**Funciones Generales**

Función centrado(cadena)

cadena\_cent <- centrar cadena en 260 caracteres

imprimir cadena\_cent

Fin de la función

Función imprime\_resultado()

Si ganaste entonces

Mientras apuesta no sea 0 hacer

limpiar la pantalla

disminuir apuesta en 1

aumentar enpartida en 1

centrado("Deposito: " + dinero + " En partida: " + enpartida + " Apuesta: " + apuesta)

esperar 0.05 segundos

Fin del bucle

Si perdiste entonces

apuesta <- apuesta \* -1

Mientras apuesta no sea 0 hacer

limpiar la pantalla

aumentar apuesta en 1

disminuir enpartida en 1

centrado("Deposito: " + dinero + " En partida: " + enpartida + " Apuesta: " + apuesta)

esperar 0.05 segundos

Fin del bucle

Fin de la condición

ganaste <- Falso

perdiste <- Falso

Fin de la función

Función comprueba\_moroso(tipo\_apuesta, interfaz, mensaje)

Si tipo\_apuesta no es un entero entonces

tipo\_apuesta = esdigito(tipo\_apuesta, interfaz, mensaje)

Mientras (apuesta + tipo\_apuesta) sea mayor que enpartida hacer

Limpiar la pantalla

Mostrar ruleta

Mostrar "Lo siento pero tienes que introducir un número de monedas menor que el que tienes en partida."

Mostrar mensaje

Solicitar al usuario y guardar en tipo\_apuesta

tipo\_apuesta = esdigito(tipo\_apuesta, interfaz, mensaje)

Devolver tipo\_apuesta

Función esdigito(comprobante, interfaz, mensaje)

Mientras comprobante no exista o comprobante no sea un dígito hacer

Limpiar la pantalla

Mostrar interfaz

Solicitar al usuario para continuar con "\nLo siento, esta opción solamente contempla números.\nVuelve a intentarlo, para ello pulsa Enter."

Mostrar mensaje

Solicitar al usuario y guardar en comprobante

Devolver comprobante convertido a entero

**Funciones Blackjack**

Función manos\_inic(mano, carta1, carta2)

imprimir línea en blanco

Para cada línea en carta\_rever hacer

centrado(mano[carta1][linea] + " " + mano[carta2][linea])

Fin del bucle

Fin de la función

Función manos\_hit(mano)

Para cada línea en carta\_rever hacer

imprimir línea en blanco

Para cada carta en mano hacer

imprimir(mano[carta][linea] + " ", sin salto de línea)

Fin del bucle

Fin del bucle

imprimir línea en blanco

Fin de la función

Función carta\_rmdon()

carta\_aleatoria <- elección aleatoria de baraja\_use

eliminar carta\_aleatoria de baraja\_use

devolver carta\_aleatoria

Fin de la función

Función suma\_valores(valores\_extraidos, sumador)

sumador <- 0

prim\_vez\_as <- Falso

Para cada carta en valores\_extraidos hacer

indice <- índice de carta[0] en valores

sumador <- sumador + valores\_num[indice]

Si carta[0] es "A" y sumador < 11 entonces

prim\_vez\_as <- Verdadero

sumador <- sumador + 10

Fin de la condición

Fin del bucle

Para cada carta en valores\_extraidos hacer

Si carta[0] es "A" y sumador > 21 y prim\_vez\_as es Verdadero entonces

prim\_vez\_as <- Falso

sumador <- sumador - 10

Fin de la condición

Fin del bucle

devolver sumador

Fin de la función

Función imprime\_manos\_hit(valores\_extraidos, sumador, mano)

carta\_aleatoria <- carta\_rmdon()

new\_carta <- crea\_cartas(carta\_aleatoria)

agregar carta\_aleatoria a valores\_extraidos

agregar new\_carta a mano

sumador <- suma\_valores(valores\_extraidos, sumador)

limpiar la pantalla

manos\_inic(mano\_c, 1, 0)

imprimir mesa

Si mano es mano\_j\_1 entonces

imprimir "1º Mano"

Si mano es mano\_j\_2 entonces

imprimir "2º Mano"

Fin de la condición

manos\_hit(mano)

devolver sumador

Fin de la función

Función crea\_cartas(carta\_aleatoria)

Si carta\_aleatoria[0] es "10" entonces

crear carta con formato específico para "10"

Sino

crear carta con formato estándar

Fin de la condición

Si carta\_aleatoria[1] es "♥" o "♦" entonces

Para cada línea en carta hacer

añadir color rojo a la línea

Fin del bucle

Sino

Para cada línea en carta hacer

añadir color negro a la línea

Fin del bucle

Fin de la condición

devolver carta

Fin de la función

Función croupier\_hit(sumador, mano)

sumador\_c <- 0

sumador\_c <- suma\_valores(valores\_extraidos\_c, sumador\_c)

Mientras sumador\_c < 17 y sumador\_c <= sumador hacer

sumador\_c <- suma\_valores(valores\_extraidos\_c, sumador\_c)

Si sumador\_c < 17 y sumador\_c <= sumador entonces

imprimir línea en blanco

carta\_aleatoria <- carta\_rmdon()

new\_carta <- crea\_cartas(carta\_aleatoria)

añadir carta\_aleatoria a valores\_extraidos\_c

añadir new\_carta a mano\_c

limpiar la pantalla

manos\_hit(mano\_c)

imprimir mesa

manos\_hit(mano)

esperar 1.5 segundos

Fin de la condición

Fin del bucle

devolver sumador\_c

Fin de la función

Función ganador\_bj(sumador)

Si sumador > sumador\_c y sumador <= 21 o sumador\_c > 21 y sumador <= 21 entonces

centrado("YOU WIN")

esperar entrada del usuario

ganaste <- Verdadero

Si sumador es igual a sumador\_c entonces

centrado("Empate")

esperar entrada del usuario

Sino

centrado("YOU LOSE")

esperar entrada del usuario

perdiste <- Verdadero

Fin de la condición

Fin de la función

Función black\_jack(valores\_extraidos, sumador)

Si valores\_extraidos[0][0] es "A" y valores\_extraidos[1][0] está en valores[9:] o valores\_extraidos[1][0] es "A" y valores\_extraidos[0][0] está en valores[9:] entonces

blackjack <- Verdadero

sumador <- 21

Fin de la condición

devolver sumador

Fin de la función

**Funciones Ruleta**

Función gira\_rulet()

Si nr\_num[contador] es menor que 10 entonces

ruleta\_anim <- reemplazar nr[contador] con bola en ruleta

Sino

ruleta\_anim <- reemplazar nr[contador] con bola y un espacio en ruleta

Fin de la condición

imprimir ruleta\_anim

Fin de la función

**Funciones Slots**

Clase Slots

Método \_\_init\_\_()

Definir slot\_one, slot\_two y slot\_three con los valores correspondientes

Fin del método

Método roll()

Inicializar a, b, c con 1

Mientras a no sea divisible por 6, generar un número aleatorio entre 70 y 110 para a

Mientras b no sea divisible por 6, generar un número aleatorio entre 60 y 90 para b

Mientras c no sea divisible por 6, generar un número aleatorio entre 70 y 100 para c

Inicializar i con 1

Para cada número en el rango de a hacer

Limpiar la pantalla

Si i es igual a 6 entonces

Reiniciar i a 0

Mover el primer elemento de slot\_one, slot\_two y slot\_three al final

Fin de la condición

Componer las listas s1, s2, s3 para imprimir

Incrementar i en 1

Llamar a la función layout con s1, s2, s3

Esperar 0.02 segundos

Fin del bucle

Reiniciar i a 1

Para cada número en el rango de b hacer

Limpiar la pantalla

Si i es igual a 6 entonces

Reiniciar i a 0

Mover el primer elemento de slot\_two y slot\_three al final

Fin de la condición

Componer las listas s1, s2, s3 para imprimir

Incrementar i en 1

Llamar a la función layout con s1, s2, s3

Esperar 0.02 segundos

Fin del bucle

Reiniciar i a 1

Para cada número en el rango de c hacer

Limpiar la pantalla

Si i es igual a 6 entonces

Reiniciar i a 0

Mover el primer elemento de slot\_three al final

Fin de la condición

Componer las listas s1, s2, s3 para imprimir

Incrementar i en 1

Llamar a la función layout con s1, s2, s3

Esperar 0.02 segundos más t^2 / 90000 segundos

Fin del bucle

Fin del método

Fin de la clase

Función layout(s1, s2, s3)

Imprimir la interfaz gráfica del juego

Para cada elemento l, m, r en s1, s2, s3 hacer

Imprimir l, m, r en la interfaz gráfica

Fin del bucle

Imprimir apuesta y enpartida en la interfaz gráfica

Fin de la función

Función welcome()

Definir la lista de bienvenida

Para cada elemento c en la lista de bienvenida hacer

Imprimir c

Esperar 0.1 segundos

Fin del bucle

Esperar 1 segundo

Fin de la función

Función premios\_slots()

Si el tercer elemento de slot\_one es igual al segundo elemento de slot\_two y el primer elemento de slot\_three es igual al segundo elemento de slot\_two entonces

Calcular la nueva apuesta

Ganaste <- Verdadero

Fin de la condición

Si el primer elemento de slot\_one es igual al segundo elemento de slot\_two y el tercer elemento de slot\_three es igual al segundo elemento de slot\_two entonces

Calcular la nueva apuesta

Ganaste <- Verdadero

Fin de la condición

Si el segundo elemento de slot\_one es igual al segundo elemento de slot\_two y el segundo elemento de slot\_three es igual al segundo elemento de slot\_two entonces

Calcular la nueva apuesta

Ganaste <- Verdadero

Fin de la condición

Si el primer elemento de slot\_one es igual al primer elemento de slot\_two y el primer elemento de slot\_three es igual al primer elemento de slot\_two entonces

Calcular la nueva apuesta

Ganaste <- Verdadero

Fin de la condición

Si el tercer elemento de slot\_one es igual al tercer elemento de slot\_two y el tercer elemento de slot\_three es igual al tercer elemento de slot\_two entonces

Calcular la nueva apuesta

Ganaste <- Verdadero

Fin de la condición

Si ganaste entonces

Imprimir "Has ganado ", apuesta, " monedas."

Esperar la entrada del usuario

Fin de la condición

Fin de la función

**Algoritmo Principal**

Mientras opc\_casino no sea igual a "4" hacer

Limpiar la pantalla

Mostrar casino\_intr

Solicitar al usuario "¿A qué juego quieres jugar?: " y guardar en opc\_casino

Si opc\_casino es igual a "4" entonces

Salir del programa

Limpiar la pantalla

Mostrar casino\_intr

Si opc\_casino no es igual a "3" entonces

Mostrar "¿Con cuanto dinero quieres entrar a jugar?"

Mostrar "Ahora mismo tienes {deposito}"

Solicitar al usuario y guardar en enpartida

enpartida = esdigito(enpartida, casino\_intr, "¿Con cuanto dinero quieres entrar a jugar?\nAhora mismo tienes {deposito}")

Mientras enpartida sea mayor que deposito hacer

Limpiar la pantalla

Mostrar casino\_intr

Mostrar "¿Con cuanto dinero quieres entrar a jugar?"

Mostrar "Ahora mismo tienes {deposito}"

Solicitar al usuario y guardar en enpartida

enpartida = esdigito(enpartida, casino\_intr, "¿Con cuanto dinero quieres entrar a jugar?\nAhora mismo tienes {deposito}")

Si enpartida es mayor que deposito entonces

Mostrar "Lo siento, pero las monedas con las que ingreses a la partida tiene que ser menor que el que tienes en el depósito."

Solicitar al usuario para continuar

deposito = deposito - enpartida

Si opc\_casino es igual a "1" entonces

Definir salir como Falso

Definir finish como Falso

Definir seguir como "s"

Mientras salir sea Falso hacer

Limpiar la pantalla

Si finish es Verdadero entonces

Mostrar mesa

Solicitar al usuario "¿Quieres jugar otra partida? (s/n): " y guardar en seguir

Limpiar la pantalla

Si seguir es igual a "s" entonces

Mostrar mesa

Mostrar "\n"

Solicitar al usuario "¿Cuánto quieres apostar?: " y guardar en apuesta

apuesta = esdigito(apuesta, mesa, "¿Cuánto quieres apostar?: ")

Mientras apuesta sea mayor que enpartida hacer

Mostrar "\nLo siento pero la apuesta tiene que ser menor que las monedas que tengas en partida"

Solicitar al usuario para continuar

Limpiar la pantalla

Mostrar mesa

Mostrar "\n"

Solicitar al usuario "¿Cuánto quieres apostar?: " y guardar en apuesta

apuesta = esdigito(apuesta, mesa, "¿Cuánto quieres apostar?: ")

Limpiar la pantalla

Definir hit como "s"

Definir blackjack como Falso

Definir finish como Falso

Definir mano\_j como lista vacía

Definir mano\_c como lista vacía

Definir sumador\_c como 0

Definir valores\_extraidos\_c como lista vacía

Definir valores\_extraidos\_j como lista vacía

Definir baraja\_use como copia de baraja

Si no se cumple la condición anterior entonces

Definir salir como Verdadero

Llamar a la función vuelta\_al\_deposito()

Mientras finish sea Falso hacer

Añadir carta\_rever a mano\_c

Para i en el rango de 4 hacer

Definir carta\_aleatoria como resultado de carta\_rmdon()

Definir new\_carta como resultado de crea\_cartas(carta\_aleatoria)

Si i es par entonces

Añadir new\_carta a mano\_j

Añadir carta\_aleatoria a valores\_extraidos\_j

Sino

Añadir new\_carta a mano\_c

Añadir carta\_aleatoria a valores\_extraidos\_c

Llamar a manos\_inic(mano\_c, 1, 0)

Mostrar mesa

Llamar a manos\_inic(mano\_j, 0, 1)

Definir sumador\_j como resultado de suma\_valores(valores\_extraidos\_j, sumador\_j)

Definir pairs como resultado de comparar el primer elemento de valores\_extraidos\_j con el segundo

Definir sumador\_j como resultado de black\_jack(valores\_extraidos\_j, sumador\_j)

Si blackjack es Verdadero entonces

Definir win como Verdadero

Definir apuesta como apuesta \* 3/2

Solicitar al usuario "BLACKJACK"

Llamar a imprime\_resultado()

Definir finish como Verdadero

Sino si pairs es Verdadero entonces

Solicitar al usuario "¿Quieres dividir la mano? (s/n): " y guardar en dividir

Si dividir es "s" y pairs es Verdadero entonces

Definir mano\_j\_1 como lista con el primer elemento de mano\_j

Definir mano\_j\_2 como lista con el segundo elemento de mano\_j

Definir sumador\_j\_1 como 0

Definir sumador\_j\_2 como 0

Definir valores\_extraidos\_j\_1 como lista con el primer elemento de valores\_extraidos\_j

Definir valores\_extraidos\_j\_2 como lista con el segundo elemento de valores\_extraidos\_j

Definir sumador\_j\_1 como resultado de imprime\_manos\_hit(valores\_extraidos\_j\_1, sumador\_j\_1, mano\_j\_1)

Definir sumador\_j\_1 como resultado de black\_jack(valores\_extraidos\_j\_1, sumador\_j\_1)

Si blackjack es Verdadero entonces

Solicitar al usuario "BLACKJACK"

Definir blackjack como Falso

Mientras hit sea "s" y sumador\_j\_1 sea menor que 21 hacer

Solicitar al usuario "¿Quieres otra carta? (s/n): " y guardar en hit

Si hit es "s" entonces

Definir sumador\_j\_1 como resultado de imprime\_manos\_hit(valores\_extraidos\_j\_1, sumador\_j\_1, mano\_j\_1)

Solicitar al usuario "Pasamos a la segunda mano"

Definir hit como "s"

Definir sumador\_j\_2 como resultado de imprime\_manos\_hit(valores\_extraidos\_j\_2, sumador\_j\_2, mano\_j\_2)

Definir sumador\_j\_2 como resultado de black\_jack(valores\_extraidos\_j\_2, sumador\_j\_2)

Si blackjack es Verdadero entonces

Solicitar al usuario "BLACKJACK"

Mientras hit sea "s" y sumador\_j\_2 sea menor que 21 hacer

Solicitar al usuario "¿Quieres otra carta? (s/n): " y guardar en hit

Si hit es "s" entonces

Definir sumador\_j\_2 como resultado de imprime\_manos\_hit(valores\_extraidos\_j\_2, sumador\_j\_2, mano\_j\_2)

Definir blackjack como Falso

Si dividir no es igual a "s" y sumador\_j no es igual a 21 entonces

Mientras hit sea igual a "s" hacer

Solicitar al usuario "¿Quieres otra carta? (s/n): " y guardar en hit

Si hit es igual a "s" entonces

Definir sumador\_j como resultado de imprime\_manos\_hit(valores\_extraidos\_j, sumador\_j, mano\_j)

Si sumador\_j es mayor que 21 entonces

Definir hit como "n"

Definir finish como Verdadero

Definir lose como Verdadero

Llamar a la función centrado con argumento Fore.RED + "YOU LOSE" + Fore.RESET

Solicitar al usuario para continuar

Llamar a la función imprime\_resultado()

Si finish es Falso entonces

Definir croupier\_play como resultado de negar (sumador\_j\_1 >= 21 y sumador\_j\_2 >= 21)

Si croupier\_play es Verdadero entonces

Limpiar la pantalla

Llamar a manos\_inic(mano\_c, 1, 0)

Mostrar mesa

Si pairs es Verdadero y dividir es igual a "s" entonces

Llamar a manos\_hit(mano\_j\_1)

Llamar a manos\_hit(mano\_j\_2)

Sino

Llamar a manos\_hit(mano\_j)

Esperar 2 segundos

Eliminar el primer elemento de mano\_c

Limpiar la pantalla

Llamar a manos\_inic(mano\_c, 0, 1)

Mostrar mesa

Si pairs es Verdadero y dividir es igual a "s" entonces

Llamar a manos\_hit(mano\_j\_1)

Llamar a manos\_hit(mano\_j\_2)

Sino

Llamar a manos\_hit(mano\_j)

Esperar 2 segundos

Definir sumador\_c como resultado de black\_jack(valores\_extraidos\_c, sumador\_c)

Definir blackjack como Falso

Si pairs es Verdadero y dividir es igual a "s" entonces

Si sumador\_c no es igual a 21 entonces

Definir sumador\_c como resultado de croupier\_hit(sumador\_j\_1, mano\_j\_1)

Sino

Si sumador\_c no es igual a 21 entonces

Definir sumador\_c como resultado de croupier\_hit(sumador\_j, mano\_j)

Sino

Eliminar el tercer elemento de mano\_c

Si pairs es Verdadero y dividir es igual a "s" entonces

Limpiar la pantalla

Llamar a manos\_hit(mano\_c)

Mostrar mesa

Mostrar "1º Mano"

Llamar a manos\_hit(mano\_j\_1)

Esperar 1.5 segundos

Llamar a black\_jack(valores\_extraidos\_j\_1, sumador\_j\_1)

Definir memo\_apuesta como apuesta

Si blackjack es Verdadero entonces

Definir apuesta como apuesta \* 3/2

Definir win como Verdadero

Definir blackjack como Falso

Solicitar al usuario "BLACKJACK"

Sino

Llamar a ganador\_bj(sumador\_j\_1)

Llamar a imprime\_resultado()

Limpiar la pantalla

Llamar a manos\_hit(mano\_c)

Mostrar mesa

Mostrar "2º Mano"

Llamar a manos\_hit(mano\_j\_2)

Esperar 1.5 segundos

Llamar a black\_jack(valores\_extraidos\_j\_2, sumador\_j\_2)

Definir apuesta como memo\_apuesta

Si blackjack es Verdadero entonces

Definir win como Verdadero

Definir apuesta como apuesta \* 3/2

Solicitar al usuario "BLACKJACK"

Sino

Llamar a ganador\_bj(sumador\_j\_2)

Llamar a imprime\_resultado()

Sino

Llamar a ganador\_bj(sumador\_j)

Llamar a imprime\_resultado()

Definir finish como Verdadero

Si opc\_casino es igual a "2" entonces

Definir finish como "s"

Mientras finish sea igual a "s" hacer

Definir dict\_apuesta\_num como diccionario vacío

Definir dict\_apuesta\_docena como diccionario con "Primera docena" : Falso, "Segunda docena" : Falso, "Tercera docena" : Falso

Definir dict\_apuesta\_colum como diccionario con "Columna 1" : Falso, "Columna 2" : Falso, "Columna 3" : Falso

Definir dict\_apuesta\_color como diccionario con "Rojo" : Falso, "Negro" : Falso

Definir dict\_apuesta\_par como diccionario con "Pares" : Falso, "Impares" : Falso

Definir dict\_apuesta\_bajo como diccionario con "Alto" : Falso, "Bajo" : Falso

Definir menu\_rul como "0"

Definir apuesta como 0

Definir tiempo como 0.001

Definir rul\_opt como entero

Definir tiempo\_rulet como número aleatorio entre 300 y 400

Definir contador como 0

Mientras menu\_rul no sea igual a "7" hacer

Limpiar la pantalla

Mostrar ruleta

Definir menu\_ruleta como "¿A qué quieres jugar?\n

1. Número

2. Docenas

3. Columnas

4. Color

5. Par o impar

6. Bajo o alto

7. Comenzar partida"

Mostrar menu\_ruleta

Solicitar al usuario y guardar en menu\_rul

Según el valor de menu\_rul hacer

Caso "1":

Mientras rul\_opt no sea igual a 777 hacer

Limpiar la pantalla

Mostrar ruleta

Mostrar "Introduce un número: "

Mostrar "Para salir introduce '777'"

Solicitar al usuario y guardar en rul\_opt

rul\_opt = esdigito(rul\_opt, ruleta, "Introduce un número: \nPara salir introduce '777'")

Si rul\_opt es menor que 37 entonces

Solicitar al usuario "Introduce la apuesta para el número {rul\_opt}: " y guardar en dict\_apuesta\_num[rul\_opt]

dict\_apuesta\_num[rul\_opt] = comprueba\_moroso(dict\_apuesta\_num[rul\_opt], ruleta, "Introduce la apuesta para el número {rul\_opt}: ")

Incrementar apuesta por dict\_apuesta\_num[rul\_opt]

Sino

Si rul\_opt no es igual a 777 entonces

Mostrar "Esta opción no está contemplada."

Solicitar al usuario para continuar

Caso "2":

Limpiar la pantalla

Mostrar ruleta

Mostrar "¿A qué docena quieres jugar?:\n

1. Primera docena (1-12)

2. Segunda docena (13-24)

3. Tercera docena (25-36)\n"

Solicitar al usuario y guardar en rul\_opt

Solicitar al usuario "\nIntroduce la apuesta para la docena: " y guardar en apuesta\_docena

apuesta\_docena = comprueba\_moroso(apuesta\_docena, ruleta, "\nIntroduce la apuesta para la docena: ")

Según el valor de rul\_opt hacer

Caso "1":

Definir dict\_apuesta\_docena["Primera docena"] como apuesta\_docena

Caso "2":

Definir dict\_apuesta\_docena["Segunda docena"] como apuesta\_docena

Caso "3":

Definir dict\_apuesta\_docena["Tercera docena"] como apuesta\_docena

Incrementar apuesta por apuesta\_docena

Caso "3":

Limpiar la pantalla

Mostrar ruleta

Mostrar "¿A qué columna quieres jugar?:\n

1. Primer columna

2. Segunda columna

3. Tercera columna \n"

Solicitar al usuario y guardar en rul\_opt

Solicitar al usuario "\nIntroduce la apuesta para la columna: " y guardar en apuesta\_columna

apuesta\_columna = comprueba\_moroso(apuesta\_columna, ruleta, "\nIntroduce la apuesta para la columna: ")

Según el valor de rul\_opt hacer

Caso "1":

Definir dict\_apuesta\_colum["Columna 1"] como apuesta\_columna

Caso "2":

Definir dict\_apuesta\_colum["Columna 2"] como apuesta\_columna

Caso "3":

Definir dict\_apuesta\_colum["Columna 3"] como apuesta\_columna

Incrementar apuesta por apuesta\_columna

Caso "4":

Limpiar la pantalla

Mostrar ruleta

Mostrar "¿A qué color quieres jugar?:\n

1. Negro

2. Rojo \n"

Solicitar al usuario y guardar en rul\_opt

Solicitar al usuario "\nIntroduce la apuesta para el color: " y guardar en apuesta\_color

apuesta\_color = comprueba\_moroso(apuesta\_color, ruleta, "Introduce la apuesta para el color: ")

Según el valor de rul\_opt hacer

Caso "1":

Definir dict\_apuesta\_color["Negro"] como apuesta\_color

Caso "2":

Definir dict\_apuesta\_color["Rojo"] como apuesta\_color

Incrementar apuesta por apuesta\_color

Caso "5":

Limpiar la pantalla

Mostrar ruleta

Mostrar "¿A qué quieres jugar, pares o impares?:\n

1. Pares

2. Impares \n"

Solicitar al usuario y guardar en rul\_opt

Solicitar al usuario "\nIntroduce la apuesta para pares / impares: " y guardar en apuesta\_par

apuesta\_par = comprueba\_moroso(apuesta\_par, ruleta, "Introduce la apuesta para pares / impares: ")

Según el valor de rul\_opt hacer

Caso "1":

Definir dict\_apuesta\_par["Pares"] como apuesta\_par

Caso "2":

Definir dict\_apuesta\_par["Impares"] como apuesta\_par

Incrementar apuesta por apuesta\_par

Caso "6":

Limpiar la pantalla

Mostrar ruleta

Mostrar "¿A qué quieres jugar, alto o bajo?:\n

1. Alto

2. Bajo \n"

Solicitar al usuario y guardar en rul\_opt

Solicitar al usuario "\nIntroduce la apuesta para alto / bajo: " y guardar en apuesta\_bajo

apuesta\_bajo = comprueba\_moroso(apuesta\_bajo, ruleta, "Introduce la apuesta para alto / bajo: ")

Según el valor de rul\_opt hacer

Caso "1":

Definir dict\_apuesta\_bajo["Alto"] como apuesta\_bajo

Caso "2":

Definir dict\_apuesta\_bajo["Bajo"] como apuesta\_bajo

Incrementar apuesta por apuesta\_bajo

Definir apuesta como apuesta multiplicada por -1

Para i en el rango de tiempo\_rulet hacer

Incrementar contador por 1

Si contador es mayor que 36 entonces

Definir contador como 0

Esperar tiempo segundos

Si i es menor que tiempo\_rulet / 1.5 entonces

Continuar al siguiente ciclo

Sino si i es mayor que tiempo\_rulet / 1.5 y i es menor que (tiempo\_rulet \* 94) / 100 entonces

Definir tiempo como tiempo elevado a la potencia de (99/100)

Sino

Definir tiempo como tiempo elevado a la potencia de (19/20)

Limpiar la pantalla

Llamar a la función gira\_rulet()

Mostrar "Ha salido el número ", Back.WHITE + nr[contador] + Back.RESET

Definir num\_aleatorio como nr\_num[contador]

Para cada par numero, dinero en dict\_apuesta\_num hacer

Si num\_aleatorio es igual a numero entonces

Incrementar apuesta por (dinero \* 36)

Si num\_aleatorio es menor que 13 y dict\_apuesta\_docena["Primera docena"] no es Falso entonces

Incrementar apuesta por (apuesta\_docena \* 3)

Sino si num\_aleatorio es menor que 25 y dict\_apuesta\_docena["Segunda docena"] no es Falso entonces

Incrementar apuesta por (apuesta\_docena \* 3)

Sino si num\_aleatorio es menor que 37 y dict\_apuesta\_docena["Tercera docena"] no es Falso entonces

Incrementar apuesta por (apuesta\_docena \* 3)

Si num\_aleatorio está en columna1 y dict\_apuesta\_colum["Columna 1"] no es Falso entonces

Incrementar apuesta por (apuesta\_columna \* 3)

Si num\_aleatorio está en columna2 y dict\_apuesta\_colum["Columna 2"] no es Falso entonces

Incrementar apuesta por (apuesta\_columna \* 3)

Si num\_aleatorio está en columna3 y dict\_apuesta\_colum["Columna 3"] no es Falso entonces

Incrementar apuesta por (apuesta\_columna \* 3)

Si num\_aleatorio está en rojo y dict\_apuesta\_color["Rojo"] no es Falso entonces

Incrementar apuesta por (apuesta\_color \* 2)

Sino si dict\_apuesta\_color["Negro"] no es Falso entonces

Incrementar apuesta por (apuesta\_color \* 2)

Si num\_aleatorio es par y dict\_apuesta\_par["Pares"] no es Falso entonces

Incrementar apuesta por (apuesta\_par \* 2)

Sino si dict\_apuesta\_par["Impares"] no es Falso entonces

Incrementar apuesta por (apuesta\_par \* 2)

Si num\_aleatorio es menor que 19 y dict\_apuesta\_bajo["Bajo"] no es Falso entonces

Incrementar apuesta por (apuesta\_bajo \* 2)

Sino si dict\_apuesta\_bajo["Alto"] es Verdadero entonces

Incrementar apuesta por (apuesta\_bajo \* 2)

Si apuesta es mayor que 0 entonces

Mostrar "Tus beneficios son ", Fore.GREEN + str(apuesta) + Fore.RESET

Solicitar al usuario para continuar

Definir win como Verdadero

Sino

Mostrar "Tus beneficios son ", Fore.RED + str(apuesta) + Fore.RESET

Solicitar al usuario para continuar

Definir lose como Verdadero

Si apuesta es menor que 0 entonces

Definir apuesta como apuesta multiplicada por -1

Llamar a la función imprime\_resultado()

Solicitar al usuario "¿Quieres jugar otra partida? (S/N): " y guardar en finish

Si finish no es igual a "s" entonces

Llamar a la función vuelta\_al\_deposito()

Si opc\_casino es igual a "3" entonces

Crear una nueva instancia de Slots y guardar en machine

Llamar a la función welcome()

Limpiar la pantalla

Mostrar "\n\n\n"

Llamar a la función centrado con argumento "Introduce cuánto dinero quieres tener en partida:"

Solicitar al usuario y guardar en enpartida

enpartida = esdigito(enpartida, "", "Introduce cuánto dinero quieres tener en partida:")

Limpiar la pantalla

Mostrar "\n\n\n"

Llamar a la función centrado con argumento "Introduce la apuesta con la quieres jugar: "

Mostrar "\n"

Llamar a la función centrado con argumento "Recuerda que en este modelo de juego la apuesta es invariable"

Llamar a la función centrado con argumento "En el caso de que quieras modificar la apuesta tendrás que salir "

Llamar a la función centrado con argumento "al menú principal e introducirla de nuevo."

Mostrar "\n"

Solicitar al usuario y guardar en apuesta

apuesta = esdigito(apuesta, "Introduce la apuesta con la quieres jugar: \nRecuerda que en este modelo de juego la apuesta es invariable\nEn el caso de que quieras modificar la apuesta tendrás que salir", "")

Restar enpartida de deposito y guardar en deposito

Guardar apuesta en memo\_apuesta

Definir finish como "s"

Mientras finish sea igual a "s" hacer

Definir apuesta como memo\_apuesta

Restar apuesta de enpartida y guardar en enpartida

Llamar al método roll de machine

Llamar a la función premios\_slots()

Llamar a la función imprime\_resultado()

Mostrar "\n"

Si enpartida es menor o igual a 0 entonces

Mostrar "Lo siento te has quedado sin monedas."

Solicitar al usuario para continuar

Sino

Solicitar al usuario "¿Quieres jugar otra vez? (S/N): " y guardar en finish

Si finish no es igual a "s" entonces

Llamar a la función vuelta\_al\_deposito()