Universidad Nacional de General Sarmiento
Introducción a la Programación
Comision 07
Estomba, Maria Florencia

Informe

El trabajo consiste en implementar funciones para el juego TutiFruti. La idea es que el jugador/ra mejore sus tiempos y aprenda nuevas palabras. En pantalla aparece una letra y una categoría, entonces el jugador debe escribir una palabra que esté dentro de esa categoría que empiece con dicha letra.

Funciones implementadas

El juego viene con algunas funciones que se encuentran vacías, las cuales se deben implementar en el archivo funciones.py, las cuales son utilizadas desde el programa principal.

⏩En la función **unaAlAzar(lista):** primero importamos de la librería el comando random: *import random,* luego se utilizó random.choice(lista), el cual sirve para seleccionar al azar elementos de una lista, en el caso de nuestro juego, una letra. Finalmente, colocamos el comando *return,* el cual devuelve un elemento de la lista que recibe por parámetro.

⏩En la función **esCorrecta(palabraUsuario, letra, item, items, listaDeTodo):** utilizamos una nueva variable, *palabraLimpia*, en la cual llamamos a una función auxiliar *limpiarPalabra(palabraUsuario)*,que se encarga de pasar la palabra a minúscula y reemplazar todas las vocales con tildes y diéresis (si tuviera), para luego guardarla en nuestra variable.
Una vez analizadas las palabras, se verifica si la palabra dentro de la variable es distinta a vacío y si en la posición inicial, es decir [0], es igual a la letra devuelta por la función unaAlAzar aleatoriamente.
Inicializamos un *contador* en 0, el cual vamos a usar para entrar al índice de *ListaDeTodo* por posición, (ya que es igual a la lista de items) recorremos con un for las listas de *ítems*, donde el *itemActual* va a ir tomando el valor de cada *items*, *contador* va a ir, valga la redundancia, contando hasta que *itemActual* sea igual a *item*. Una vez que sucede esto, usamos la variable *lista* donde traemos de regreso la listaDeTodo que contiene a la lista del item, de la posición en la que quedó contador; para luego recorrerla con otro for, y si *palabraActual* es igual a la palabra que había introducido el usuario al principio del código, entonces se emitirá un efecto de sonido dando a entender que es correcto, y retornara 10 puntos, en caso contrario se emitirá otro efecto de sonido dando a entender que la palabra es incorrecta y no sumará puntos.

⏩En la función **juegaCompu(letraAzar, listaDeTodo):** inicializamos una lista vacía, definida como *palabrasValidas*. Luego se recorren las categorías de la l*istaDeTodo* con un for, y se coloca una variable denominada *palabraEncontrada*, que por el momento se encuentra vacía. Utilizamos un segundo for para recorrer las palabras dentro de las categorías y si la *letraAzar* es igual a la palabra en la posición [0], es decir el primer carácter

que tiene esa palabra, entonces se guarda en la variable *palabraEncontrada.* En caso de no encontrar una palabra para esa categoría, no se modifica el valor de palabraEncontrada que vale "" por defecto, ya que no se encontraron palabras que sean válidas. Finalmente, se retorna la variable *palabrasValidas*, que contiene *palabraEncontrada*.

Errores encontrados al implementar las funciones

1) En la función esCorrecta(palabraUsuario, letra, item, items, listaDeTodo): Si no se coloca ninguna letra en la consola del juego y se procede a dar la tecla enter, el juego se tilda y muestra un error encontrado en cierta parte de nuestro codigo:

```
def esCorrecta(palabraUsuario, letra, item, items listaDeTodo):
        palabraUsuario=palabraUsuario.lower()
    ⇨⇨⇨⇨ if(palabraUsuario[0]==letra):      ⇦⇦⇦⇦
            contador=0
            for itemActual in items:
                if itemActual==item:
                    lista=listaDeTodo[contador]
                    for palabraActual in lista:
                        if palabraActual==palabraUsuario:
            return 10
```

Al analizarlo, nos hemos dado cuenta que en la condición del if, estábamos asumiendo que la palabra ingresada por el usuario no era vacía y accediamos a la posición [0] , la cual no existe.
Para solucionarlo, hemos utilizado otra condición en la rama del if, por lo que se ejecutará siempre y cuando ambas sean correctas, ya que estamos utilizando un operador lógico como es **and**, el cual evalúa que ambas condiciones sean verdaderas, de esta manera devolverá verdadero. Por lo que si el usuario pone una letra o algo que sea distinto a vacío, el código no terminara en error.

```
def esCorrecta(palabraUsuario, letra, item, items, listaDeTodo):
        palabraUsuario=palabraUsuario.lower()
    ⇨⇨⇨⇨ if(palabraUsuario!="" and palabraUsuario[0]==letra):      ⇦⇦⇦⇦
            contador=0
            for itemActual in items:
                if itemActual==item:
                    lista=listaDeTodo[contador]
                    for palabraActual in lista:
                        if palabraActual==palabraUsuario:
            return 10
```

2) Otro de los problemas detectados surgió cuando al ingresar una palabra que correspondía a cierta categoría, la cual ya sabíamos que se encontraba en las listas, nos devolvía como si no fuera parte de las mismas y no sumaba punto. El problema que descubrimos utilizando debug era que ciertas palabras se encontraban guardadas en los archivos con un espacio luego de terminada la palabra, como por ejemplo en la categoría Frutas y verduras la cadena limon. Al hacerse la comparación de *palabraUsuario* y *palabraLimpia* en la función *esCorrecta()*, igualaba lo siguiente:
“limon”==”limon “
Para enmendarlo chequeamos todas los archivos de lectura que tuvieran el mismo problema y eliminamos el espacio final.

3) Los efectos de sonido en la función esCorrecta(), con el propósito que la palabra ingresada por el usuario, si se encontraba en las listas reproduciera un sonido y en caso contrario otro, no sucedía, Estuvimos investigamos cuál podría ser el posible error, y detectamos que el formato en el que habíamos descargado el sonido, “mp3”, no lo reproducia, por lo que pasamos los efectos a formato “wav”, que según vimos para los sonidos en pygame solo permiten usar archivos *ogg* o *wav*.

Funciones auxiliares
Se implementaron algunas funciones que resuelven tareas intermedias:

🔼Función de lectura de archivos:
Esta función se utiliza para poder leer desde el juego, los archivos guardados en nuestra carpeta. Definimos la función como *lectura* y le pasamos un parámetro, en este caso nombreArchivo, ya que nuestro juego utiliza varios archivos con distintos nombres.
Se procederá a abrir los archivos en el formato txt de nuestro archivo, y se indicará que tipo de acción realizaremos, en nuestro caso de lectura, es decir **r**. Luego, se guardará en una variable llamada *contenido*, usamos la funcion splitLine que toma la cadena *contenido*, lo corta por *“\n”* y termina retornando una lista de palabras, para eso usamos el comando *return*. Previo a retornar la lista los archivos pasan por la función *limpiarPalabra(contenido)*, la cual pasará nuestro archivo a minúscula y reemplaza todas las vocales con tildes y diéresis (si tuviera).

🔼Funcion para limpiar palabras:
Definimos la función como *limpiarPalabra* y le pasamos un parámetro llamado *palabra.*
Primero pasamos todas las palabras que reciba la función a minúscula e inicializamos una variable, llamada *reemplazos*, la cual contiene una lista de tuplas, donde en la primera mostrará la vocal que contenga tilde o diéresis y en la segunda la vocal sola.
Utilizaremos un for para poder recorrer las tuplas denominadas como *caracCon*(carácter con tilde o diéresis) y *caracSin*(carácter sin tilde o diéresis) en la variable *reemplazos*. Se procederá a reemplazar una por la otra utilizando la función

*palabraLimpia.replace(caracCon,caracSin),*se guardará en la variable *palabraLimpia* y se retorna.

Opcionales

El juego contiene algunas modificaciones:

★Música de fondo:

En la función *def main():* que se encuentra en el programa principal se encontró comentada la siguiente línea de código: # pygame.mixer.init()

Con la ayuda de un video tutorial y utilizando las funciones guardadas en la biblioteca de pygame, cargamos la música de la siguiente manera: `pygame.mixer.music.load("")`, dentro de los paréntesis y con comillas se coloca el nombre del archivo a reproducir, y para poder reproducirlo utilizamos: `pygame.mixer.music.play()`, dentro de los paréntesis se coloca un parámetro, en este caso la cantidad de veces que queremos que se reproduzca nuestro archivo).

★Efectos de sonido:

En la función *esCorrecta():* luego de la condición del if donde iguala palabraUsuario *y palabraLimpia,* colocamos los efectos de sonido, *si son iguales reproduce un tipo de sonido y suma 10 puntos, de lo contrario colocamos otro tipo de sonido y no suma puntaje. nombramos una variable llamada sonido la cual guarda un recurso que nos devuelve pygame* `pygame.mixer.Sound("")`, dentro de los paréntesis y comillas se coloca el nombre del archivo, seguido de `sonido.play()`, que utilizamos para reproducir el mismo.

★Fondo del juego:

De igual forma que usamos para la música, utilizamos la biblioteca de pygame, y cargamos el fondo del juego utilizando una variable que llamamos *fondoJuego*, donde guardamos un recurso que nos devuelve pygame como `pygame.image.load('')`, dentro de los paréntesis y las comillas se pone el nombre del archivo, y pusimos sobre el fondo negro que se encontraba en el juego la imagen nueva. Reemplazamos screen.fill(COLOR_FONDO) por `screen.blit(fondoJuego,[0,0])`

★Color de fuente:

Se modificó el color de la fuente en la configuración para que no haga contraste con el nuevo fondo. Para esto se utilizaron los códigos de color RBG.

★Letra ñ:

Primero la agregamos en la lista *abc="ñ",* luego la incluimos en la función *def dameLetra Apretada():* en su equivalente numérico

```
elif key == 241:
   return ("ñ")
```

para que cuando el usuario aprieta esa letra la muestre por pantalla.

Por último, en las funciones *def dibuja*r y def *dibujarSalida* dentro de *extras.py*, iniciamos una variable llamada *letra* que contiene un conjunto de caracteres, para esto se buscó información, donde utilizamos unicode el cual es un formato común de caracteres: `letra=letra.encode("latin1")`.