Universidad Nacional General Sarmiento

Introducción a la programación

TRABAJO PRÁCTICO

PYGAME TutiFrutiUNGS

Turno: noche

Comisión: 07

Profesores:

* Luis Varonesi
* Cinthya Cardozo

Alumnos:

* Agustín Quiroga
* Sebastián Aranda
* Alejandro Mancilla

Introducción:

Se trata del juego TutiFruti y fue pensado para mejorar el vocabulario, la velocidad de escritura, la memoria y principalmente para el entretenimiento de los usuarios.

Este consiste en contestar en el menor tiempo posible y de manera correcta según los ítems (colores, países, animales, nombres, frutas, verduras, marcas). Para que sea considerada correcta la respuesta del usuario, deberá comenzar con la letra seleccionada aleatoriamente por la computadora y respetando cada uno de los ítems que el juego va arrojando en pantalla.

Si la respuesta del usuario es correcta este acumula 10 puntos, en caso que la respuesta sea incorrecta se descontaran 5 puntos y el usuario no ingresa palabra alguna el juego descontara 10 puntos.

Consignas:

Se deben modificar las diferentes funciones que se encuentran en funcionesVacias.py para el correcto funcionamiento del programa y en caso de necesitarlo, crear funciones auxiliares.

Funciones principales:

La función **unaAlAzar**(lista): recibe una lista, el abecedario por ejemplo,

y devuelve una letra al azar.

La función **esCorrecta**(palabraUsuario, letra, item, items, listaDeTodo) recibe una palabra, una letra, un item, la lista de items y la listaDeTodo, y devuelve un puntaje positivo si la palabra es correcta y uno negativo en caso contrario. Se considera correcta solo si la palabra aparece en el item correcto y empieza con la letra correcta.

La función **juegaCompu**(letraAzar, listaDeTodo) dada una letra y la lista de listas devuelve una lista con una palabra al azar de cada item que empiece con esa letra, en caso de no tener palabra para algún item devolverá una cadena vacía en esa posición de la lista. Por ejemplo: si la letra es "m" la lista de ítems = [[“azul”, “celeste”, “marrón”] , [“Argentina" , “Uruguay” , “Venezuela” , “Brasil”] , [“mono” , “jirafa” , “marmota” , “mula”] entonces podrá devolver [ “marrón” , “mono”]

Configuraciones realizadas:

unaAlAzar: Retorna una letra al azar para que se use en el juego.

Se configura esta función, creando una variable a la cual se le asigna el valor seleccionado de forma aleatoria. Esta función recibe una lista y devuelve un elemento al azar utilizando la función random.choice (antes se importó “import random”), y retornando el valor asignado a la variable antes mencionada.

Observaciones: No tuvimos contratiempo al trabajar sobre esta función.

esCorrecta: Verifica si la respuesta del usuario comienza con la letra indicada y si esta se encuentra en el ítem correspondiente.

Esta función fue configurada en un principio utilizando la variable de control del programa principal “i”, para que recorra las distintas listas dentro de la matriz (listaDeTodo), ya que el valor asignado en un principio a la variable “i” es 0 (cero) e iba de forma creciente, lo que facilitaba el recorrido por los distintos ítems dentro de la matriz.

Observaciones: Nos encontramos con varios problemas al configurar esta función:

1. Modificamos la función principal agregando el parámetro i, lo que estaba prohibido.
2. Si el usuario no ingresaba letra alguna y presionaba la tecla Enter, se producía un error y se debía reiniciar el juego.

Para la solución de los problemas nombrados anteriormente, se elimina la modificación que se le había hecho a los parámetros de la función y se agrega una variable de control dentro de la función, realizando así el correcto recorrido dentro de la matriz (listaDeTodo).

También, se soluciona el error que se generaba si el usuario no ingresaba letra alguna, agregando una condición que verifica que si palabraUsuario es igual a vacía, retorne -10.

juegaCompu: Retorna una palabra escogida aleatoriamente, que cumpla condiciones necesaria (que comience con la letra al azar y que la palabra escogida aleatoriamente pertenezca al ítems correspondiente).

Esta función fue configurada tomando en cuenta la variable de control “i” como en el caso anterior y utilizando un random.choice(), esta selecciona una palabra al azar, que comienza con la letra que retorna la función unaAlAzar(), teniendo en cuenta el ítem correspondiente.

Observaciones: la función presenta una falla, entrando en ciertas ocasiones en un ciclo infinito, generando así un error, lo que lleva al reinicio tanto de PyScripter como del Pygame.

Luego, se eliminan espacios en blancos en los archivos de texto (listas utilizadas como items), que era lo que posiblemente generaban que el programa ingrese en un ciclo infinito.

Función agregada:

abrirArchivos:

Esta función recibe una lista guardada en un archivo de texto que es abierta en el programa principal y que se cierra después de la matriz llamada listaDeTodo que recibe todas las listas.

Funciones:

----------------------------------------------------------------------

def unaAlAzar(lista):

letra=random.choice(lista)

return letra

----------------------------------------------------------------------

def esCorrecta(palabraUsuario, letra, item, items, listaDeTodo):

i=0

for elem in items:

if elem==item:

if palabraUsuario=="":

return -10

if (palabraUsuario[0]==unaAlAzar(letra) and palabraUsuario in listaDeTodo[i]):

return 10

else:

return -5

i=i+1

----------------------------------------------------------------------

def juegaCompu(letra, listaDeTodo):

salida=[]

posibles=[]

aleatorio=""

vacio=" "

for lista in listaDeTodo:

for elemento in lista:

if elemento[0]==unaAlAzar(letra):

posibles.append(elemento)

if len(posibles) == 0:

salida.append(vacio)

else:

aleatorio=random.choice(posibles)

salida.append(aleatorio)

posibles=[]

return salida

----------------------------------------------------------------------

def abrirArchivos(lista):

listaNueva=[]

for i in lista:

listaNueva.append(i[:-2])

return listaNueva

----------------------------------------------------------------------

Extras:

* Se le agrega la letra ñ a la variable abc, luego se declara la codificación UTF-8 (Unicode Transformation Format -8), para que Python reconozca los acentos y la ñ.

Declaración de la codificación:

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

* Se importa el codec y se utiliza para la apertura de los archivos de texto para que Python reconozca los acentos y la letra ñ.

Nombre\_Variable = codec.open(“archivo.txt”, “r”, “utf-8”)

* Se configura la tecla ñ en el archivo llamado extras.py, agregando un elif a la función dameLetraApretada(key).

elif key == K\_SEMICOLON:

return("ñ")

* Se modifica la función dibujar() en el archivo extras.py, y se coloca un condicional a la variable ren4 que si la letra seleccionada es la letra ñ, no se aplique la condición letra.upper(), ya que el pygame tiene problemas al mostrar en pantalla la letra Ñ en mayúscula, por lo tanto será la única letra que se mostrara en minúscula.

También, se modifica a color rojo la línea horizontal que divide la pantalla donde se muestra lo que escribe el usuario.

pygame.draw.line(screen, (200, 10, 10), (0, ALTO - 70), (ANCHO, ALTO - 70), 5)

* Al igual que en el anterior se modifica en la función dibujarSalida() el color de la línea horizontal a color rojo.

pygame.draw.line(screen, (200, 10, 10), (0, ALTO - 70), (ANCHO, ALTO - 70), 5)

* En el archivo configuración.py, se modificaron los colores de la palabra ingresada por el usuario, el tiempo, los puntos e items a negro, y el color de fondo a blanco.

COLOR\_TEXTO = (0, 0, 0)

COLOR\_FONDO = (255, 255, 255)

* Se modifica el programa principal:

Se agrega una imagen de fondo mediante la variable fondoPresentacion:

fondoPresentacion = pygame.image.load("ImagenSonido/python.png").convert()

screen.blit(fondoPresentacion,[0,0])

Se agrega música, mediante el comando:

pygame.mixer.music.load("ImagenSonido/musica.ogg") # Carga la música.

pygame.mixer.music.play() # Comienza la música.

Se agrega una condición dentro del ciclo (while) en el que se ejecuta el juego, que se utiliza para finalizar la reproducción de la música.

if i==len(items):

pygame.mixer.music.stop()

Por último, se agregan dos variables llamadas acierto y error, donde se cargan los sonidos utilizados al acertar una palabra o al fallar.

acierto = pygame.mixer.Sound("ImagenSonido/aciertos.ogg")

error = pygame.mixer.Sound("ImagenSonido/error.ogg")

y se agrega una condición if para reproducir el sonido al acertar o al fallar, activando el sonido cuando a la variable sumar se le asigna el valor 10 (acertar), -10 o -5 (errar).

if sumar==10:

acierto.play()

else:

if sumar==-10 or sumar==-5:

error.play()