

### Licenciatura en Sistemas

# Trabajo Práctico

“El Juego del Mago Goma”

### Introducción a la Programación

(Segundo semestre de 2018)

Resumen: El trabajo consiste en Realizar el famoso juego de las sílabas, en el cual se escribe una palabra cuya primera sílaba sea la última de la que aparece en pantalla. La computadora hace lo mismo y comienza una  
batalla en búsqueda de palabras que cumplan ese requisito.

Integrantes: Barrientos, Lucas (wallsofjericho7@hotmail.com)

Ramua, Ian (ianramua@yahoo.com)

**1. INTRODUCCION**

El juego actualmente consta de varios archivos, un archivo con el programa principal, un archivo con la configuración; un archivo con los extras; un archivo con la función separador de silabas y por ultimo un archivo con funciones vacías. En este último archivo se encuentran las funciones esenciales para ejecutar correctamente el programa, pero se encuentran solo sus argumentos y no están definidas. Los argumentos de las funciones vacías a realizar fueron las siguientes:

* La función **lectura** (archivo, salida).
* La función **nuevaPalabra** (silabas).
* La función **silabasTOpalabra** (silaba).
* La función **palabraTOsilaba** (palabra).
* La función **dameUltimaSilaba** (enSilabas).
* La función **damePrimeraSilaba** (enSilabas).
* La función **esValida** (palabraUsuario, palabraUsuarioEnSilabas, palabraActual,  
  palabraEnSilabas, listaPalabrasDiccionario).
* La función **Puntos** (palabraUsuario).
* La función **procesar** (palabraUsuario, palabraUsuarioEnSilabas, palabraActual,  
  palabraEnSilabas, listaPalabrasDiccionario).
* La función **buscarPalabraQueEmpieceCon** (silaba, lemarioEnSilabas).

**2. DESARROLLO**

Al iniciar el trabajo, primeramente enfatizamos resolver las definiciones de aquellas funciones que era de menor complejidad, es decir, las funciones que tomaban no más de un argumento (**nuevaPalabra**, **silabasTOpalabra**, **dameUltimaSilaba** y **damePrimeraSilaba**) exceptuando la función **Puntos** y la función **palabraTOsilaba**, en donde la primera de estas fue realizada al final por problema de comprensión y la segunda ya estaba realizada previamente en el archivo “funcionesSeparador.py”. Luego, definimos las funciones más complejas, que eran aquellas funciones que tomaban más de un argumento (**lectura**, **esValida**, **procesar** y **buscarPalabraQueEmpieceCon**).

Las funciones de menor complejidad no resultaron de por de más complicadas, fueron realizadas varias pruebas en donde a pesar de algún que otro error, no se originó una dificultad considerable, exceptuando la función **Puntos**, que resultaba compleja resolverla si no teníamos una mirada general de las funciones más complejas como eran las funciones **esValida** y **procesar**. Luego, una vez terminado con las funciones menos complicadas, encaramos el desarrollo de las funciones más complicadas, en donde encontramos problemas con el desarrollo de los requisitos que estas funciones requerían. A pesar de incontables métodos para llegar a una solución, notábamos que las funciones contenían argumentos que estaban de más o que no eran necesarios establecerlos. Una vez consultado con los profesores, se decidió modificar determinados argumentos que serán remarcados en **FUNCIONALIDADES PRINCIPALES**.

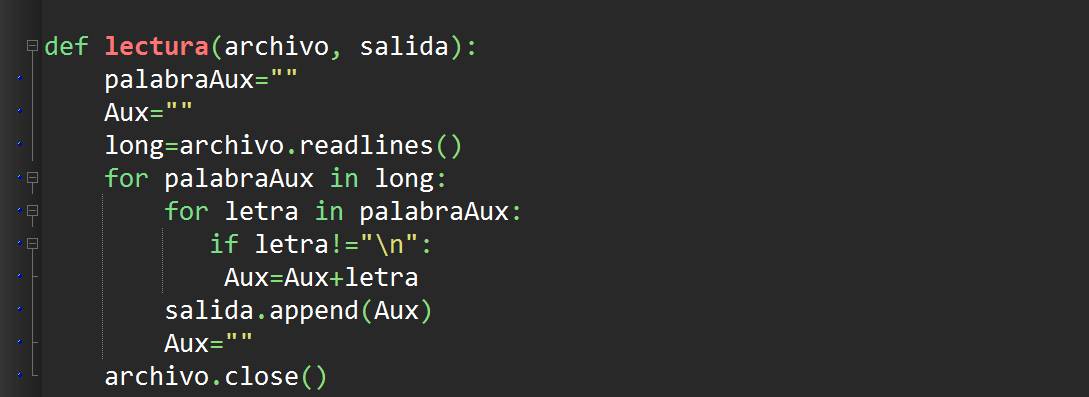
**3. FUNCIONALIDADES PRINCIPALES**

A continuación se detallará por orden de aparición, la idea general de cada función vacía, su resolución, el código realizado, una descripción de los parámetros que toma y los valores que devuelve y/o modifica.

* La función **lectura** (archivo, salida):

El trabajo de esta función consiste en tomar un archivo de texto abierto en el programa principal y almacenarla en una lista. Esta función solo consiste en guardar el contenido de un archivo de texto y guardarla en una lista sin devolver ningún valor o parámetro.

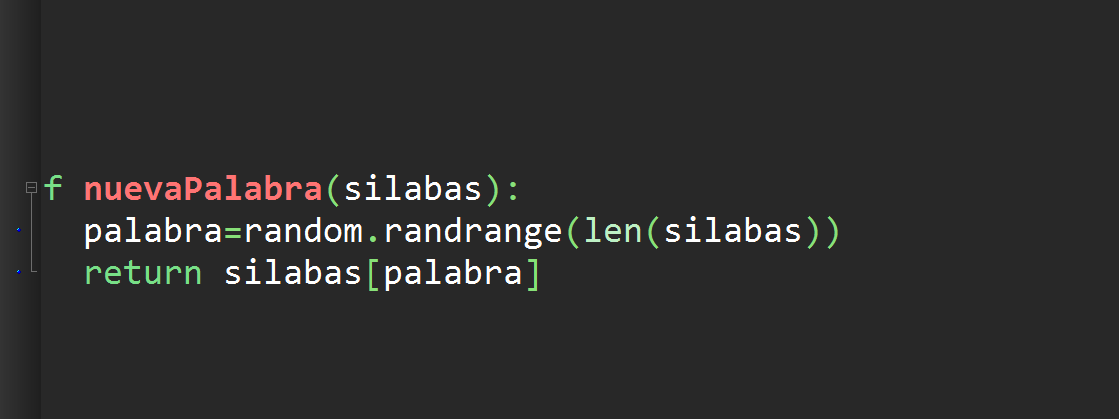
Desarrollo de la función:



* La función **nuevaPalabra** (silabas):

Esta función se encarga de recibir una lista con todas las palabras ya separadas en silabas, en donde elige una de estas palabras y la devuelve.

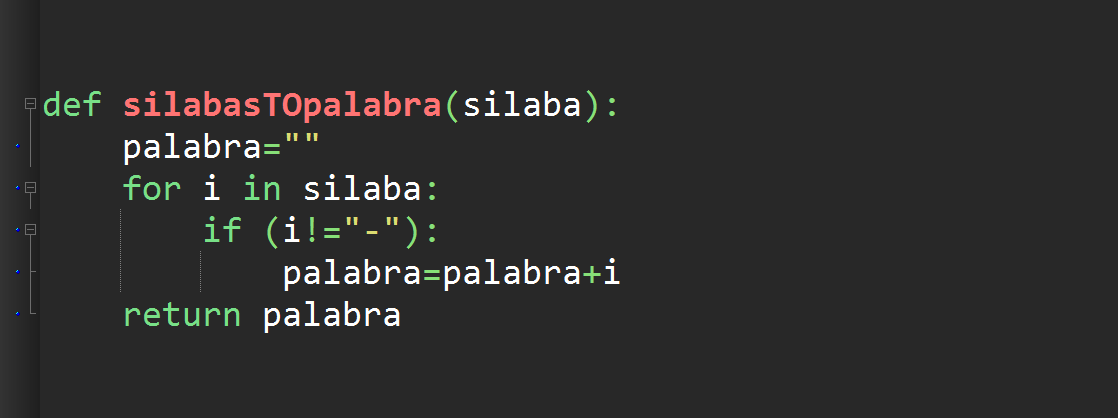
Desarrollo de la función:

****

* La función **silabasTOpalabra** (silaba):

La rol que cumple esta función es recibir una palabra separada en silabas y recorre la palabra para encontrar “-“, al encontrarlo lo ignora y al finalizar el recorrido devuelve la palabra sin separaciones.

Desarrollo de la función:



* La función **palabraTOsilaba** (palabra):

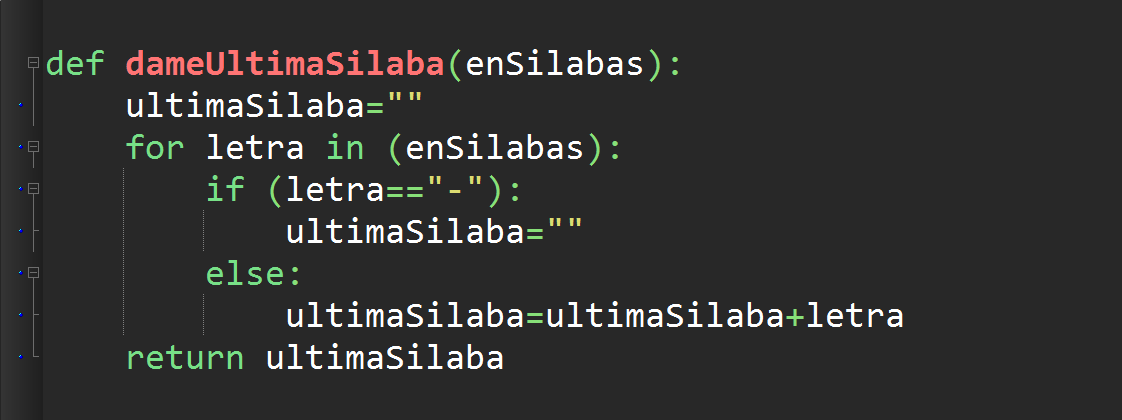
El trabajo de esta función consiste en separar en silabas una palabra ingresada, agregando un “-“ en medio de cada silaba encontrada.

El desarrollo de esta función ya estaba previamente realizado por los profesores, por el cual se expondrá dicha función al final de este trabajo, en el **PROGRAMA PRINCIPAL** (“funcionSeparador.py”).

* La función **dameUltimaSilaba** (enSilabas):

Esta función consiste recibir una palabra separada en silabas y la última silaba señalada en esta palabra.

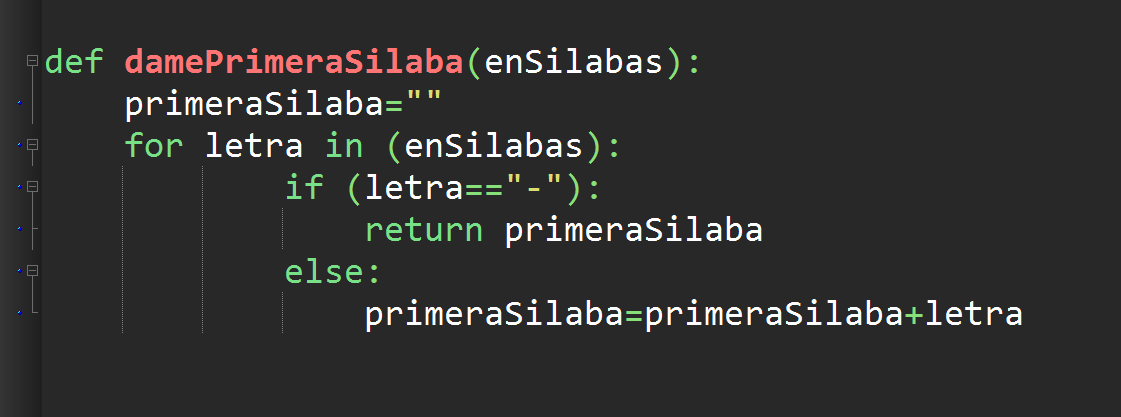
Desarrollo de la función:



* La función **damePrimeraSilaba** (enSilabas):

En este caso, esta función recibe una palabra separada en silabas y la recorre letra por letra hasta encontrar el primer “-“, al encontrarlo devuelve la primera silaba de esta palabra.

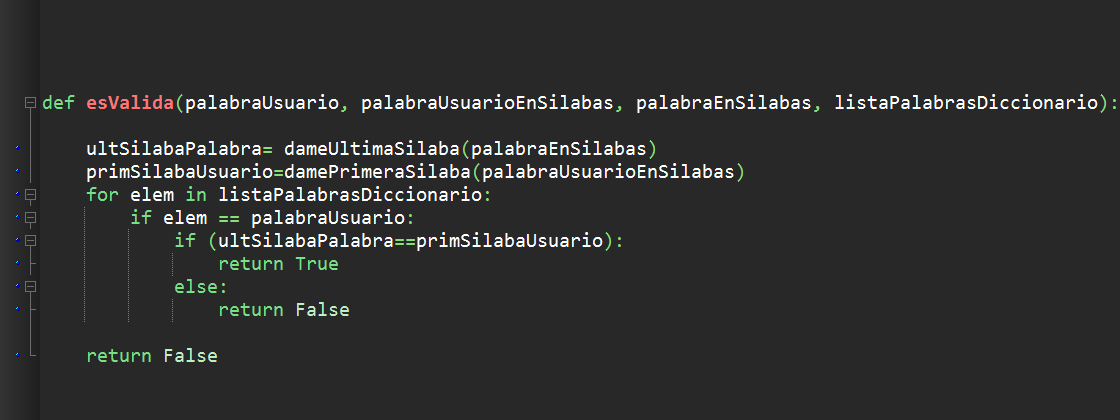
Desarrollo de la función:



* La función **esValida** (palabraUsuario, palabraUsuarioEnSilabas, palabraActual,  
  palabraEnSilabas, listaPalabrasDiccionario):

En esta función decidimos eliminar el parámetro “palabraActual” porque resultaba redundante a la hora de encontrar una solución. Una vez dicho esto, esta función tiene como objetivo reconocer si la palabra que ingreso el usuario es correcta, es decir, que se encuentre en el lemario de palabras y que además la primera silaba de la “palabraUsuario” sea la misma silaba que la última de la “palabraEnSilabas”. De forma tal que al ser correcta esta función devolverá un True, en caso contrario devolverá un False.

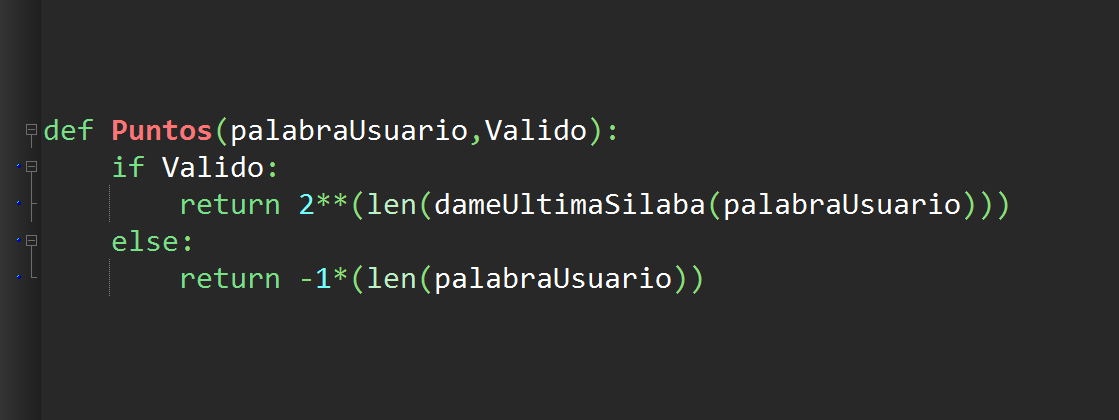
Desarrollo de la función:



* La función **Puntos** (palabraUsuario):

El labor de esta función consiste en recibir una palabra y al cumplir los requisitos que establece la función **esValida** (palabraUsuario, palabraUsuarioEnSilabas,  
palabraEnSilabas, listaPalabrasDiccionario), devuelve 2\*\* la longitud de la última silaba si la palabra ingresada cumple con los requisitos y si es errónea, resta la longitud de la palabraUsuario.

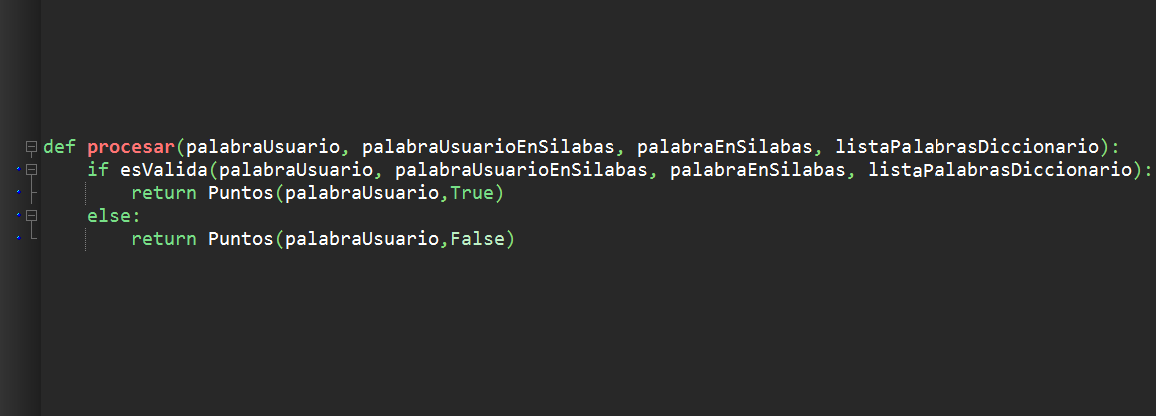
Desarrollo de la función:

****

* La función **procesar** (palabraUsuario, palabraUsuarioEnSilabas, palabraActual,  
  palabraEnSilabas, listaPalabrasDiccionario):

Como lo realizado en la función **esValida**, decidimos no prescindir del argumento “palabraActual” para optimizar el trabajo de la función. Esta función se encarga de verificar si la palabraUsuario es correcta y al chequearla devolver un puntaje utilizando la función anteriormente definida, **Puntos** (palabraUsuario).

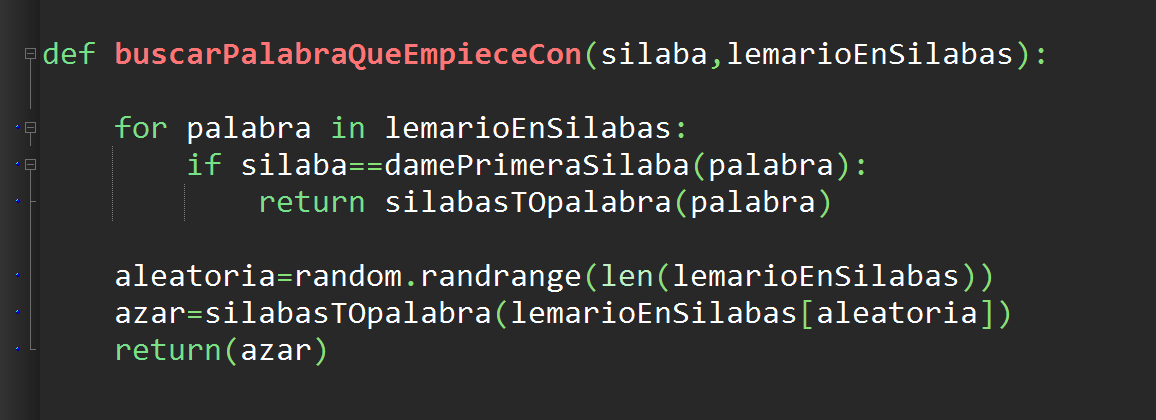
Desarrollo de la función:

****

* La función **buscarPalabraQueEmpieceCon** (silaba, lemarioEnSilabas):

Esta función recibe una silaba y a partir de esa silaba busca dentro del lemarioEnSilabas palabras que empiecen con esa silaba, al finalizar la búsqueda, toma una palabra separada en silabas del listado y la devuelve sin separaciones.

Desarrollo de la función:

****

**5. PROGRAMA PRINCIPAL**

A continuación, para cerrar el trabajo, se presentará todos los comandos del trabajo que estarán bajo al nombre de “funcionesRESUELTO.py” y además, todos los comandos que contenían los archivos “configuración.py”, “extras.py”, “funcionesSeparador.py” y “principal.py” de manera tipeada.

**“configuración.py”:**

from collections import namedtuple

TAMANNO\_LETRA = 20

TAMANNO\_LETRA\_GRANDE = 80

FPS\_inicial = 3

TIEMPO\_MAX = 61

ANCHO = 800

ALTO = 600

COLOR\_LETRAS = (20,200,20)

COLOR\_FONDO = (0,0,0)

COLOR\_Ñ =(200,60,50)

COLOR\_TEXTO = (200,200,200)

COLOR\_TIEMPO\_FINAL = (200,20,10)

Punto = namedtuple('Punto','x y')

**“extras.py”:**

import pygame

from pygame.locals import \*

from configuracion import \*

def dameLetraApretada(key):

if key == K\_a:

return("a")

elif key == K\_b:

return("b")

elif key == K\_c:

return("c")

elif key == K\_d:

return("d")

elif key == K\_e:

return("e")

elif key == K\_f:

return("f")

elif key == K\_g:

return("g")

elif key == K\_h:

return("h")

elif key == K\_i:

return("i")

elif key == K\_j:

return("j")

elif key == K\_k:

return("k")

elif key == K\_l:

return("l")

elif key == K\_m:

return("m")

elif key == K\_n:

return("n")

elif key == K\_o:

return("o")

elif key == K\_1:

return("ñ")

elif key == K\_p:

return("p")

elif key == K\_q:

return("q")

elif key == K\_r:

return("r")

elif key == K\_s:

return("s")

elif key == K\_t:

return("t")

elif key == K\_u:

return("u")

elif key == K\_v:

return("v")

elif key == K\_w:

return("w")

elif key == K\_x:

return("x")

elif key == K\_y:

return("y")

elif key == K\_z:

return("z")

elif key == K\_KP\_MINUS:

return("-")

elif key == K\_SPACE:

return(" ")

else:

return("")

def escribirEnPantalla(screen, palabra, posicion, tamano, color):

defaultFont= pygame.font.Font( pygame.font.get\_default\_font(), tamano)

ren = defaultFont.render(palabra, 1, color)

screen.blit(ren, posicion)

def dibujar(screen, palabraUsuario, palabraActual, puntos, segundos):

defaultFont= pygame.font.Font( pygame.font.get\_default\_font(), TAMANNO\_LETRA)

defaultFontGrande= pygame.font.Font( pygame.font.get\_default\_font(), TAMANNO\_LETRA\_GRANDE)

#Linea Horizontal

pygame.draw.line (screen, (255,255, 255), (0, ALTO-70), (ANCHO, ALTO-70), 5)

#muestra lo que escribe el jugador

screen.blit (defaultFont.render (palabraUsuario, 1, COLOR\_TEXTO), (190, 570))

#muestra el puntaje

screen.blit (defaultFont.render ("Puntos: " + str (puntos), 1, COLOR\_TEXTO), (680, 10))

#muestra los segundos y puede cambiar de color con el tiempo

if(segundos<15):

ren = defaultFont.render("Tiempo: " + str(int(segundos)), 1, COLOR\_TIEMPO\_FINAL)

else:

ren = defaultFont.render("Tiempo: " + str(int(segundos)), 1, COLOR\_TEXTO)

screen.blit(ren, (10, 10))

#muestra la palabra

screen.blit(defaultFontGrande.render(palabraActual, 1, COLOR\_LETRAS), (ANCHO//2-len(palabraActual)\*TAMANNO\_LETRA\_GRANDE//4,ALTO-400))

#muestra su longitud

## screen.blit(defaultFontGrande.render(str(len(palabraActual)), 1, (200,20,10)), (ANCHO-400,ALTO-500))

**“funcionSeparador.py”**

class char():

def \_\_init\_\_(self):

pass

class char\_line():

def \_\_init\_\_(self, word):

self.word = word

self.char\_line = [(char, self.char\_type(char)) for char in word]

self.type\_line = ''.join(chartype for char, chartype in self.char\_line)

def char\_type(self, char):

if char in set(['a', 'á', 'e', 'é','o', 'ó', 'í', 'ú']):

return 'V' #strong vowel

if char in set(['i', 'u']):

return 'v' #week vowel

if char=='x':

return 'x'

if char=='s':

return 's'

else:

return 'c'

def find(self, finder):

return self.type\_line.find(finder)

def split(self, pos, where):

return char\_line(self.word[0:pos+where]), char\_line(self.word[pos+where:])

def split\_by(self, finder, where):

split\_point = self.find(finder)

if split\_point!=-1:

chl1, chl2 = self.split(split\_point, where)

return chl1, chl2

return self, False

def \_\_str\_\_(self):

return '<'+self.word+':'+self.type\_line+'>'

def \_\_repr\_\_(self):

return '<'+repr(self.word)+':'+self.type\_line+'>'

class silabizer():

def \_\_init\_\_(self):

self.grammar = []

def split(self, chars):

rules = [('VV',1), ('cccc',2), ('xcc',1), ('ccx',2), ('csc',2), ('xc',1), ('cc',1), ('vcc',2), ('Vcc',2), ('sc',1), ('cs',1),('Vc',1), ('vc',1), ('Vs',1), ('vs',1), ('vxv',1), ('VxV',1), ('vxV',1), ('Vxv',1)]

for split\_rule, where in rules:

first, second = chars.split\_by(split\_rule,where)

if second:

if first.type\_line in set(['c','s','x','cs']) or second.type\_line in set(['c','s','x','cs']):

#print 'skip1', first.word, second.word, split\_rule, chars.type\_line

continue

if first.type\_line[-1]=='c' and second.word[0] in set(['l','r']):

continue

if first.word[-1]=='l' and second.word[-1]=='l':

continue

if first.word[-1]=='r' and second.word[-1]=='r':

continue

if first.word[-1]=='c' and second.word[-1]=='h':

continue

return self.split(first)+self.split(second)

return [chars]

def \_\_call\_\_(self, word):

return self.split(char\_line(word))

def lectura(archivo, silabas):

for linea in archivo.readlines():

silabas.append(linea[0:-1])

def separador(palabra):

s = silabizer()

si=s(palabra)

nueva=""

for elem in si:

encontre=False

elem=str(elem)

for e in elem:

if e!="<" and e!=":" and not encontre:

nueva=nueva+e

if e==":":

nueva=nueva+"-"

encontre=True

return nueva[:-1]

**“principal.py”:**

#! /usr/bin/env python

import os, random, sys, math

import pygame

from pygame.locals import \*

from configuracion import \*

from extras import \*

from funcionesSeparador import \*

from funcionesRESUELTO import \*

#Funcion principal

def main():

#Centrar la ventana y despues inicializar pygame

os.environ["SDL\_VIDEO\_CENTERED"] = "1"

pygame.init()

#pygame.mixer.init()

#Preparar la ventana

pygame.display.set\_caption("El juego del Mago Goma...")

screen = pygame.display.set\_mode((ANCHO, ALTO))

#tiempo total del juego

gameClock = pygame.time.Clock()

totaltime = 0

segundos = TIEMPO\_MAX

fps = FPS\_inicial

puntos = 0

palabraUsuario = ""

lemarioEnSilabas=[]

listaPalabrasDiccionario=[]

archivo= open("lemario.txt","r")

archivo2= open ("lemarioSilabas.txt","r")

#lectura del diccionario

lectura(archivo, listaPalabrasDiccionario)

#lectura del archivo en silabas

lectura(archivo2, lemarioEnSilabas)

#elige una al azar

palabraEnSilabas=nuevaPalabra(lemarioEnSilabas)

palabraActual=silabasTOpalabra(palabraEnSilabas)

dibujar(screen, palabraUsuario, palabraActual, puntos,segundos)

while segundos > fps/1000:

# 1 frame cada 1/fps segundos

gameClock.tick(fps)

totaltime += gameClock.get\_time()

if True:

fps = 3

#Buscar la tecla apretada del modulo de eventos de pygame

for e in pygame.event.get():

#QUIT es apretar la X en la ventana

if e.type == QUIT:

pygame.quit()

return()

#Ver si fue apretada alguna tecla

if e.type == KEYDOWN:

letra = dameLetraApretada(e.key)

palabraUsuario += letra

if e.key == K\_BACKSPACE:

palabraUsuario = palabraUsuario[0:len(palabraUsuario)-1]

if e.key == K\_RETURN:

#pasa la palabra a silabas

palabraUsuarioEnSilabas=palabraTOsilaba(palabraUsuario)

#chequea si es correcta y suma o resta puntos

puntos += procesar(palabraUsuario, palabraUsuarioEnSilabas,palabraActual, palabraEnSilabas, listaPalabrasDiccionario)

#busca la ultima silaba y busca una palabra que empiece asi

silaba=dameUltimaSilaba(palabraUsuarioEnSilabas)

palabraEnSilabas=buscarPalabraQueEmpieceCon(silaba,lemarioEnSilabas)

palabraActual=silabasTOpalabra(palabraEnSilabas)

palabraUsuario = ""

segundos = TIEMPO\_MAX - pygame.time.get\_ticks()/1000

#Limpiar pantalla anterior

screen.fill(COLOR\_FONDO)

#Dibujar de nuevo todo

dibujar(screen, palabraUsuario, palabraActual,puntos,segundos)

pygame.display.flip()

while 1:

#Esperar el QUIT del usuario

for e in pygame.event.get():

if e.type == QUIT:

pygame.quit()

return

archivo.close()

archivo2.close()

#Programa Principal ejecuta Main

if \_\_name\_\_ == "\_\_main\_\_":

main()

**“funcionesRESUELTO.py”:**

from principal import \*

from configuracion import \*

import random

import math

def lectura(archivo, salida):

palabraAux=""

Aux=""

long=archivo.readlines()

for palabraAux in long:

for letra in palabraAux:

if letra!="\n":

Aux=Aux+letra

salida.append(Aux)

Aux=""

archivo.close()

def nuevaPalabra(silabas):

palabra=random.randrange(len(silabas))

return silabas[palabra]

def silabasTOpalabra(silaba):

palabra=""

for i in silaba:

if (i!="-"):

palabra=palabra+i

return palabra

def palabraTOsilaba(palabra):

nueva=separador(palabra)

return nueva

def dameUltimaSilaba(enSilabas):

ultimaSilaba=""

for letra in (enSilabas):

if (letra=="-"):

ultimaSilaba=""

else:

ultimaSilaba=ultimaSilaba+letra

return ultimaSilaba

def damePrimeraSilaba(enSilabas):

primeraSilaba=""

for letra in (enSilabas):

if (letra=="-"):

return primeraSilaba

else:

primeraSilaba=primeraSilaba+letra

def esValida(palabraUsuario, palabraUsuarioEnSilabas, palabraEnSilabas, listaPalabrasDiccionario):

ultSilabaPalabra= dameUltimaSilaba(palabraEnSilabas)

primSilabaUsuario=damePrimeraSilaba(palabraUsuarioEnSilabas)

for elem in listaPalabrasDiccionario:

if elem == palabraUsuario:

if (ultSilabaPalabra==primSilabaUsuario):

return True

else:

return False

return False

def Puntos(palabraUsuario,Valido):

if Valido:

return 2\*\*(len(dameUltimaSilaba(palabraUsuario)))

else:

return -1\*(len(palabraUsuario))

def procesar(palabraUsuario, palabraUsuarioEnSilabas, palabraEnSilabas, listaPalabrasDiccionario):

if esValida(palabraUsuario, palabraUsuarioEnSilabas, palabraEnSilabas, listaPalabrasDiccionario):

return Puntos(palabraUsuario,True)

else:

return Puntos(palabraUsuario,False)

def buscarPalabraQueEmpieceCon(silaba,lemarioEnSilabas):

for palabra in lemarioEnSilabas:

if silaba==damePrimeraSilaba(palabra):

return silabasTOpalabra(palabra)

aleatoria=random.randrange(len(lemarioEnSilabas))

azar=silabasTOpalabra(lemarioEnSilabas[aleatoria])

return(azar)