

**Materia:** Programación Orientada a Objetos (COM102)

**Profesor:** Mtro. Giancarlo Xavier Benítez Villacreses

**Fecha de entrega:** 12/05/2022

**Ciclo:** 1222

**Nombre del proyecto:** **BlackJack de otra galaxia**

|  |  |  |
| --- | --- | --- |
| **Miembros del Equipo** | | |
| **ID** | **Nombre** | **Carrera** |
| 0241823 | Enrique Ulises Báez Gómez Tagle | IID&C |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Rúbricas** | | |
| **ID** | **1-identify** | |
| **IP** | **ASA** |
| 0241823 |  |  |
|  |  |  |

**Descripción del proyecto:**

Se diseñó un programa que permite a un usuario jugar Blackjack (21) contra la computadora.

Clase Bajara:

* La clase baraja crea un arreglo “mazo” con las cartas (A, 2-10, J, Q, K) de los 4 palos (C, P, T, D), dando como resultado un arreglo de 52 cartas.
* Cada vez que se crea un mazo, se barajea en automático
* La clase baraja posee un método para pedir carta, el cual retorna la última carta del arreglo “mazo” y la descuenta del arreglo total (51, 50, 49, 48…. Etc.)
* Ninguna carta sale más de una vez durante la ejecución del programa o hasta que se cree una nueva baraja y se descarte la anterior.

Clase Jugador:

* Al inicio del juego se solicita el nombre del jugador (longitud > 3) y el monto inicial (mayor a cero) con sus respectivos validadores.
* La misma clase jugador se utiliza para el croupier (la computadora) de manera automática.
* El jugador tiene un arreglo llamado “mano” que almacena las cartas y un método que devuelve el conteo de dichas cartas.
* Tiene un método para realizar la apuesta y un método para realizar el depósito de fichas.

Ronda:

* Al iniciar la ronda, se solicita el monto de la apuesta. La apuesta no puede ser mayor a los fondos disponibles del jugador ni ser negativa.
* En caso de que el jugador no tenga fondos o no sean suficientes, da la opción para depositar más fichas y volver a realizar la apuesta.
* En caso de que el mazo de cartas tenga menos de 15 cartas, se descarta.
* Al perder, se debe descuenta el monto de la apuesta del balance del jugador.
* La mano del jugador siempre es visible, mientras que en la mano del croupier solo 1 carta es visible para el jugador.
* El jugador puede escoger pedir N cartas o quedarse.

Resumen:

* Cuando finaliza la ronda, se realiza la comparación:
  + Si el jugador supera 21, derrota instantánea.
  + Si el jugador y el croupier no superan 21 y tienen el mismo conteo, se decreta empate.
  + Si el jugador tiene exactamente 21 y supera al croupier, el jugador gana con BlackJack.
  + Si el jugador es menor a 21 pero mayor al croupier, el jugador gana.
  + Si el croupier es menor o igual a 21 y el jugador no supera al croupier, entonces el jugador pierde lo apostado.
  + Si el croupier supera 21, victoria instantánea del jugador.
* Al finalizar la comparación, se realiza el depósito correspondiente.
* Se revelan las manos de croupier y del jugador, y se muestra el nuevo balance del jugador.
* Se da la opción para jugar otra ronda o salir.

**Problemas encontrados y solución aplicada:**

Debido al diferente planteamiento de las versiones, se adecuó cada una de acuerdo a los requerimientos solicitados para esa entrega.

**CLI VERSION:**

1. Main.py

Se importa todo lo de los demás archivos así como las librerías time y os. También se define una función para limpiar la consola.

*from* Clases.Jugador *import* \*  
*from* Clases.Jugar *import* \*  
*from* Clases.Baraja *import* \*  
*import* time  
*import* os  
  
*def* clear\_console():  
 *if* os.name == 'nt':  
 os.system('CLS')  
 *if* os.name == 'posix':  
 os.system('clear\_console')

runGame = *True*  
band = *False*

Se crean dos variables banderas que controlaran nuestro flujo principal.

Se crea un objeto baraja, se solicita el nombre y monto (con validaciones) y se crean los objetos player (p) y casa (h).

print("♠ ♥ ♣ ♦ ♠ ♥ ♣ ♦ ♠ ♥ ♣ ♦ ♠ ♥ ♣ ♦ BIENVENIDO A BLACKJACK ♠ ♥ ♣ ♦ ♠ ♥ ♣ ♦ ♠ ♥ ♣ ♦ ♠ ♥ ♣ ♦")  
print("\nCaptura de datos")  
print("------------------------------")  
b = Baraja()  
name = validar\_nombre()  
money = validar\_monto\_inicial()  
p = Jugador(name, money)  
h = Jugador("Croupier", 1000000)

Menú:

*while* runGame *is True*:  
  
 print("\nMENU PRINCIPAL")  
 print("-------------------\n")  
 print("1) Elegir el número de barajas para juego")  
 print("2) Jugar solo contra la casa")  
 print("3) Salir")  
 op = int(input("Seleccione una opción: "))  
 time.sleep(3)  
 clear\_console()  
  
 *if* op == 1:  
 b.numdecks()  
 b.barajear()  
 band = *True*  
  
 *elif* op == 2:  
 *if* band *is True*:  
 jugar(b, p, h)  
 p.detalles()  
 *else*:  
 print("No ha elegido el número de barajas")  
 print("Por favor, vuelva a intentarlo")  
 print("\n")  
 *continue*  
 *elif* op == 3:  
 runGame = *False*  
print("Gracias por jugar con nosotros")  
 print("♠ ♥ ♣ ♦ ♠ ♥ ♣ ♦ ♠ ♥ ♣ ♦ ♠ ♥ ♣ ♦ SALIENDO DE BLACKJACK ♠ ♥ ♣ ♦ ♠ ♥ ♣ ♦ ♠ ♥ ♣ ♦ ♠ ♥ ♣ ♦")  
 print("------------------------------")  
  
 *else*:  
 print("Opción inválida")  
 print("Por favor, vuelva a intentarlo")  
 print("\n")  
 *continue*

1. Baraja.py

Se importa numpy y se crea el método para pedir carta.

*import* numpy *as* np  
  
*def* pedir\_carta(baraja, player):  
 *if* len(baraja.decks) > 15:  
 card = baraja.decks[-1]  
 baraja.decks.pop()  
 player.mano.append(card)  
 *else*:  
 *pass*

Creamos la clase baraja con todas las cartas posibles, el método para elegir número de barajas y el método para barajear.

*class* Baraja:  
 hearts = ["♥A", "♥2", "♥3", "♥4", "♥5", "♥6", "♥7", "♥8", "♥9", "♥10", "♥jack", "♥queen", "♥king"]  
 clubs = ["♣A", "♣2", "♣3", "♣4", "♣5", "♣6", "♣7", "♣8", "♣9", "♣10", "♣jack", "♣queen", "♣king"]  
 diamonds = ["♦A", "♦2", "♦3", "♦4", "♦5", "♦6", "♦7", "♦8", "♦9", "♦10", "♦jack", "♦queen", "♦king"]  
 spades = ["♠A", "♠2", "♠3", "♠4", "♠5", "♠6", "♠7", "♠8", "♠9", "♠10", "♠jack", "♠queen", "♠king"]

deck = [hearts, clubs, diamonds, spades]  
 decks = []  
  
 *def* numdecks(*self*):  
 print("Elija el número de barajas para jugar\n")  
 print("1 Baraja")  
 print("2 Barajas")  
 print("3 Barajas")  
 print("4 Barajas")  
 print("5 Barajas")  
 print("6 Barajas")  
 print("7 Barajas")  
 print("8 Barajas")  
 option = int(input("Opción: "))  
 *for* n *in* range(option):  
 *for* i *in* range(4):  
 *for* j *in* range(13):  
 *self*.decks.append(*self*.deck[i][j])

*def* barajear(*self*):  
 np.random.shuffle(*self*.decks)

1. Game.py

Se importa time, termcolor y random, así como las funciones definidas en los otros archivos. Y se definen los carteles para victoria, derrota y empate.

*import* time  
*import* random  
  
*from* termcolor *import* colored  
*from* Clases.Jugador *import* \*  
*from* Clases.Baraja *import* \*  
  
  
*def* print\_win():  
 print(colored('GANASTE', 'green'))  
  
  
*def* print\_loose():  
 print(colored('PERDISTE', 'red'))  
  
  
*def* print\_tie():  
 print(colored('EMPATE', 'blue'))

Se define la función añadir carta que pedirá carta en caso de que todavía haya más de 15.

*def* add\_card(baraja, player):  
 *if* len(baraja.decks) > 15:  
 pedir\_carta(baraja, player)  
 *else*:  
 *pass*

Posteriormente la funcion checar puntaje, pregunta por otra carta y revisa cada una de los criterios de victoria/derrota, para así ajustar las banderas correspondientes y realizar el depósito pertinente.

*def* check\_points(baraja, house, player\_one, apuesta, countplayer\_one, count\_house):  
 runcheck\_points = *True*  
blackjack = *False*  
loose = *False*  
loose\_h = *False*  
  
 *if* len(baraja.decks) <= 15:  
 runcheck\_points = *False*  
  
 *while* runcheck\_points *is True*:  
 *if* countplayer\_one == 21:  
 blackjack = *True*  
runcheck\_points = *False*  
 *elif* countplayer\_one > 21:  
 print\_loose()  
 runcheck\_points = *False*  
 *else*:  
 print("La carta de la casa es: ", house.mano[0])  
 print("Tus cartas son: ", player\_one.mano)  
 print("¿Quieres pedir otra carta?")  
 print("1) Si")  
 print("2) No")  
 player\_option = int(input())  
  
 *if* player\_option == 1:  
 add\_card(baraja, player\_one)  
 countplayer\_one = check\_value(player\_one.mano)  
 *if* countplayer\_one > 21:  
 loose = *True*  
runcheck\_points = *False*  
 *else*:  
 *for* i *in* range(5):  
 *if* count\_house < 17:  
 add\_card(baraja, house)  
 count\_house = check\_value(house.mano)  
 *if* count\_house > 21:  
 loose\_h = *True*  
 *else*:  
 runcheck\_points = *False*  
  
print("Mano jugador: ", player\_one.mano, " = ", countplayer\_one, " pts")  
 print("Mano casa: ", house.mano, " = ", count\_house, " pts")  
  
 *if* blackjack:  
 print\_win()  
 player\_one.ganadas += 1  
 player\_one.money += apuesta  
 player\_one.detalles()  
  
 *elif* loose\_h:  
 print\_win()  
 player\_one.ganadas += 1  
 player\_one.detalles()  
 *else*:  
 *if not* loose:  
 *if* countplayer\_one > count\_house:  
 print\_win()  
 player\_one.ganadas += 1  
 player\_one.detalles()  
  
 *elif* countplayer\_one < count\_house:  
 print\_loose()  
 player\_one.perdidas += 1  
 player\_one.money -= apuesta  
 player\_one.detalles()  
 *else*:  
 print\_tie()  
 player\_one.empates += 1  
 player\_one.detalles()  
 *else*:  
 print\_loose()  
 player\_one.perdidas += 1  
 player\_one.money -= apuesta  
 player\_one.detalles()  
  
 time.sleep(3)  
 print("Cartas restantes: ", len(baraja.decks))

Por último, la función start game reparte las cartas a los jugadores. Después saca el puntaje individual (checar valor) y al final revisa quien ganó.

*def* start\_game(baraja, player\_one, house):  
 apuesta = player\_one.apostar()  
 player\_one.mano = []  
 house.mano = []  
  
 *if* len(baraja.decks) > 15:  
 *for* i *in* range(4):  
 *if* i < 2:  
 random\_card = random.choice(baraja.decks)  
 house.mano.append(random\_card)  
  
 *for* e *in* range(len(baraja.decks)):  
 *if* baraja.decks[e] == random\_card:  
 baraja.decks.pop(e)  
 *break*  
 *else*:  
 random\_card = random.choice(baraja.decks)  
 player\_one.mano.append(random\_card)  
  
 *for* e *in* range(len(baraja.decks)):  
 *if* baraja.decks[e] == random\_card:  
 baraja.decks.pop(e)  
 *break*  
  
count\_house = check\_value(house.mano)  
 countplayer\_one = check\_value(player\_one.mano)  
  
 check\_points(baraja, house, player\_one, apuesta, countplayer\_one, count\_house)  
  
 *else*:  
 *pass*

1. Jugador.py

Métodos para validar nombre, monto inicial, monto de apuesta, monto de fichas y el método que lleva la puntuación de las cartas.

*def* validar\_nombre():  
 run\_name = *True*  
 *while* run\_name *is True*:  
 n = input("Ingrese su nombre: ")  
 *if* len(n) < 3:  
 print("El nombre es muy corto")  
 *else*:  
 *return* n  
  
  
*def* validar\_monto\_inicial():  
 run\_monto = *True*  
 *while* run\_monto *is True*:  
 n = input("Ingrese el monto inicial: ")  
 *if* n.isdigit():  
 n = float(n)  
 *return* n  
 *else*:  
 print("Monto inválido")  
  
  
*def* validar\_monto\_apuesta():  
 run\_apuesta = *True*  
 *while* run\_apuesta *is True*:  
 n = input("Ingrese el monto a apostar: ")  
 *if* n.isdigit():  
 n = float(n)  
 *return* n  
 *else*:  
 print("Monto inválido")  
  
  
*def* validar\_monto\_fichas():  
 run\_fichas = *True*  
 *while* run\_fichas *is True*:  
 print("Cada ficha equivale a $100")  
 fichas = input("Ingrese el número de fichas a depositar: ")  
 *if* fichas.isdigit():  
 fichas = int(fichas)  
 money = fichas \* 100  
 *return* money  
 *else*:  
 print("Número inválido")  
  
  
*def* check\_value(cards):  
 count = 0  
 *for* i *in* range(len(cards)):  
 *if* "2" *in* cards[i]:  
 count += 2  
 *elif* "3" *in* cards[i]:  
 count += 3  
 *elif* "4" *in* cards[i]:  
 count += 4  
 *elif* "5" *in* cards[i]:  
 count += 5  
 *elif* "6" *in* cards[i]:  
 count += 6  
 *elif* "7" *in* cards[i]:  
 count += 7  
 *elif* "8" *in* cards[i]:  
 count += 8  
 *elif* "9" *in* cards[i]:  
 count += 9  
 *elif* "10" *in* cards[i]:  
 count += 10  
 *elif* "jack" *in* cards[i]:  
 count += 10  
 *elif* "queen" *in* cards[i]:  
 count += 10  
 *elif* "king" *in* cards[i]:  
 count += 10  
 *elif* "A" *in* cards[i]:  
 *if* count >= 11:  
 count += 1  
 *else*:  
 count += 11  
 *return* count

Clase Jugador con su constructor, su cartel de monto, su cartel de información completa, método apostar y método depositar fichas.

*class* Jugador:  
  
 *def* \_\_init\_\_(*self*, name, money):  
 *self*.nombre = name  
 *self*.money = money  
 *self*.ganadas = 0  
 *self*.perdidas = 0  
 *self*.empates = 0  
  
 *def* detalles(*self*):  
 print("-----------------------------")  
 print("Monto restante: ", *self*.money)  
 print("-----------------------------")  
  
 *def* detallesfinal(*self*):  
 print("-----------------------------")  
 print("Nombre: ", *self*.nombre)  
 print("Monto restante: ", *self*.money)  
 print("Partidas ganadas: ", *self*.ganadas)  
 print("Partidas perdidas: ", *self*.perdidas)  
 print("Partidas empatadas: ", *self*.empates)  
 print("Porcentaje de victorias: ", (*self*.ganadas / (*self*.ganadas + *self*.perdidas + *self*.empates)) \* 100)  
 print("-----------------------------")  
  
 *def* apostar(*self*):  
 apuesta = validar\_monto\_apuesta()  
 *while* apuesta > *self*.money:  
 print("No puede apostar más de lo que tiene")  
 f = input("¿Desea depositar fichas? (S/N): ")  
 *if* f == "S":  
 *self*.depositar\_fichas()  
 *self*.apostar()  
 *elif* f == "N":  
 *self*.apostar()  
 *else*:  
 print("Opcion invalida")  
 *self*.apostar()  
 print("Apuesta realizada")  
 *return* apuesta  
  
 *def* depositar\_fichas(*self*):  
 deposito = validar\_monto\_fichas()  
 *self*.money += deposito  
 print("Deposito de fichas realizado")

1. Jugar.py

Aquí están definidas todas las funciones auxiliares del juego.

La primera función se encarga de recibir la respuesta para otra ronda e invoca al método que corresponda.

*from* Clases.Game *import* start\_game  
  
  
*def* ask\_round(baraja, jugador, croupier):  
 *if* len(baraja.decks) > 15:  
 respuesta = input("¿Quiere seguir jugando? (S/N): ")  
 *if* respuesta == "S":  
 start\_game(baraja, jugador, croupier)  
 *elif* respuesta == "N":  
 jugador.detallesfinal()  
 print("Regresando al menu principal.............\n--------------------------------\n")  
 *else*:  
 print("Respuesta no valida, intente de nuevo")  
  
 *else*:  
 print("No hay mas cartas, regresando al menu principal")  
 print("--------------------------------\n")

La siguiente valida el número de cartas para dar paso al juego.

*def* nuevo\_juego(baraja, jugador, croupier):  
 *if* len(baraja.decks) > 15:  
 start\_game(baraja, jugador, croupier)  
 ask\_round(baraja, jugador, croupier)  
 *else*:  
 *return*

La última sirve para saber que tipo de bienvenida darle al usuario y a partir de la entrada, imprime un cartel e invoca la acción correspondiente.

*def* jugar(baraja, jugador, croupier):  
 juego = *True*  
partidas = 0  
 *while* juego *is True*:  
 *if* partidas == 0:  
 *if* len(baraja.decks) > 15:  
 respuesta = input("¿Quiere comenzar a jugar? (S/N): ")  
 *if* respuesta == "S":  
 partidas = 1  
 print("Juego iniciado\n")  
 nuevo\_juego(baraja, jugador, croupier)  
 *if* len(baraja.decks) <= 15:  
 print("Juego terminado\n")  
 juego = *False*  
 *elif* respuesta == "N":  
 print("Regresando al menu principal....\n--------------------------------\n")  
 juego = *False*  
  
 *else*:  
 *if* len(baraja.decks) <= 15:  
 print("Juego terminado\n")  
 juego = *False*  
 *else*:  
 respuesta = input("¿Quiere seguir jugando? (S/N): ")  
 *if* respuesta == "S":  
 *if* len(baraja.decks) <= 15:  
 print("Juego terminado\n")  
 juego = *False*  
 *else*:  
 nuevo\_juego(baraja, jugador, croupier)  
 *elif* respuesta == "N":  
 jugador.detallesfinal()  
 print("Regresando al menu principal....\n--------------------------------\n")  
 juego = *False*

**Descripción del interfaz gráfica:**

A continuación la descripción de la versión con interfaz gráfica.

**GUI VERSION:**

Graphic User Interface adapted for Galactic BlackJack, based on high school´s blackjack and Pygame for UP. Given UI: 5020BJ\_wtran29

1. Main.py

Se importa el contenido del archivo Funcs, se muestra el cartel de juego empezado y se invoca la función principal.

*from* Funcs *import* \*  
  
*if* \_\_name\_\_ == "\_\_main\_\_":  
 print("Blackjack iniciado")  
 juego()

1. Funcs.py

En este archivo están definidas todas las funciones que construyen la interfaz gráfica.

Inclusión de librería y definición del método de música.

*import* random  
*import* tkinter  
*import* pygame  
  
pygame.mixer.init()  
  
  
*def* play\_music():  
 pygame.mixer.music.load("imperial.mp3")  
 pygame.mixer.music.play(loops=0)

Inclusión de una serie de imágenes que conforman la baraja.

*def* carga\_cartas(fotos):  
 deck\_s = ["♥", "♣", "♦", "♠"]  
 extras = ["J", "Q", "K"]  
 ext = "png"  
  
 *for* symbol *in* deck\_s:  
 *for* carta *in* range(1, 11):  
 img = tkinter.PhotoImage(file="Fotos/{}-{}.{}".format(str(carta), symbol, ext))  
 fotos.append((carta, img))  
 *for* carta *in* extras:  
 img = tkinter.PhotoImage(file="Fotos/{}-{}.{}".format(str(carta), symbol, ext))  
 fotos.append((10, img))

Método para quitar la carta de hasta arriba del deck y colocarla al final.

*def* quita\_pon(frm):  
 carta = deck.pop(0)  
 deck.append(carta)  
 tkinter.Label(frm, image=carta[1]).pack(side='left')  
 *return* carta

Método para puntuar la mano.

*def* puntua(mano):  
 total = 0  
 a = *False*  
 *for* carta *in* mano:  
 carta\_act = carta[0]  
 *if* carta\_act == 1 *and not* a:  
 a = *True*  
carta\_act = 11  
 total += carta\_act  
 *if* total > 21 *and* a:  
 total -= 10  
 a = *False*  
 *return* total

Checks para la casa y el jugador.

*def* casa\_check():  
 casa\_tot = puntua(mano\_casa)  
 *while* 0 < casa\_tot < 17:  
 mano\_casa.append(quita\_pon(carta\_casa\_frm))  
 casa\_tot = puntua(mano\_casa)  
 casa\_tot\_label.set(casa\_tot)  
  
 jug\_total = puntua(jug\_mano)  
 *if* jug\_total > 21:  
 res.set("¡Perdiste!")  
 *elif* casa\_tot > 21 *or* casa\_tot < jug\_total:  
 res.set("¡Ganaste!")  
 *elif* casa\_tot > jug\_total:  
 res.set("¡La casa gana!")  
 *else*:  
 res.set("¡Empataste!")  
  
  
*def* jug\_check():  
 jug\_mano.append(quita\_pon(jug\_carta\_frm))  
 jug\_total = puntua(jug\_mano)  
 jug\_total\_label.set(jug\_total)  
 *if* jug\_total > 21:  
 res.set("¡Perdiste!")

Funciones de inicio del juego

*def* inicio():  
 jug\_check()  
 mano\_casa.append(quita\_pon(carta\_casa\_frm))  
 casa\_tot\_label.set(puntua(mano\_casa))  
 jug\_check()  
  
  
*def* comenzar():  
 *global* carta\_casa\_frm  
 *global* jug\_carta\_frm  
 *global* mano\_casa  
 *global* jug\_mano  
  
 carta\_casa\_frm.destroy()  
 carta\_casa\_frm = tkinter.Frame(carta\_frm, bg="#01E9FD")  
 carta\_casa\_frm.grid(row=0, column=1, sticky='ew', rowspan=2)  
  
 jug\_carta\_frm.destroy()  
 jug\_carta\_frm = tkinter.Frame(carta\_frm, bg="#01E9FD")  
 jug\_carta\_frm.grid(row=2, column=1, sticky='ew', rowspan=2)  
  
 res.set("")  
  
 mano\_casa = []  
 jug\_mano = []  
 inicio()

*def* juego():  
 play\_music()  
 inicio()  
 root.mainloop()

Función para barajear:

*def* barajear():  
 random.shuffle(deck)

Función para finalizar el juego:

*def* exit():  
 pygame.mixer.music.stop()  
 root.destroy()  
 print("Blackjack finalizado")

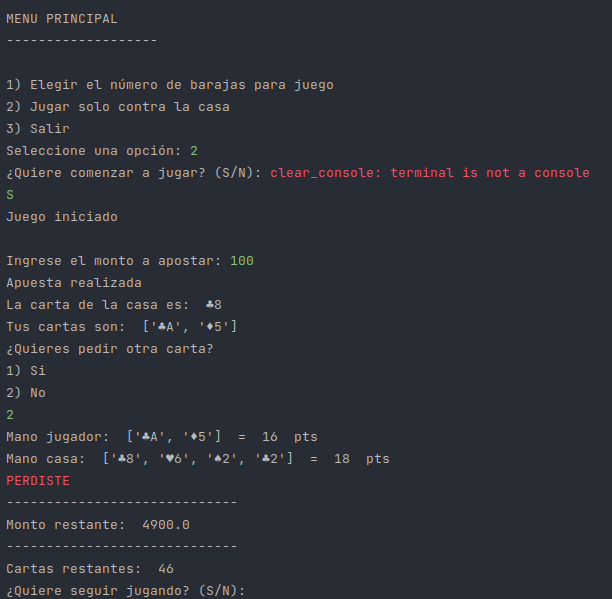
Configuración de la ventana, y los letreros y botones.

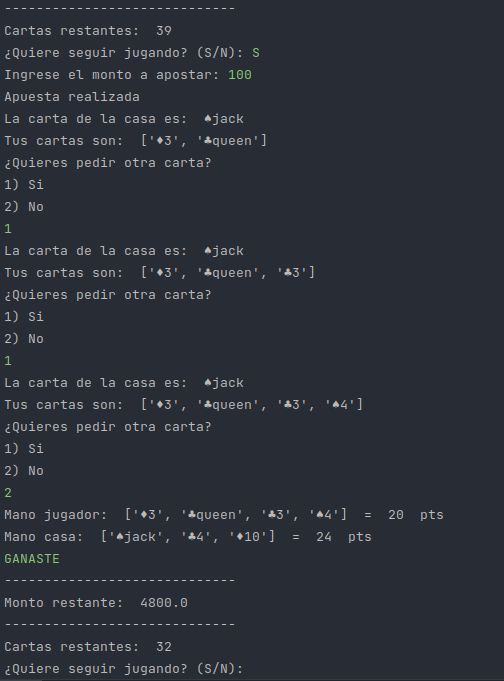
root = tkinter.Tk()  
root.title("KA - Blackjack def otra galaxia")  
root.geometry("{0}x{1}+0+0".format(root.winfo\_screenwidth(), root.winfo\_screenheight()))  
root.configure(bg="#01E9FD")  
  
root.columnconfigure(0, weight=2)  
root.columnconfigure(1, weight=2)  
root.columnconfigure(2, weight=2)  
root.columnconfigure(3, weight=0)  
root.columnconfigure(4, weight=5)  
root.columnconfigure(5, weight=0)  
  
res = tkinter.StringVar()  
result = tkinter.Label(root, textvariable=res)  
result.configure(bg="#01E9FD", font=("Arial", 50))  
result.grid(row=0, column=0, columnspan=3)  
  
carta\_frm = tkinter.Frame(root, borderwidth=1, bg="black")  
carta\_frm.grid(row=1, column=0, sticky='ew', columnspan=3, rowspan=2)  
  
casa\_tot\_label = tkinter.IntVar()  
tkinter.Label(carta\_frm, text="Croupier", bg="black", fg="white", font=("Arial", 40)).grid(row=0, column=0)  
tkinter.Label(carta\_frm, textvariable=casa\_tot\_label, bg="black", fg="white", font=("Arial", 40)).grid(row=1, column=0)  
carta\_casa\_frm = tkinter.Frame(carta\_frm, bg="black")  
carta\_casa\_frm.grid(row=0, column=1, sticky='ew', rowspan=2)  
  
jug\_total\_label = tkinter.IntVar()  
tkinter.Label(carta\_frm, text="Jugador", bg="black", fg="white", font=("Arial", 40)).grid(row=2, column=0)  
tkinter.Label(carta\_frm, textvariable=jug\_total\_label, bg="black", fg="white", font=("Arial", 40)).grid(row=3, column=0)  
jug\_carta\_frm = tkinter.Frame(carta\_frm, bg="black")  
jug\_carta\_frm.grid(row=2, column=1, sticky='ew', rowspan=2)  
  
button\_frame = tkinter.Frame(root)  
button\_frame.grid(row=3, column=1, columnspan=3, sticky='w')  
  
pedir\_button = tkinter.Button(button\_frame, text="Pedir otra carta", command=jug\_check, padx=8, width=30, height=5,  
 bg="blue", fg="white", font=("Arial", 16))  
pedir\_button.grid(row=0, column=0)  
  
quedar\_button = tkinter.Button(button\_frame, text="Quedarse", command=casa\_check, padx=5, width=30, height=5,  
 bg="green", fg="white", font=("Arial", 16))  
quedar\_button.grid(row=0, column=1)  
  
reinicia\_button = tkinter.Button(button\_frame, text="Comenzar de nuevo", command=comenzar, width=30, height=5,  
 bg="yellow",  
 fg="black", font=("Arial", 16))  
reinicia\_button.grid(row=0, column=2)  
  
barajear\_button = tkinter.Button(button\_frame, text="Barajear", command=barajear, padx=2, width=30, height=5,  
 bg="purple", fg="white", font=("Arial", 16))  
barajear\_button.grid(row=0, column=3)  
  
salir\_button = tkinter.Button(button\_frame, text="Salir", command=exit, padx=2, width=30, height=5, bg="red",  
 fg="white", font=("Arial", 16))  
salir\_button.grid(row=0, column=4)  
  
cartas = []  
carga\_cartas(cartas)  
  
deck = list(cartas) + list(cartas)  
barajear()  
  
mano\_casa = []  
jug\_mano = []

**Capturas de pantalla:**

**CLI VERSION:**







**GUI VERSION:**

