Trabajo práctico en etapas

FIUBLE

PARTE 1 - 1C 2022

Indice

1. Introduccion

1.1. Objetivo

Construir un programa en etapas, que se pueda jugar, teniendo en cuenta las reglas que se detallan en cada una de las etapas.

1.2. Historia

El juego FIUBLE será un clon de WORDLE, juego que tomó popularidad hace no mucho tiempo en 2021.

Para la persona que no conozca el juego, puede visitar el siguiente link https://wordle.danielfrg.com/ para familializarse.

1.2.1. Jugabilidad del WORDLE

En este juego el objetivo es adivinar una palabra secreta y para hacerlo se tiene un máximo de 6 intentos.

En todos los intentos se arriesga una palabra de la misma cantidad de letras que la que se intenta adivinar, es decir, si la palabra a adivinar tiene 5 letras entonces el arriesgo debe tener 5 letras, ni más ni menos.

Luego de un arriesgo, se le indica al jugador por **cada una** de las letras de la palabra arriesgada si esta:

- No aparece en la palabra, con el color **gris**
- Aparece en la palabra pero no en el lugar correcto, con el color **amarillo**
- Aparece en la palabra y está en el lugar correcto, con el color **verde**

En el caso que en la palabra aparezca 2 veces o más la misma letra y si un arriesgo tiene 2 veces esa misma letra, entonces ambas van a aparecer en amarillo, una verde y una en amarillo o las 2 en verde, dependiendo el caso.

Ejemplo:
Palabra a adivinar: CASAS
Arriesgo: SOFAS

Resultado: **SOFAS**

Se considera que la persona ganó el juego si en menos de 6 intentos adivina la palabra de 5 letras, caso contrario se considera que perdió.

1.3. Producto final

Al terminarse de desarrollar esta primera parte del trabajo práctico se debe tener un programa que permita una jugabilidad de 2 jugadores, donde el ganador de la partida sea aquel que acumule la mayor cantidad de puntos en una partida.

2. Etapa 1 - Versión Reducida

En esta etapa el objetivo es programar una pequeña parte del juego.
En esta sección se pide que:

- La palabra a adivinar sea siempre la misma y tenga 5 letras
- Se mostrará la siguiente interfaz al usuario inicialmente:

> Palabra a adivinar: ? ? ? ? ?
> Arriesgo:

- Luego de un arriesgo, se debe mostrar cada letra con su color correspondiente (ver seccion 1.2.1) y mostrar la palabra la cual se estaba intentando adivinar. Por ejemplo:

> Palabra a adivinar: C A S A S
> Arriesgo: **S O F A S**

  Se debe considerar el caso de las letras repetidas (ver seccion 1.2.1)

  Recomendacion: Dejar espacios en blanco entre que se ingresa el arriesgo y se muestra la respuesta al usuario.

- Luego que se muestra el resultado al usuario se debe mostrar inmediatamente debajo el mensaje "Ganaste!" o "Perdiste!" en caso de que adivine la palabra o no.

3. Etapa 2 - Validación de ingresos

En esta etapa, se agrega a la etapa anterior la validación del ingreso del usuario, esto es:

- La palabra a ingresar debe tener la misma cantidad de caracteres que la que se intenta adivinar
- No se pueden ingresar números ni caracteres especiales
- Un arriesgo puede contener acentos pero estos deben ser tratados como palabras sin acentos, es decir "SÓFÁS" genera el mismo resultado que "SOFAS"
- Los ingresos del usuario pueden ser todas letras mayúsculas, todas letras minúsculas o una mezcla de ambos, todos estos casos deben funcionar como si fuesen uno solo, es decir "SOFAS" genera el mismo resultado que "soFas" o "sofas". En pantalla se debe mostrar todo en letras mayúsculas.

Si por alguno de estos casos el ingreso es inválido, entonces se debe informar al usuario detallando cuale/s es/son la/s razon/es por la/s cual/es es inválido y se debe volver a pedir el ingreso de un nuevo arrieso, esto debe repetirse hasta que el usuario ingrese un arriesgo válido o se detenga el programa.

4. Etapa 3 - Aumentando la complejidad de FIUBLE

En esta etapa el juego va a empezar a tomar forma, para esto se requiere que:

- El usuario tenga un máximo de **5** intentos para adivinar la palabra oculta, en caso de que el usuario:

  - adivine la palabra oculta en 5 intentos o menos, se muestra (igual que antes) el mensaje "Ganaste!"
  - no adivine la palabra oculta en 5 intentos o menos, se muestra (igual que antes) el mensaje "Perdiste!"

- La palabra a adivinar ya no es siempre la misma, esta debe ser elegida de forma aleatoria al inicio de cada partida de entre las **palabras válidas** (sigue siendo de 5 letras). Ver Anexo.
- La interfaz mostrada al usuario debe ser la siguiente:

```
Palabra a adivinar: ? ? ? ? ?
? ? ? ? ?
? ? ? ? ?
? ? ? ? ?
? ? ? ? ?
? ? ? ? ?
Arriesgo:
```

Si luego de esto el usuario arriesga BARDO, se debe mostrar el resultado del arriesgo de la siguiente forma (notese que "Palabra a adivinar" sigue indicando la palabra a adivinar)

```
Palabra a adivinar: ? A ? ? ?
B A R D O
? ? ? ? ?
? ? ? ? ?
? ? ? ? ?
? ? ? ? ?
Arriesgo:
```

Si luego de esto se arriesga CARPA, entonces se debe mostrar el resultado del arriesgo de la siguiente forma

```
Palabra a adivinar: C A ? ? ?
B A R D O
C A R P A
? ? ? ? ?
? ? ? ? ?
? ? ? ? ?
Arriesgo:
```

Como ultimo ejemplo, ahora el usuario decide arriesgar VELOZ, el resultado se debe mostrar de la siguiente manera

```
Palabra a adivinar: C A ? ? ?
B A R D O
C A R P A
V E L O Z
? ? ? ? ?
? ? ? ? ?
Arriesgo:
```

Pregunta extra: Cual es la palabra oculta en este caso?

Recordar que sin importar si se pierde o se gana, la palabra oculta debe ser mostrará al usuario.

5. Etapa 4 - Tiempo de juego, puntaje y volver a jugar

En esta etapa se va a cronometrar el tiempo de juego (cuanto se tarda en adivinar la palabra oculta), dar la opción de volver a jugar luego de perder o ganar y llevar un sistema de puntos en base a las partidas jugadas en una corrida del programa. Para esto se agregan los siguientes requerimientos:

- Luego que se termina de adivinar una palabra (si esta se adivinó) se debe mostrar después del mensaje de "Ganaste!" el mensaje "Tardaste XX minutos y YY segundos en adivinar la palabra". Por ejemplo:

```
Palabra a adivinar: B A R D O
B A R D O
? ? ? ? ?
? ? ? ? ?
? ? ? ? ?
? ? ? ? ?
Arriesgo:
Ganaste! Tardaste 0 minutos y 10 segundos en adivinar la palabra
```

  No se debe mostrar el tiempo si no se adivina la palabra.

- Luego que se termina de adivinar una palabra, sin importa si realmente se adivinó la palabra o no, se debe mostrar los puntos obtenidos/perdidos... El sistema de puntajes será el siguiente:

Intentos	Puntos
1	50
2	40
3	30
4	20
5	10
5+	-100

  La interfaz que se le va a mostrar al usuario va a ser la siguiente:

```
Palabra a adivinar: B A R D O
B A R D O
? ? ? ? ?
? ? ? ? ?
? ? ? ? ?
? ? ? ? ?
Arriesgo:
Ganaste! Tardaste 0 minutos y 10 segundos en adivinar la palabra

Obtuviste un total de 50 puntos
```

  Si se pierde, la interfaz va a ser la siguiente:

Palabra a adivinar: P A G A R
P U N T O
A R C O S
P A L A S
P A J A R
P A S A R
Arriesgo:
Perdiste!

Perdiste un total de 100 puntos

- Los puntos se deben poder acumular ya que se permite volver a jugar.

  Por lo tanto si un jugador que tenía acumulados 50 puntos pierde, entonces sus puntos bajan a -50. Siempre se le tiene que indicar al usuario cuantos puntos tenía previamente (a excepción del primer juego). Un ejemplo de esto es:

Palabra a adivinar: P A G A R
P U N T O
A R C O S
P A L A S
P A J A R
P A S A R
Arriesgo:
Perdiste!

Perdiste un total de 100 puntos, tenes acumulados -50 puntos

Si luego de esta partida gana la siguiente, entonces se vuelven a acumular los puntos

Palabra a adivinar: B A R D O
B A R D O
? ? ? ? ?
? ? ? ? ?
? ? ? ? ?
? ? ? ? ?
Arriesgo:
Ganaste! Tardaste 0 minutos y 10 segundos en adivinar la palabra

Obtuviste un total de 50 puntos, tenes acumulados 0 puntos.

Luego que se termine cada partida se le preguntará al usuario si desea o no seguir jugando, se debe validar que este ingreso sea válido. La interfaz finalmente deberá quedar de la forma:

```
Palabra a adivinar: B A R D O
B A R D O
? ? ? ? ?
? ? ? ? ?
? ? ? ? ?
? ? ? ? ?
Arriesgo:
Ganaste! Tardaste 0 minutos y 10 segundos en adivinar la palabra

Obtuviste un total de 50 puntos, tenes acumulados 0 puntos.

Desea jugar otra partida? (S/N)
```

En caso de que el usuario ingrese S o s, entonces se inicia otra partida; si ingresa N o n, entonces se termina el programa y en cualquier otro caso vuelve a pedir un ingreso hasta que éste sea válido.

6. Etapa 5 - Extensión a 2 jugadores

En esta etapa se modificará el programa ya existente para que 2 jugadores puedan competir entre si.

Para poder realizar esto se deben cumplir los siguiente requerimientos:

- Al inciar el programa, se debe pedir a los usuarios que ingresen sus nombres
- Aleatoriamente se elige uno de ellos y se le otorga el primer turno
- Los usuarios se van alternando para arriesgar una palabra, el programa debe indicar claramente qué turno es de que usuario
- Si un usuario adivina la palabra, entonces éste gana los puntos correspondientes al acierto (contando los intentos de la misma forma que se contaba de forma individual pero sumando el total de intentos de ambos usuario). Por ejemplo: Si el usuario A inicia adivinando la palabra y la adivina en su segundo intento, entonces éste va a ganar 30 puntos (ya que el total de intentos es de 3; 2 de él y 1 del otro usuario B)
- Si el usuario A adivina la palabra, entonces no sólo éste gana XX puntos, sinó que el usuario B va a perder esos XX puntos.
- Si la palabra no es adivinada, entonces el usuario que inició adivinando perderá 100 puntos y el otro usuario perderá 50 puntos.
- Luego de terminar una partida, se mostrará para cada usuario los puntos obtenidos/perdidos en esa partida y los acumulados hasta el momento. Se preguntará a los usuarios si desean seguir jugando, en caso que respondan:
  - **SI**, entonces el jugador que no habia empezado adivinando en la anterior partida, va a ser el primero en adivinar en esta.
  - **NO**, entonces se mostrará un mensaje que diga " El ganador es {NOMBRE USUARIO GANADOR} con un total de XX puntos"

  Recordar: Para cada partida debe mostrarse el tiempo en adivinar la palabra, si es que ésta se adivina.

7. Condiciones de entrega

Las siguientes condiciones deben ser respetadas para que la entrega sea considerada válida:

1. El programa debe ser desarrollado en Python, aplicando correctamente los conceptos dados en clase y respetando las buenas prácticas descriptas en el PEP de la cátedra. Cada función que forma parte del código debe tener debajo de su firma, una descripción corta de cuál es su objetivo y quien es el autor ó responsable de dicha función.

2. El código correspondiente a la Parte 1, debe ser subido al campus. El nombre a dar al archivo será TP1_NombreGrupo.py. Deberán reemplazar NombreGrupo, por el nombre dado a su grupo. Si la entrega está compuesta por más de un archivo .py, generar un .zip con todos los archivos .py, y nombrarlo de igual modo, pero con extensión zip.

3. Deberán grabar 2 videos y subirlos a un canal de Youtube, ó a Google Drive. El primer video, cada integrante del equipo, deberá contar mostrando el código, qué parte estuvo bajo su responsabilidad y los puntos de solución dados, que considere más relevantes. El video total no debe superar los 10 minutos. Comenzar cada uno de los relatos, diciendo el nombre y apellido del quien relata. La calidad del video debe ser buena, y debe haber homogeneidad en las presentaciones.

4. Deberán grabar un segundo video, en el que se muestre al menos una jugada completa, y que contemple distintos casos que muestran que la aplicación responde según lo esperado. Deberán ir relatando los eventos de la jugada. En este caso el video puede estar realizado por 1 único integrante.

8. Anexo

Tódo lo que se encuentra en esta sección será proporcionado por la cátedra en un archivo llamado **utiles.py** el cual deberá ser **importado** en su proyecto

8.1. Colores

Para poder imprimir por pantalla utilizando colores, se recomienda que utilicen el siguiente código

```
def obtener_color(color):
    colores = {
        "Verde" = "\x1b[32m"
        "Amarillo" = "\x1b[33m"
        "GrisOscuro" = "\x1b[90m"
        "Defecto" = "\x1b[39m"
    }
    return colores[color]
```

Siendo el modo de utilización el siguiente:

```
print(obtener_color("Verde") + "A " + obtener_color("Amarillo") + "B " +
    obtener_color("GrisOscuro") + "C" + obtener_color("Defecto"))
```

Es importante que siempre terminen asignando el color como *Defecto* porque si no las siguientes impresiones se imprimirán con el último color utilizado.

8.2. Palabras válidas

Para obtener las palabras que serán consideradas como válidas por el programa, se hará llamado a la siguiente función que retornará la lista de palabras de 5 letras permitidas:

```
def obtener_palabras_validas():
    return ["abran", "abria", "acojo", "actuo", "aguda", "agudo", "algas", "almas",
     "alojo", "alojo", "altas", "altos", "andes", "anima", "apodo", "arcos", "ardan
    ", "ardes", "arios", "azote", "bajas", "bajan", "bardo", "bates", "bayas", "
    bebes", "besen", "besos", "botas", "bodas", "bondi", "bonos", "borre", "botan",
     "botes", "bruta", "cagas", "cajas", "callo", "calma", "campo", "canas", "capas
    ", "caros", "casan", "casas", "cazan", "cazas", "caida", "caido", "ceder", "
    cenas", "cepas", "ceras", "cerdo", "cerco", "ceros", "cerro", "ciega", "ciego",
     "cines", "clava", "clavo", "calvo", "cogen", "coger", "colas", "coles", "coman
    ", "conos", "capas", "capaz", "copos", "copas", "coral", "corra", "corre", "
    cosas", "coses", "croar", "cruje", "cuida", "culta", "culto", "cunas", "curso",
     "dagas", "datos", "debes", "dedos", "densa", "dijes", "doman", "domar", "donan
    ", "donas", "dones", "dotes", "dudan", "dunas", "duros", "echas", "echan", "
    edita", "ellos", "emana", "emoji", "enoja", "enojo", "entes", "envio", "erizo",
     "errar", "error", "espia", "euros", "evita", "evito", "falla", "falta", "fetos
    ", "filas", "firme", "focos", "fosos", "frias", "fugas", "fumar", "gafas", "
    galas", "galos", "ganas", "gases", "gatos", "genes", "giras", "giros", "goles",
     "gorra", "grave", "grite", "grito", "hielo", "heces", "habia", "hacen", "hacia
    ", "hacha", "hecho", "hijas", "hilos", "hojas", "hugos", "ideas", "iglus", "
    islas", "india", "jefes", "jerga", "jodas", "jugos", "jamon", "kenia", "kodak",
     "kayak", "lacra", "libro", "lados", "lagos", "lamen", "larga", "latas", "lazos
    ", "lejos", "lenta", "lento", "libre", "linda", "locas", "locos", "lomos", "
    loros", "losas", "luces", "leche", "lucha", "luche", "magos", "malas", "males",
     "malos", "mamas", "manca", "manco", "manos", "manda", "mapas", "marco", "mares
    ", "matar", "mayas", "mazos", "mesas", "metas", "metes", "miles", "minas", "
    mirar", "mitos", "modas", "mojar", "modos", "mojan", "moles", "monas", "monos",
     "monte", "moras", "moros", "mozas", "mocos", "mulas", "multa", "muros", "musas
    ", "nabos", "nadar", "naves", "nazis", "nubes", "nudos", "nieve", "nunca", "
    nacer", "necio", "necia", "obras", "odiar", "odios", "ollas", "ombus", "ondas",
     "onzas", "opera", "orcas", "orden", "otras", "ovulo", "paces", "pajas", "palas
    ", "palma", "palos", "panes", "parda", "parar", "pares", "pases", "patos", "
    pecas", "peces", "penas", "pense", "perdi", "pesas", "pesca", "pesos", "pesas",
     "peces", "pican", "pedir", "pisar", "pleno", "plena", "pocas", "pocos", "podar
    ", "poder", "podia", "ponen", "poner", "posee", "pozos", "p jar", "pujan", "
```

```
    pulen", "pulir", "pumas", "puros", "quema", "quise", "quito", "queso", "rabia",
     "rabos", "ramos", "ratas", "ratos", "redes", "rejas", "remos", "retos", "reyes
    ", "rifas", "rimas", "riman", "rimar", "roban", "rodan", "rojas", "rojos", "
    rosas", "rotar", "rugir", "runas", "rusas", "rusos", "sabia", "serio", "sacar",
     "salgo", "salga", "salta", "salto", "selva", "sanar", "sapos", "sedes", "santa
    ", "seria", "serio", "sobar", "sonar", "subir", "suela", "sumar", "super", "
    tacos", "talar", "tejas", "temas", "temen", "temer", "tener", "tenso", "tensa",
     "tiros", "titan", "togas", "tomar", "tonta", "tonto", "torpe", "traje", "trios
    ", "urnas", "untar", "umami", "urgar", "vacas", "vagos", "vagas", "vasca", "
    velos", "venas", "vidas", "vigas", "vinos", "volar", "votos", "votar", "video",
     "yates", "yemas", "yenes", "yogur", "zetas", "zonas", "zurda", "zurdo", "zorro
    "]
```