

Universidad Nacional General Sarmiento

Introducción a la programación

Profesores:   
Alberto Lecot y Esteban Fassio

Estudiantes:   
Martin Servetti  
Cristian Sante

Comisión: 5

Título del informe: Diccionar.io

**INTRODUCCIÓN**

El presente trabajo práctico consiste en la creación de un juego programándolo con Pyscripter. El juego está conformado por una serie de códigos conectados entre sí, entre ellos Principal, FuncionesVacias, Extras y Configuración. Nosotros nos concentraremos en FuncionesVacias, donde crearemos las funciones básicas del juego. También, si es necesario, mencionaremos si agregamos cosas en el resto del código.

El juego consiste en que aparezcan en pantalla definiciones sacadas de un diccionario y el jugador deber escribir la palabra que se está definiendo. Se contará con la ayuda en pantalla de 2 letras pertenecientes a la palabra y también la longitud de la palabra. El jugador comienza el juego con 5 “vidas”. En caso de no adivinar la palabra tiene la opción de presionar 1 y saltear la palabra pero en ese caso se le restara una vida, como también se le restara una vida si falla la palabra. Si llegado el momento el usuario se queda sin vidas el juego llega a su fin. Por otro lado cada vez que adivine una palabra se le sumarán puntos.

A continuación detallaremos las funciones utilizadas en FuncionesVacias.

**FUNCIONES VACIAS:**

* **def lectura(listaPalabras, listaDefiniciones):** Esta función lee el archivo diccionario.txt y carga en listaPalabras las palabras y en listaDefiniciones las definiciones. Cada palabra tiene que ocupar la misma posición que su respectiva definición.
* **def eligePalabra(listaPalabras, letras):** elige de la lista palabras una que contiene las letras al azar.
* **def damePosicion(palabra,listaPalabras):** devuelve la posición de un elemento en una lista de elementos.
* **def dameDefinicion(palabra,listaPalabras,listaDefiniciones):** devuelve la definición de la palabra.
* **def azar(listaPalabras):** elige 2 letras al azar.
* **def puntuar(candidata,palabra):** recibe una palabra y la candidata y si es correcta retorna el puntaje correspondiente.

-Si es vocal suma 1 punto.

-Si es consonante suma 2 puntos.

-Si es consonante difícil (j,k,q,w,x,y,z) suma 5 puntos.

* **def esCorrecta(candidata,palabra):** recibe la candidata y devuelve un True si es correcta o un False si no lo es.
* **def esta(palabra,listaPalabras):** recibe un elemento y una lista y devuelve True si encuentra al elemento en la lista o False en caso contrario.
* **def vidasrestantes(screen,contador):** esta función dibuja en pantalla las vidas restantes del usuario.

**Dificultades que encontramos:**

1- En la función lectura tuvimos inconvenientes con la palabra “migración”. El problema se generaba por la vocal con tilde. La cual no era registrada cuando el usuario la ingresaba por teclado. Para solucionar eso usamos:

if j=="ó":

j="o"

Con esto le indicamos que si encuentra la vocal “ó” la reemplace por la misma letra pero sin tilde.

2- Al guardar el archivo FuncionesVacias y volver a abrirlo Python no reconocía la cadena **“ó”**. Esto generaba que cada vez que abríamos el archivo teníamos que corregir este problema. Al parecer, para poder utilizar caracteres extendidos en Python es necesario indicarle que estamos haciendo tal cosa, ya que sino, por defecto únicamente contempla que estamos utilizando ASCII de 7bits, en los que no se contemplan ni eñes ni acentos ni ningún otro carácter que no esté en el teclado americano. Para solucionar este pequeño problema lo que hicimos fue, al comienzo del código poner ***# -\*- coding: UTF-8 -\*-***

3- Otro inconveniente fue con la palabra “intervencionismo”. Esta al aparecer su longitud en pantalla excedía el ancho de la misma. Para corregirlo lo que hicimos fue modificar en la parte de Extras un valor para que la palabras y los “ \_ “ que aparecen en pantalla se corran un poco hacia la izquierda.

4- Para que las palabras de las definiciones no se corten lo que hicimos fue modificar en el archivo Extras:

for letra in definicion:

if (x>ANCHO-250) and letra==" ":

x=40

y=y+TAMANNO\_LETRA

else:

x=x-1

**Otros detalles en la parte del código principal:**

Para que las palabras no se repitan una vez que el jugador las adivinó o las salteó lo que hicimos fue utilizar el índice de la palabra y definición que salió, y luego aplicar un POP para eliminarlos de las lista. Con esto evitamos que se repitan. La parte del código es la siguiente:

indice=damePosicion(palabra,listaPalabras)

listaPalabras.pop(indice)

listaDefiniciones.pop(indice)

Para controlar las vidas del jugador se utiliza un contador. Este inicia en 5 y por cada error al tratar de adivinar la palabra o en caso de saltearla se le restará 1.