérico
Por favor, ingrese un número válido.
Ingrese la fila: _
```

Para estos casos tenemos la función `solicitar_posicion_valida(mensaje)` que hace uso del método `valor.isdigit()` para validar que se reciban dígitos numéricos

```
def solicitar_posicion_valida(mensaje):
    valor = input(mensaje)
    while not valor.isdigit(): # Verificar que la entrada sea un número
        print("Por favor, ingrese un número válido.")
        valor = input(mensaje)
    return int(valor) - 1 # Restamos 1 para ajustar a índices de la matriz
```

Y en consecuencia se solicita una entrada válida

### Caso de Prueba 2:

¿Qué pasa si se ingresa un número de fila o columna mayor o menor al rango permitido?

```
Ingrese la fila: 7 print("Por favor, ingrese un número")
Ingrese la columna: 0 valor = input(mensaje)
Ingrese una fila y columna valida return int(valor) - 1 # Restamos 1 para ajustar a índices de la matriz
Ingrese la fila: 4
Ingrese la columna: 7
Ingrese una fila y columna valida
Ingrese la fila: 0
Ingrese la columna: 4
Ingrese una fila y columna valida
Ingrese la fila:
jugar = 'si'
pares encontrados = 0
```

Para estos casos se ha implementado la función `validar_posicion(fila,columna,m_cuadrada)` que verifica que si las filas o columnas ingresadas estén fuera del rango deseado se solicite nuevamente según:

```

def validar_posicion(fila,columna,m_cuadrada):
    #validar que la fila y columna esten dentro de la matriz
    while fila<0 or fila>=m_cuadrada or columna<0 or columna>=m_cuadrada:
        print("Ingrese una fila y columna valida")
        fila=int(input("Ingrese la fila: "))-1
        columna=int(input("Ingrese la columna: "))-1
    return [fila,columna]

```

### Caso de Prueba 3 y 4:

¿Qué pasa si en su segunda carta ingresa una fila y columna válida pero es la misma que acaba de destapar?

```

Ingrese la fila: 5
Ingrese la columna: 3
      1      2      3      4      5      6
1      -      -      -      -      -      -
2      -      -      -      -      -      -
3      -      -      -      -      -      -
4      -      -      -      -      -      -
5      -      -      12     -      -      -
6      -      -      -      -      -      -
Ingrese la fila y columna de la segunda carta
Ingrese la fila: 5
Ingrese la columna: 3
La casilla ya ha sido destapada
Ingrese la fila y columna valida
Ingrese la fila: _

```

¿Y si en general ya hay una casilla destapada pero se vuelve a solicitar esa posición para destapar?

```

13      1      2      3      4      5      6
14  1      -      -      -      -      -      -
15  2      -      -      -      -      -      -
16  3      -      -      -      -      -      -
17  4      -      -      -      -      18      -
18  5      -      -      -      -      -      -
19  6      18      -      -      -      -      -
20 El jugador Manuel ha encontrado una par
21
22 Desea seguir jugando? si/no: si
23
24 Turno del jugador Luis
25
26 Ingrese la fila y columna de la primera carta
27 Ingrese la fila: 6
28 Ingrese la columna: 1
29 La casilla ya ha sido destapada
30 Ingrese la fila y columna valida
31 Ingrese la fila: 4
32 Ingrese la columna: 5
33 La casilla ya ha sido destapada
34 Ingrese la fila y columna valida
35 Ingrese la fila:

```

Para ambos casos tenemos la misma función que se aplica en cada turno y es `validar_casilla_sin_voltear(simbolo,matriz,fila,columna)`

```

def validar_casilla_sin_voltear(simbolo,matriz,fila,columna):
    #validar que la casilla no haya sido destapada
    while matriz[fila][columna]!=simbolo:
        print("La casilla ya ha sido destapada")
        print("Ingrese la fila y columna valida")
        fila=int(input("Ingrese la fila: "))-1
        columna=int(input("Ingrese la columna: "))-1
    return [fila,columna]

```

Esta función valida que si la posición de la fila y columna en la matriz es distinta al símbolo, en este caso "\*", entonces debe solicitar nuevamente los inputs pues esta casilla ya ha sido volteada anteriormente.

### Caso de Prueba 5:

¿Si responden con algo distinto a “si” o “no” cuando se les pregunta si quieren continuar ?

```
Desea seguir jugando? si/no: s
Ingrese una opcion valida si/no: n
Ingrese una opcion valida si/no: nosesisiosino
Ingrese una opcion valida si/no: _
```

Un pequeño bucle en la parte final del main se encarga de evaluar esos casos según:

```
if num_jugador == 2 and jugar != "no":
    jugar = input("Desea seguir jugando? si/no: ")
    while jugar != "si" and jugar != "no":
        jugar = input("Ingrese una opcion valida si/no: ")
    if jugar == "no":
        validar_final_forzado(pares_1, pares_2, puntos_1, puntos_2, cant_bonus_1, cant_b
```

### Caso de Prueba 6:

Si se responde negativamente el juego se termina y siguiendo las instrucciones, se muestra la cantidad de pares y los puntos obtenidos de cada jugador.

```
Ingrese una opcion valida si/no: no
El juego se ha detenido
Pares jugador 1: 0
Pares jugador 2: 1
Puntos jugador 1: 0
Puntos jugador 2: 100
Proyecto git:(master) x
```

La función que se encarga de validar este final forzado es

`validar_final_forzado(pares_1,pares_2,puntos_1,puntos_2,cant_bonus_1,cant_bonus_2)`

```
def validar_final_forzado(pares_1,pares_2,puntos_1,puntos_2,cant_bonus_1,cant_bonus_2):
    #validar si el juego ha terminado forzadamente y mostrar ganador
    print("El juego se ha detenido")
    print("Pares jugador 1: ",pares_1)
    print("Pares jugador 2: ",pares_2)
    print("Puntos jugador 1: ",puntos_1)
    print("Puntos jugador 2: ",puntos_2)
    if cant_bonus_1:
        print("Bonus points jugador 1: ",cant_bonus_1*1000)
    if cant_bonus_2:
        print("Bonus points jugador 2: ",cant_bonus_2*1000)
```

### Caso de Prueba 7:

14		1	2	3	4	5	6		22 65
15	1	18	12	9	7	2	5		
	2	16	10	1	11	14	4		52 42
16	3	12	8	3	17	11	1		
	4	6	15	4	7	3	13		63 46
17	5	14	15	8	6	17	9		
	6	18	16	13	5	10	2		66 15
18	El jugador manuel ha encontrado una par								
	El juego ha terminado								24 35
	El ganador es el manuel								
	Bonus points multiplicados por el total de pares destapados: 18000								

Cuando se termina el juego se muestra al ganador con su nombre, en este caso “manuel” y la cantidad de bonus points multiplicados por el total de pares destapados como indican las instrucciones del proyecto.