Universidad de
SanAndrés

Ahorcado

Pensamiento Computacional

Trabajo Práctico N°2

1. Objetivos:

Este trabajo práctico busca implementar una versión del famoso juego “Ahorcado” en su versión de consola interactiva.

2. Alcance del Trabajo Práctico

Temas que comprende este trabajo práctico:

- Variables y tipos
- Funciones: reuso de comportamiento y encapsulamiento
- Condicionales y ciclos
- Secuencias: cadenas, listas, tuplas
- Archivos
- Modularización: uso de módulos propios y estándares de python.

3. Introducción

El ahorcado (también llamado *hangman*) es un juego de adivinanzas de lápiz y papel para dos o más jugadores. Un jugador piensa en una palabra, frase u oración y el otro trata de adivinarla según lo que sugiere por letras o dentro de un cierto número de oportunidades.

En su versión mas sencilla, es un simple juego por turnos en el que un “anfitrión” propone una palabra secreta la cual debe ser adivinada por el jugador antes de que el mismo se quede sin oportunidades.

Las reglas son:

1. Las palabras a utilizar no pueden contener caracteres fuera del alfabeto inglés (ASCII 97-122).
2. El jugador debe decidir una posible letra para rellenar la palabra por turno.
3. Si la palabra secreta contiene una aparición o más de la letra seleccionada por el jugador. Todas las apariciones son reveladas y no pierde oportunidades. Por ejemplo, si la palabra a adivinar es “computadora”, la letra seleccionada es la o, y originalmente se mostraba _ _ _ _ _ _ _ a _ _ _ a, entonces pasa a mostrarse _ o _ _ _ _ a _ o _ a).
4. En caso que la letra seleccionada por el jugador no se encuentre en la palabra el mismo perderá una oportunidad.
5. El jugador tiene **5 oportunidades** para adivinar la palabra.
6. Si el jugador **logra** descifrar la palabra antes de quedarse sin oportunidades **gana**.
7. Si el jugador **no logra** descifrar la palabra antes de perder sus 5 oportunidades **pierde**.

4. Desarrollo del TP / Especificaciones

El programa de tener 2 modos de uso. En un modo, la computadora elige una palabra y el usuario debe adivinarla, en el segundo modo, el usuario elige una palabra y la computadora debe adivinarla.

> **Nota:** el programa debe ser insensible a mayúsculas y minúsculas.

La solución debe cumplir los siguientes requisitos.

El programa debe comenzar con un menú principal el cual presente las siguientes opciones:

- Play
- See words list
- Quit

Ejemplo de lo que se ve en la consola de python:

```
1. Play
2. See words list
3. Quit
Select at most one option >
```

4.1 Opción “Play”

En caso de seleccionar `Play` se pasa a mostrar el menú principal de esta modalidad, con las siguientes acciones:

- Human hangman
- Computer hangman
- Go back

Ejemplo de lo que se ve en la consola de python:

```
1. Human hangman
2. Computer hangman
3. Go back
>
```

Opción “Human hangman”

En la modalidad de *Human hangman*, la computadora elige al *azar* una palabra de un conjunto de palabras disponibles y luego el usuario debe adivinarla. Para ello, comienza el juego mostrando una serie de _ indicando la cantidad de letras, los intentos restantes y solicita el ingreso de una letra. Por ejemplo, si la palabra es *pensamiento* se verá lo siguiente:

```
1 _ _ _ _ _ _ _ _ _ _ _ _
```

Luego se mostrarán la cantidad de intentos restantes y se solicitará el ingreso de una letra. Si el juego recién comienza, se verá lo siguiente:

```
1 5 tries remaining.
2 Choose a character:
```

En este momento, y en cualquier momento que tenga la oportunidad de adivinar una letra, el usuario debe ingresar un caracter o una palabra, y apretar ENTER para confirmar su decisión. El programa interpreta de la misma manera las letras en mayúscula como en minúscula (por ejemplo, si el usuario propone la letra “A”, el programa la interpreta como una “a”, si ingresa la palabra “monTOtO” se interpreta como “montoto”).

Si se ingresa una letra y la misma se encuentra en la palabra secreta, entonces se descubrirán todas las letras acertadas hasta el momento, y **no** se descontarán intentos. Por ejemplo, si la palabra es *pensamiento* y la letra seleccionada es la `e`, se verá lo siguiente:

```
1 _ e _ _ _ _ _ e _ _ _   (e)
```

Nótese que al lado de la palabra (incompleta por el momento) aparece una letra *e*, entre paréntesis. Esa letra indica las letras que se han probado, e irá mostrando todas las letras que el usuario pruebe, sin repeticiones (es decir, si el usuario ingresa 2 veces la e, únicamente se mostrará una vez).

Si la palabra no contiene a la letra ingresada, se descontará un intento a los intentos restantes. Por ejemplo, si la palabra es *pensamiento* y la letra seleccionada es la `z`, entonces se descontará un intento.

Luego, si aún quedan intentos, se mostrará un mensaje con la cantidad de intentos restantes y se pedirá el ingreso de una nueva letra. Esta secuencia se repetirá hasta que se agoten los intentos o se descubra la palabra. Por ejemplo, podría continuar con la siguiente secuencia:

```
_ e _ _ _ _ _ e _ _ _   (e)
5 tries remaining.
Choose a character: r
_ e _ _ _ _ _ e _ _ _   (er)
4 tries remaining.
Choose a character: s
_ e _ s _ _ _ e _ _ _   (ers)
4 tries remaining.
Choose a character: x
_ e _ s _ _ _ e _ _ _   (ersx)
```

Si el usuario completa la palabra, o ingresa una palabra en lugar de una letra y acierta, se mostrará el mensaje

```
Congratulations! The word was pensamiento and you discovered it!
```

Si no adivinó la palabra en la cantidad de intentos estipulada, o ingresa una palabra y no acierta, entonces se mostrará un mensaje en la consola de python donde se le dice al jugador cual era la palabra a adivinar, por ejemplo:

```
Bad luck! There are no more tries. The word was: pensamiento.
```

Al finalizar la partida, debemos volver al menú principal.

Opción: “Computer hangman”

En la modalidad de *Computer hangman*, el usuario elige una palabra, válida dentro de un conjunto de palabras disponibles y luego la computadora debe adivinarla.

La estrategia (o algoritmo) que utiliza la computadora para proponer las letras afecta las chances de que ésta gane la partida. Por ejemplo, un algoritmo válido sería que proponga letras al azar, pero muy probablemente esto la haga perder siempre. Es decir que el diseño del algoritmo que sigue la computadora para proponer las letras tienen un impacto sobre sus chances de ganar. Alentamos a que diseñen algoritmos que tiendan a maximizar las chances de ganar de la computadora.

Para comenzar, lo primero que debe ocurrir es que se le pida al usuario una palabra y se aguarda su ingreso. Al ingresar la palabra, se verifica que la misma exista dentro del conjunto de palabras conocidas por la computadora, y de ser así, se guarda como palabra a adivinar. Si la palabra no está presente en las palabras conocidas, se pide que ingrese una palabra nuevamente. Por ejemplo, si las palabras conocidas son “pensamiento” y “computacional”, una secuencia de ejecución sería la siguiente:

```
Please, input a word for the computer to guess: computasional
'computasional' is not in the list of possible words.
Please, input a word for the computer to guess: computacional
Secret word saved!
```

Luego, se muestra la cantidad de intentos restantes, la computadora muestra su selección y espera la respuesta del usuario. Por ejemplo, si la palabra es *computacional* se verá lo siguiente:

```
_ _ _ _ _ _ _ _ _ _ _ _ _
5 tries remaining.
Is the letter 'z' present in the word? [y/n]:
```

Al dar la respuesta, el juego debe descontar intentos restantes si la letra no está presente en la palabra elegida, o volver a preguntar sin descontar intentos en caso contrario. El juego repetirá la secuencia hasta que la computadora adivine la palabra o los intentos se agoten. Al ocurrir uno de estos dos eventos, se deberá mostrar en pantalla un mensaje que indique el resultado, tal como se hizo en la versión *Human hangman*. Al igual que en el caso del *Human hangman*, se deben mostrar las letras que ya se han utilizado para adivinar la palabra. También se admite que la computadora intente acertar la palabra completa, en cuyo caso, en lugar de preguntar por una letra con el mensaje `Is the letter 'X'present in the word? [y/n]` (donde X es una letra cualquiera), se debe preguntar por una palabra completa con el mensaje `Is the word 'word'? [y/n]` (donde word es una palabra cualquiera). Si arriesga una palabra, el desarrollo es igual que con el humano: gana o pierde automáticamente si acertó o no acertó, respectivamente.

Al finaliza la partida, volvemos al menú principal.

4.2 Opción: “See words list”

En este caso, el programa muestra el listado de palabras y vuelve al menu. Por ejemplo, si las palabras que conoce el programa son “pensamiento” y “computacional”, mostrará en la consola:

```
List of possible words: computacional, pensamiento.
```

4.3 Opción: “Quit”

Si el usuario ingresa esta opción, el programa debe finalizar de manera normal (sin arrojar errores).

4.4 Ejemplos de uso

A continuación se muestran ejemplos de uso del juego para distintos casos.

Ver lista de palabras

```
Let's play hangman!
What do you want to do?
1. Play
2. See words list
3. Quit
Select at most one option > 2
list of possible words: computacional, pensamiento.
What do you want to do?
1. Play
2. See words list
3. Quit
```

Una partida donde el humano debe adivinar la palabra

```
What do you want to do?
1. Play
2. See words list
3. Quit
Select at most one option > 1
How do you want to play?
1. Human hangman
2. Computer hangman
3. Go back
Select at most one option > 1

_ _ _ _ _
5 tries remaining.
Choose a character: z
_ _ _ _ _    (z)
4 tries remaining.
Choose a character: a
_ _ _ _ _    (az)
3 tries remaining.
Choose a character: e
_ _ _ _ _    (aez)
2 tries remaining.
Choose a character: i
_ i _ _ _    (aeiz)
2 tries remaining.
Choose a character: p
_ i _ _ _    (aeipz)
```

```
1 try remaining.
Choose a character: m
Bad luck! There are no more tries. The word was: linux.
What do you want to do?
1. Play
2. See words list
3. Quit
Select at most one option > 3
```

Una partida donde la computadora debe adivinar

```
What do you want to do?
1. Play
2. See words list
3. Quit
Select at most one option > 1
How do you want to play?
1. Human hangman
2. Computer hangman
3. Go back
Select at most one option > 2
Please, input a word for the computer to guess: computasional
'computasional' is not a known word.
Please, input a word for the computer to guess: computacional
Secret word saved!

_ _ _ _ _ _ _ _ _ _ _ _ _
5 tries remaining.
Is the letter 'z' present in the word? [y/n]: n
_ _ _ _ _ _ _ _ _ _ _ _ _    (z)
4 tries remaining.
Is the letter 'a' present in the word? [y/n]: y
_ _ _ _ _ _ a _ _ _ _ a _    (a)
4 tries remaining.
Is the letter 'c' present in the word? [y/n]: y
c _ _ _ _ _ a c _ _ _ a _    (acz)
4 tries remaining.
Is the letter 'r' present in the word? [y/n]: n
c _ _ _ _ _ a c _ _ _ a _    (acrz)
3 tries remaining.
Is the letter 'p' present in the word? [y/n]: y
c _ _ p _ _ a c _ _ _ a _    (acprz)
3 tries remaining.
Is the letter 'o' present in the word? [y/n]: y
c o _ p _ _ a c _ o _ a _    (acoprz)
3 tries remaining.
Is the letter 'v' present in the word? [y/n]: n
c o _ p _ _ a c _ o _ a _    (acoprvz)
2 tries remaining.
Is the letter 'h' present in the word? [y/n]: n
```

```
c o _ p _ _ a c _ o _ a _    (achoprvz)
1 try remaining.
Is the letter 'x' present in the word? [y/n]: n
Bad luck! There are no more tries. The word was: computacional.
What do you want to do?
1. Play
2. See words list
3. Quit
Select at most one option > 3
```

5. Restricciones

La realización de este trabajo comprende las siguientes restricciones.

Integrantes

Le trabajo puede realizarse en grupos de hasta 2 (dos) integrantes.

Implementación

La implementación cumple con las especificaciones desarrolladas en el punto **4. Desarrollo del TP / Especificaciones**.

Reusabilidad

El o los módulos desarrollados tiene que poder ser *importados* en python, poniendo a disponibilidad de un tercero las funciones desarrolladas, pero sin ejecutar el juego. El juego debe ser ejecutado únicamente mediante la ejecución del programa o la ejecución de una función. Por ejemplo, para jugar se puede ejecutar el siguiente comando desde una consola de Python:

```
>>> from hangman import run_game
>>> run_game()
```

Módulos

Sólo está permitido utilizar, además de los módulos estándar de python, los siguientes módulos:

- módulos propios

- `numpy`
- `scipy`
- `matplotlib`

Si se desea utilizar cualquier otro módulo, por ejemplo `pytorch`, se debe consultar a los docentes antes y brindar una justificación de su uso.

6. Entrega

Se debe realizar una entrega digital a través del campus de la materia, compuesta por un único archivo comprimido con todos los contenidos del trabajo.

El nombre de dicho archivo debe cumplir el siguiente formato:

```
tp2_legajoA_legajoB.zip
```

donde `legajoA` y `legajoB` son los legajos de ambos integrantes del grupo (ordenados de menor a mayor) y `.zip` es la extensión del archivo comprimido (también puede ser `.zip`, `.tar`, `.gz`, o `.tar.gz`, **pero no puede ser .rar**).

A su vez, el archivo comprimido debe contener los siguientes elementos:

1. El código fuente en (posiblemente) varios archivos de python (.py).
2. La documentación del desarrollo **en formato pdf**, que contiene los siguientes items:
    1. Carátula del trabajo práctico, nombre y apellido del alumno y dirección de correo electrónico.
    2. Objetivos del trabajo.
    3. Diseño del programa. Esto incluye una explicación sobre cómo funcionan las distintas partes del programa y las alternativas de diseño que fueron consideradas.
    4. Reseña sobre problemas encontrados y soluciones.
    5. Indicaciones para ejecutar correctamente el programa y las pruebas.
    6. Resultados de ejecuciones, incluyendo capturas de pantalla (como imagen o texto), bajo condiciones normales e inesperadas de entrada.
    7. Bibliografía.

Nota: el informe debe estar redactado en *correcto* castellano.