Nombres: Aquino Gaspar Abel, Casais Nicolás Javier, Kemerer Ferrisi Mauricio Comisión 5, primer semestre año 2022

Trabajo Práctico Final

El presente trabajo abordará la creación y la funcionalidad de un juego, el cual estará compuesto por una función principal y varias funciones extras que permitirán el correcto desarrollo del mismo. Habiendo recibido la función principal realizada, procedimos a agregar las funciones que se en contraban vacías. Para ello, definimos la función lectura que recibe el archivo a leer; recorremos cada una de las palabras y les sacamos el salto de línea, las palabras con ñ (ya que esta letra no está entre las letras configuradas) y convertimos las mayúsculas en minúsculas para evitar erro res. Luego de estos pasos, se acumulan en 3 listas distintas que corresponderán a los distintos niveles que posee el juego.

```
def lectura(archivo, salida, salida2, salida3, puntos):
    long = archivo.readlines()
    for elem in long:
        if "ñ" not in elem:
            elem = elem.strip("\n")
            elem.lower()
            if len(elem) <= 6:
                salida.append(elem)
            if len (elem) > 6 and len(elem) <= 10:
                salida2.append(elem)
            if len(elem) > 10:
                salida3.append(elem)
```

La segunda función (nuevaPalabra) tomará una palabra aleatoria de la lista que corresponda se gún el nivel y la devolverá para que el jugador realice la separación en sílabas. Para esta función, consideramos esta alternativa como la más simple de obtener una palabra aleatoria ya que permi tía, en pocas líneas, mostrar una palabra dependiendo del nivel que se esté jugando.

```
def nuevaPalabra(lista, lista2, lista3, puntos):
    if puntos < 50:
        palabraRandom = random.choice(lista)
    if puntos >= 50 and puntos <= 100:
        palabraRandom = random.choice(lista2)
    if puntos > 100:
        palabraRandom = random.choice(lista3)
    return palabraRandom
```

La tercera función (silabasTOpalabra) recorrerá la palabra devuelta por la función anterior y le sa cará los guiones y los espacios con los que el usuario puede realizar la separación en sílabas. Con respecto a esta función, consideramos que esta alternativa nos permitiría recorrer la palabra y sacar los caracteres que sirven como separadores.

```
def silabasTOpalabra(silaba):
    palabra_nueva = ""
    for char in silaba:
        if char == "-" and char == " ":
            char = ""
            palabra_nueva = palabra_nueva + char
```

```
        palabra_nueva = palabra_nueva + char
    return palabra_nueva
```

La cuarta función (palabraTOsilaba)llama a la función separador que fue realizada por los profeso res y separa la palabra que aparece en pantalla. Al estar realizada la función para separar, nos resultó rápido tener que sólo hacer el llamado a la misma.

```
def silabear(palabra):
    palabraTexto = str(palabra)
    a = pyphen.Pyphen(lang="es")
    palabraSilaba = a.inserted(palabraTexto)
    return palabraSilaba
```

La quinta función (esCorrecta)comprobará que la palabra introducida por el usuario esté separada en sílabas correctamente, teniendo en cuenta que el usuario puede separar la palabra con espa cio o guion. Para esta función, creíamos que debíamos realizar un trabajo similar a la tercera fun ción ya que debíamos tener en cuenta los caracteres que introduzca el usuario para separar.

```
def esCorrecta(palabraEnSilabasEnPantalla, palabra):
    palabra_nueva = ""
    for char in palabra:
        if char == " ":
            char = "-"
        palabra_nueva = palabra_nueva + char
    if palabraEnSilabasEnPantalla == palabra_nueva:
        return True
```

La sexta y última función (puntaje) devolverá el puntaje según la longitud de la palabra y el nivel en el que se encuentre. En el primer nivel, las palabras correctas darán 3 puntos; en el segundo, darán 5 puntos y, en el último nivel, darán 10 puntos. Con respecto a las palabras incorrectas, res tarán 1 punto. Para los puntajes, debimos decidir la manera en la que sumaría el acierto por parte del usuario y, para ello, nos encontramos con distintas alternativas. Finalmente, optamos por otor gar mayor puntaje mientras el nivel sea más difícil.

```
def puntaje(palabra):
    if len(palabra) <= 6:
        return 3
    elif len(palabra) > 6 and len(palabra) <=10:
        return 5
    else:
        return 10
```

Función agregada

La función menuYReglas toma tres parámetros: una bandera, una segunda bandera y un screen (que es la pantalla del juego) las banderas sirven para saltar de un ciclo while a otro, lo que nos permitirá con un espacio ir hacia la sección de reglas y apretando la p (play) avanzar ya directa mente hacia el juego, se puede jugar sin necesidad de ver y pasar por la sección de las reglas (¡no es recomendado!), sólo para aquellos impacientes quienes quieran jugar ante todo riesgo. Esta función se encuentra en el archivo extras, junto con la función dibujar y dameLetra Apretada, las cuales son

compañeras en la interacción con el usuario, para profundizar un poco más la función destacamos que:
-Ambos bucles en la función tienen cargada una imagen la cual servirá de fondo de la sec ción.
-Ambos bucles finalizan gracias al uso de las banderas (variables ya implementadas en el pro grama principal y las cuales son parámetros de la función).
-Se dispone de un color agregado (el blanco) en el archivo de configuración cuya variable se nombró COLOR_CONF, como el color de las configuraciones del menú y las reglas. -En la pri mera pantalla se describe el nombre del juego y que opciones se disponen, al apretar espacio se avanza hacia las reglas, al apretar p (play) se juega directamente. -En la sección de reglas se explican las reglas básicas del juego, en la esquina inferior izquierda podemos ver los dere chos reservados del juego a la Comisión 5.

Sistema de puntuación y niveles:

NIVELES

El juego dispone de tres (3) niveles de dificultad, los cuales no son elegibles, sino que se miden por el sistema de puntuación al cuál se explicará más adelante. El primer nivel toma palabras que tienen seis letras o menos, es decir que presenta una dificultad menor. El segundo nivel toma palabras mayores a seis letras pero menores o iguales a diez, como si se tratara de un nivel intermedio. El tercer y último nivel del juego, toma palabras con una cantidad de letras mayor a diez, pre sentando así la dificultad más alta del juego del mago goma.

SISTEMA DE PUNTUACIÓN

Este sistema es aquel que se rige gracias a la función puntaje(palabra) la cual como se ob serva toma como parámetro la palabra que va a salir en pantalla como se indica en el pro grama principal: puntaje(palabraEnPantalla)
La función expresa:

```
def puntaje(palabra):
    if len(palabra) <= 6:
        return 3
    elif len(palabra) > 6 and len(palabra) <=10:
        return 5
    else:
        return 10
```

Es decir, si la palabra en pantalla que se tiene que separar en sílabas tiene una longitud de 6 le tras o menos, la puntuación será de 3 puntos, si las letras son mayores a 6 pero iguales o menores que 10, se obtiene 5 puntos y en cambio si tiene más de 10 letras, se otorga la canti dad máxima de puntaje, que es 10 puntos.
No debe dejar de tener en cuenta que la función puntaje es llamada siempre y cuando la función esCorrecta devuelva un valor verdadero (True) el cual nos indica que la separación en sílabas de nosotros como usuario ha sido exitosa y nos permite así llamar a la función puntaje para recibir nuestra respectiva suma de puntos.

FUNCIÓN SILABEAR

La función silabear cumplirá el mismo rol que la función separador. Esta función importa Pyphen que realizará esta tarea ya que posee un diccionario y separa las palabras que se encuentran en él. De las distintas alternativas que nos encontramos en internet, fue la que nos ayudó a separar en sílabas en pocas líneas de código.

SISTEMA DE RECORDS

Para crear un marcador con puntajes tuvimos que crear una serie de funciones, que se conectan entre sí.
La primera función “record” tal y como se hizo con la función “Lectura”, importamos datos de un archivo “.txt” para luego guardar, dichos datos, en dos listas que se corresponden, en la primera se guardan los puntos de los usuarios, y en la otra, se guarda el puntaje.

```
def record(nombres, puntos):
    lista = []
    archivo= open("record.txt","r")
    lineas = archivo.readlines()

    for i in lineas:
        i = i.strip("\n")
        lista.append(i)

    for j in range(9,0,-2):
        pun = lista[j]
        puntos.append(pun)

    for k in range(8,(0-1),-2):
        k = int(k)
        nom = lista[k]
        nombres.append(nom)
```

Continuamos con la función “ordenaLista” con la cual se puede ordenar de mayor a menor los puntajes de una lista, pero, sin afectar a la correspondencia que tiene con otra lista:

```
def ordenaLista(listaP, listaN):
    for recorrido in range(1, len(listaP)):
      for posicion in range(len(listaP) - recorrido):

        if int(listaP[posicion]) > int(listaP[posicion + 1]):
            temp = listaP[posicion]
            temp1 = listaN[posicion]
            listaP[posicion] = listaP[posicion + 1]
            listaN[posicion] = listaN[posicion + 1]
            listaP[posicion + 1] = temp
            listaN[posicion + 1] = temp1
```

Seguimos utilizando otra función, “darVuelta” en esta caso, para mantener la correspondencia entre las dos listas, esta vez, para invertir la lista, de modo que se mantenga un orden decreciente en cuanto a los puntajes de los usuarios.

```
def darVuelta(lista1, lista2):
    listaP = []
    listaN = []
    for i in range(5, 0, -1):
        listaP.append(lista1[i])
        listaN.append(lista2[i])
    lista1 = listaP
    lista2 = listaN
```

Con las anteriores listas, podemos, ahora sí, añadir a la lista de récords, el posible nuevo puntaje de un usuario. Para eso, usaremos la función “reemplazaRec” para añadir a las listas, el puntaje del usuario, en caso de superar alguno, este tomará su lugar y el puntaje anterior disminuye de posición en el marcador.

```
def reemplazaRec(listaN, listaP, newPuntaje, newUser):
    listaN.append(newUser)
    listaP.append(newPuntaje)
    ordenaLista(listaP, listaN)
    listaP.reverse()
    listaN.reverse()
```

Finalmente, para guardar de nuevo los datos en el archivo “txt”, para que, se guarde en caso de que haya un nuevo récord, se utiliza la función “guardoRec”, mediante un ciclo tomamos los cinco mejores puntajes, excluyendo al sexto (el cual puede el puntaje del usuario, o de algún puntaje superado) para luego, escribirlo en el archivo que se utilizó en la primera función.

```
def guardoRec(listaNombres, listaPuntos):
    lista = []
    for i in range(0, 5):
        n = listaNombres[i]
        p = str(listaPuntos[i])
        lista.append(n)
        lista.append(p)
    f = open("record.txt", "w")
    for k in range(0, len(lista)):
        carac = str(lista[k])
        f.write(carac + "\n")
    f.close()
```