matias galarza

comision 13  Introducción a la programación

Documentacion tp-final

Trabajo practico final

### ‘Introduccion

Para el trabajo practico final se nos dio el desafío de terminar un juego dado por el profesor. Este proyecto consiste en un juego similar al ahorcado donde el usuario deberá ir adivinando palabras con las pistas que se le van dando.

# MENU DE INICIO Y selección DE DIFICULTAD

main\_MENU ():

Muestra en pantalla los botones del menú principal (jugar, quitar, etc.)

play ():

Esta función nos muestra las 3 dificultades que hay en el juego: fácil (4 letras), medio (6 letras) y difícil (8 letras).

Con esta función el usuario puede elegir la longitud de palabras con las que quiere jugar.

## ANOTACIONES

Se utilizo un menú de internet y se lo adapto al proyecto. Para realizar la adaptación se hizo una traducción del inglés al español y se quitaron los botones que no eran de utilidad.

No hubo muchas complicaciones en agregarlo al proyecto, aunque pasarle la dificultad al programa principal si nos tomó unas horas más.

# FUncion principal del proyecto

PRINCIPAL (Largo\_elegido):

if len(palabraUsuario) == len(palabraCorrecta):  
 if enlista(palabraUsuario,listaPalabrasDiccionario):  
 gano = revision(palabraCorrecta, palabraUsuario, correctas, incorrectas, casi)  
 ListaDePalabrasUsuario.append(palabraUsuario)  
 puntos += puntaje(correctas, casi, palabraUsuario)  
 palabraUsuario = ""  
 intentos -= 1  
 else:  
 ventana\_notas("lemario")  
else:  
 ventana\_notas("largo")

Es la función principal del programa, se encarga de leer el archivo con las palabras, sacar una palabra de la lista, controlar el tiempo que tiene el usuario para adivinar la palabra, establecer la dificultad, y llamar a la función que nos permite mostrar todo por pantalla.

El código de la línea 70 hasta la 80 sirve para validar si el usuario está ingresando una palabra dentro del lemario y del largo elegido anteriormente, en caso contrario el usuario no podrá ingresar la palabra.

## Anotaciones

Con esta parte del código no tuvimos mayores problemas ya que no había mucho que hacer sobre ella, utilizamos las funciones que estaban en FUNCIONES VACIAS sin mucha dificultad, abrimos el archivo que contiene todas las palabras en modo lectura y además añadimos la variable” largo\_elegido” que es de donde se obtendrá la dificultad elegida en el menú principal.

# archivo funcionesvacias

lectura (archivo, salida, largo)

def lectura(archivo, salida, largo):  
 lineas = archivo.readlines()  
 for linea in lineas:  
 cadena = quitar\_salto(linea)  
 if len(cadena) == largo:  
 salida.append(cadena)  
 print(salida)

Esta función recibe un archivo de texto por parámetro, al que va leer línea por línea y agregara a una lista de la función principal las palabras que coinciden con el largo elegido (que también es pasado por parámetro).

Nuevapalabra(lista)

def nuevaPalabra(lista):  
 palabra = random.choice(lista)  
 return palabra

Con esta parte del código se elige una palabra de manera aleatoria de la lista que se pasa por parámetros.

La utilizamos para elegir la palabra que tiene que adivinar el usuario.

Revision (palabraCorrecta, palabra, correctas, incorrectas, casi)

def revision(palabraCorrecta, palabra, correctas, incorrectas, casi):  
 cont = 0  
 for i in range(len(palabraCorrecta)):  
 if palabraCorrecta[i] == palabra[i]:  
 correctas.append(palabra[i])  
 cont = cont + 1  
 elif palabra[i] in palabraCorrecta:  
 casi.append(palabra[i])  
 else:  
 incorrectas.append(palabra[i])  
 if cont == len(palabraCorrecta):  
 return True  
 return False

Con esta función se hace una revision de la palabra ingresada por el usuario.

Recorre la cadena carácter por carácter comparándola con la palabra correcta contando la cantidad de aciertos del usuario. Mientras se esta en el ciclo los caracteres van añadiéndose a correctas, incorrectas y casi (el carácter se agrega a la lista cuando la letra esta en la palabra, pero no en la misma posición)

quitar\_salto(palabra)

def quitar\_salto(palabra):  
 cadena = ""  
 for letra in palabra:  
 if letra != "\n":  
 cadena = cadena + letra  
 return cadena

Esta función le quita el salto de línea a una palabra.

El archivo de texto utilizado para guardar las palabras las almacenaba con un salto de línea y para evitar con problemas de longitud en las palabras.

## anotaciones

Al principio hubo inconvenientes por el salto de línea que tenían las palabras sacadas del archivo, sin embargo, después de eso logramos conseguir un buen funcionamiento de este módulo del juego.

# ARCHIVO EXTRAS

dameLetraApretada(key)

Esta función simplemente nos devuelve la tecla que fue presionada por el usuario. Después se agregaría la letra ñ a la función utilizando el código ASCII.

Dibujar (screen, listaDePalabrasUsuario, palabraUsuario, puntos, segundos, gano, correctas, incorrectas, casi, largo, intentos)

Esta función se utiliza para dibujar todos los elementos necesarios para jugar en la pantalla. Dentro de lo elementos que se pueden ver en pantalla son:

* La cantidad de intentos
* El tiempo que le queda al usuario
* Las teclas que fueron presionas con distintos colores (para que el usuario sepa en letras de la palabra acertó)
* Las palabras que fueron ingresadas por el usuario
* Mensajes que le indican al usuario si gano o perdió

Mensaje (gano, intentos, largo)

def mensaje(gano, intentos ,puntos):  
 if gano:  
 ventana\_secundaria(puntos)  
 elif intentos == 0:  
 return "Se acabaron los intentos :("  
 else:  
 return "Fallaste :("

Esta función nos muestra en pantalla un mensaje que le indica al usuario el estado de la partida.

Texto (screen, Font, texto, color, x, y)

def texto(screen, font, texto, color, x, y):  
 tipo\_font = pygame.font.Font(font, 20)  
 text = tipo\_font.render(texto, True, color)  
 rectangulo = text.get\_rect()  
 rectangulo.center = (x, y)  
 screen.blit(text, rectangulo)

Esta función nos deja mostrar texto por pantalla con los elementos que le pasamos por parámetro. Se creo para simplificar el proceso de dibujar texto en la pantalla y para tener un código más limpio.

## Anotaciones

Con esta parte del proyecto tuvimos el problema con el parpadeo molesto que resolvimos después de un tiempo limpiando la cache del proyecto y reescribiendo el archivo. También decidimos crear una función que haga todo el proceso de mostrar texto por pantalla, ya que era un procedimiento que requería varias líneas de código y así logramos tener un código más limpio.

## ANOTACIONES: dificultades en el desarrollo

Al cerrar el programa con el pygame.QUIT salían varios errores molestos después de que dejaba de ejecutarse. Después de mucha investigación se solucionó añadiendo la función sys.exit() después de utilizar QUIT.

También faltaba no permitirle al usuario ingresar palabras que no estén el lemario y que no sean del mismo largo seleccionado. Se utilizo una ventana que sirve para notificarle al usuario una de estas dos situaciones.

Conclusion

Lo que más retraso el avance fue el salto de línea que tenían todas las palabras del archivo, pero después de arreglarlo el proceso fue mucho más rápido y con menos inconvenientes.

Se logro un juego funcional que se caracteriza por ser un desafío para el usuario.