

AHORCADO

Fundamentos de la programación I



UNIVERSIDAD ALFONSO X EL SABIO

Docente: Beatriz Magan Pinto

Integrantes:

- Aura Mora
- Nicolas Timoneda

Madrid 2022

ÍNDICE

PARTE 1	3
Esquema del sistema.	3
Diagrama de flujo.....	3
 PARTE 2	4
Introducción y alcance: explicación del proyecto y su alcance.....	4
Motivación y objetivos: explicación de por qué han decidido realizar este proyecto y los objetivos que persiguen con él.	5
Estructura de datos empleados en el código.....	5
Estructuras del flujo de control empleados en el código.....	5
Control de excepciones empleadas en el código.	7
Casos particulares y tasas de éxito.	8
Dificultades encontradas.	9
Valores que nos ha aportado el trabajo:	9
Conclusiones.	10
Anexo al repositorio	10

Esquema del sistema.

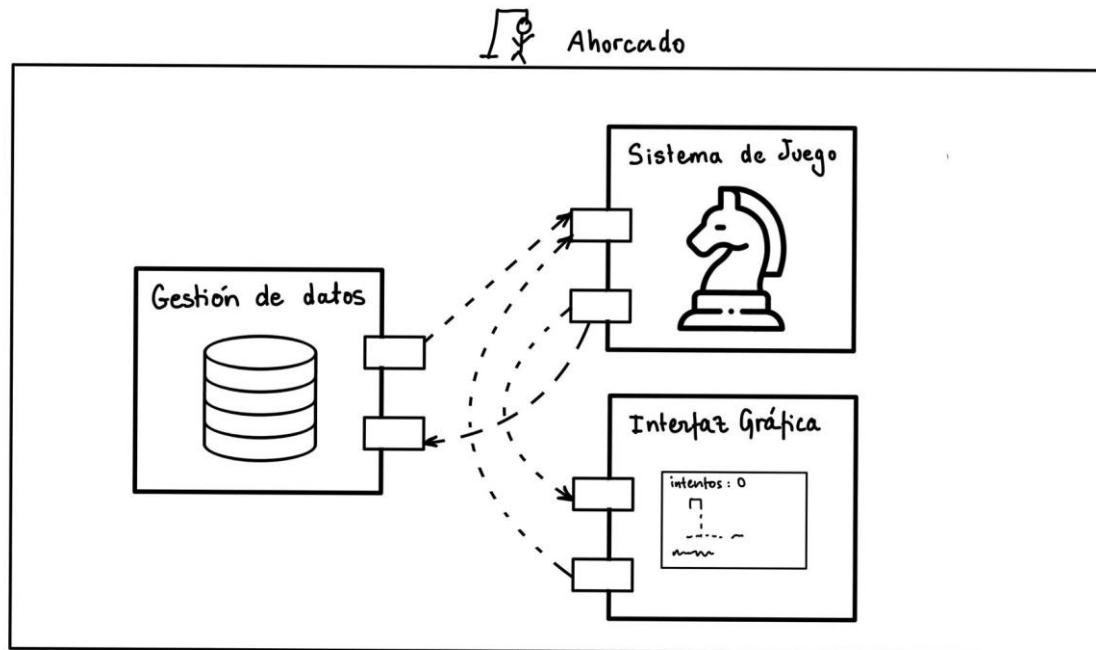
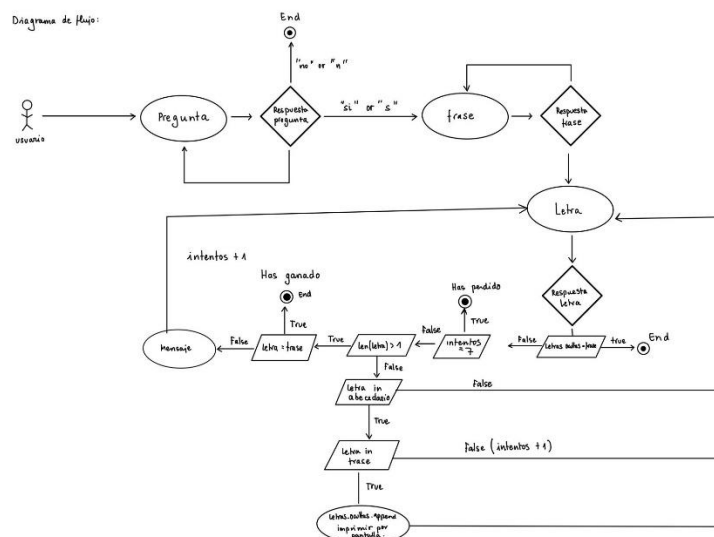


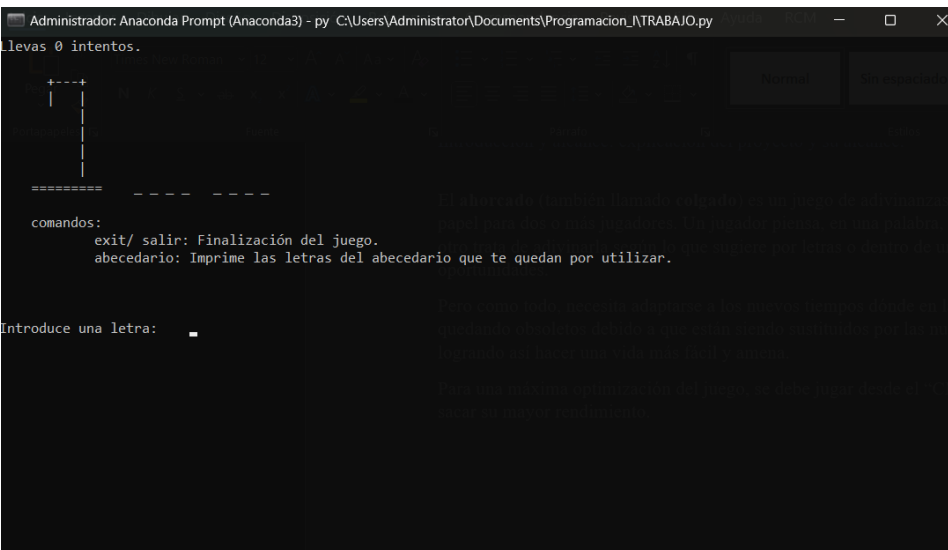
Diagrama de flujo



El esquema del sistema y diagrama de flujo han sido creado a mano mediante una aplicación llamada Good Notes.

Introducción y alcance: explicación del proyecto y su alcance.

Pero como todo, necesita adaptarse a los nuevos tiempos dónde en lápiz y papel se están quedando obsoletos debido a que están siendo sustituidos por las nuevas tecnologías, logrando así hacer una vida más fácil y amena.



Administrator: Anaconda Prompt (Anaconda3) - py C:\Users\Administrador\Documents\Programacion_I\TRABAJO.py

```

+---+
|   |
|   |
|   |
|   |
|   |
|   |
+---+
=====
comandos:
exit/ salir: Finalización del juego.
abecedario: Imprime las letras del abecedario que te quedan por utilizar.

Introduce una letra:

```

- La frase o palabra solo puede estar compuesta por las letras del abecedario.
- El jugador solamente podrá introducir una letra por turno, o en caso de saber la respuesta, podrá introducirla.
- Si el jugador introduce una letra que no está en la palabra o frase, se considera un fallo y, en consecuencia, se dibuja una parte de un muñeco colgado de una cuerda.
- El jugador tiene un número limitado de intentos ('has perdido' equivale a intentos = 7) ya que, si se termina de dibujar el muñeco, y no ha adivinado la palabra, habrá perdido.
- Si el jugador acierta la palabra o frase antes de que se termine de dibujar el muñeco, habrá ganado.

El alcance del proyecto es llegar recrear, mediante un código programado a través de Python, un juego tan popularmente conocido como es “El ahorcado”, y que cualquier persona sea capaz de jugarlo.

Motivación y objetivos: explicación de por qué han decidido realizar este proyecto y los objetivos que persiguen con él.

Nosotros hemos decidido realizar este proyecto porque nos resultaba bastante impresionante que un juego tan típico y conocido por todos, como es “El ahorcado”, se puede replicar de forma casi exacta a través de Python, basándonos en los contenidos teóricos pertenecientes a las unidades didácticas 2 y 3.

Con el objetivo principal que tenemos con este proyecto es poner en práctica los principios de programación y conocimientos teóricos que hemos obtenido, de manera que les podamos dar un uso práctico y útil.

Estructura de datos empleados en el código.

Las estructuras de datos sirven para almacenar información de una forma determinada y según los objetivos que persiga el programador. Todas las estructuras de datos empleadas en el código se encuentran en las unidades de teoría de la asignatura. Las que nosotros hemos empleado son variables, input, diccionarios y arrays:

VARIABLES:

Una palabra a la que se le asigna un valor, este puede ser de cualquier tipo float, int, string...

INPUT:

Recibe información del usuario mediante teclado, y este lo introduce.

LISTAS:

Lista = ['elemento1', 'elemento2', 'elemento3'].

DICCIONARIOS:

Es una colección de elementos donde cada uno tiene una llave **key** y un valor **value**. En nuestro programa lo hemos utilizado para a cada dibujo del muñequito del ahorcado atribuirle un valor.

Ejemplo: nuestro diccionario se llama ahorcado y le vamos dando valores al dibujo del muñeco.

Key : 0

Value: aparece el ahorcado.



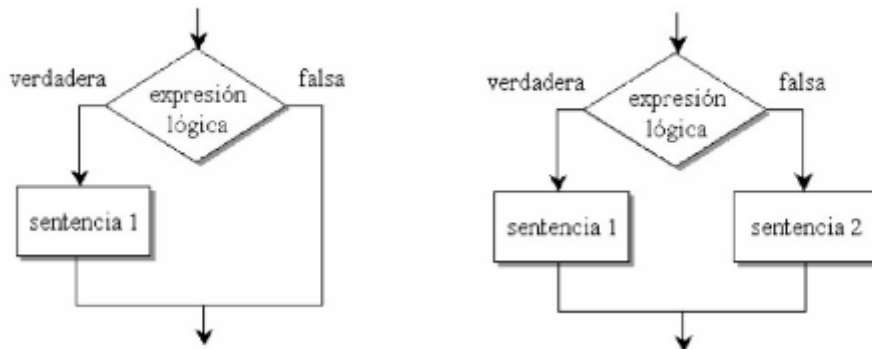
Estructuras del flujo de control empleados en el código.

Nuestro código se basa en tres estructuras que son las de if-elif-else, while y for por las que vamos controlando el flujo de nuestro código.

SENTENCIA IF-ELSE:

- Es una bifurcación o sentencia condicional de una, dos o más ramas. La sentencia de control evalúa la condición lógica o booleana. Si esta condición es cierta entonces se ejecuta la sentencia o sentencias (1) que se encuentra a continuación. En caso contrario, se ejecuta la sentencia (2) que sigue a else (si esta existe). La sentencia puede constar opcionalmente de una o dos ramas con sus correspondientes sentencias.

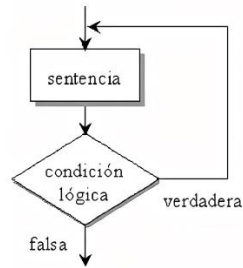
Flujograma de la sentencia if. Con una rama (a la izquierda) y con dos ramas (a la derecha).



SENTENCIA WHILE:

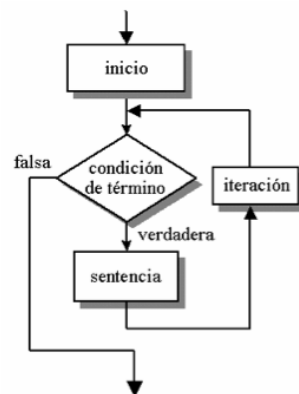
- Es un bucle o sentencia repetitiva con una condición al principio. Se ejecuta una sentencia mientras sea cierta una condición. La sentencia puede que no se ejecute ni una sola vez.
- Explicación del bucle While: lo primero que hace la intérprete java es irse al método “main” empieza la ejecución del programa hacia abajo. Cuando se encuentra con un bucle while, si la condición que estamos evaluando es cierta, la intérprete java ejecutará las líneas que hay dentro de while y lo repetirá tantas veces como indique la condición, mientras la condición sea verdad seguirá ejecutando las líneas de código. Cuando la condición deje de ser verdad dejará de ejecutar las líneas que hay Enel interior y continuará desde la llave de cierre hasta abajo.
- Puede ocurrir que desde la primera vez sea falsa la condición, entonces no ejecutará ni siquiera una vez las líneas que hay en el interior.
- Puede ocurrir que nunca deje de ser cierta la condición, en este caso entraría en lo que se conoce un bucle infinito, el intérprete de phython en while y ejecutará una y otra vez las líneas de códigos dentro del while.

Flujograma de la sentencia while:



SENTENCIA FOR:

Por otro lado, tanto la sentencia de inicio como la de iteración pueden componerse de varias sentencias separadas por comas.



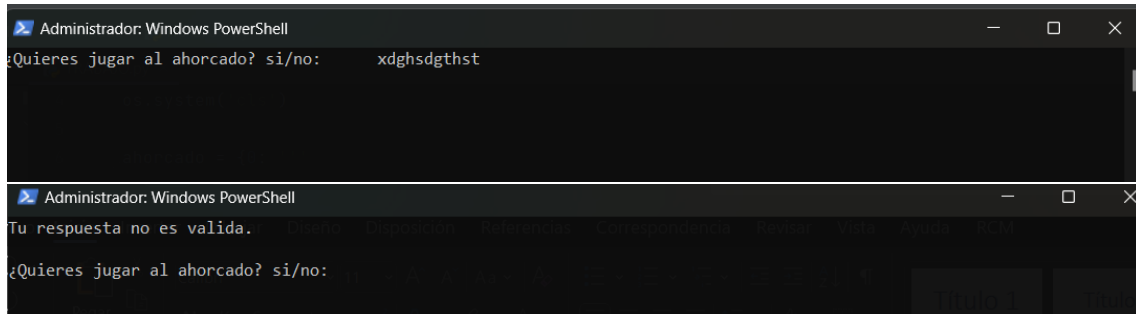
Control de excepciones empleadas en el código.

No tenemos ninguna expresión como tal, lo que tenemos es a través de “if” vamos controlando las posibles entradas de las variables no válidas de la variable y de esta forma se va filtrando la información hasta que llegue a ser una válida.

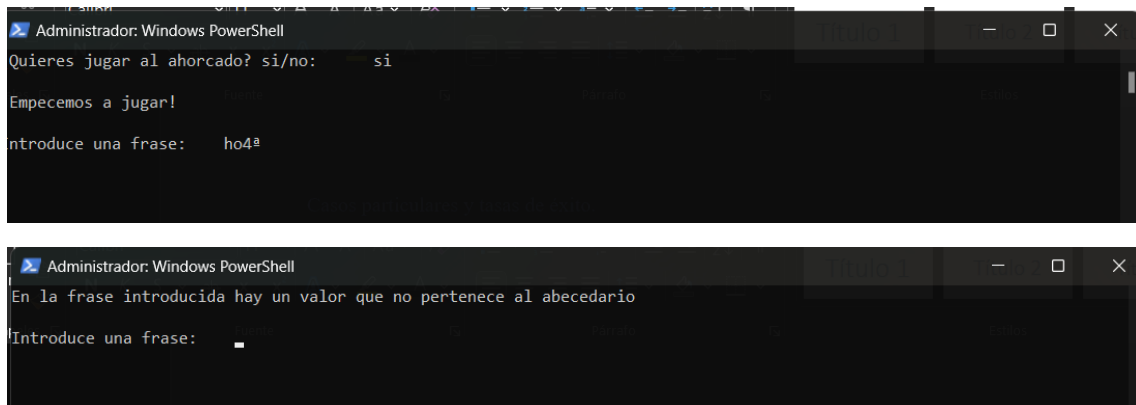
Casos particulares y tasas de éxito.

En el caso de introducir por teclado valores erróneos:

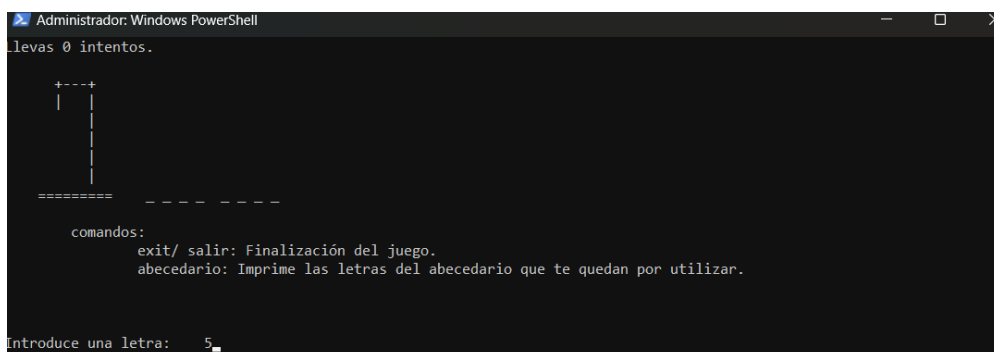
1. Te pregunta si quieres jugar, y responden algo distinto a si o no:



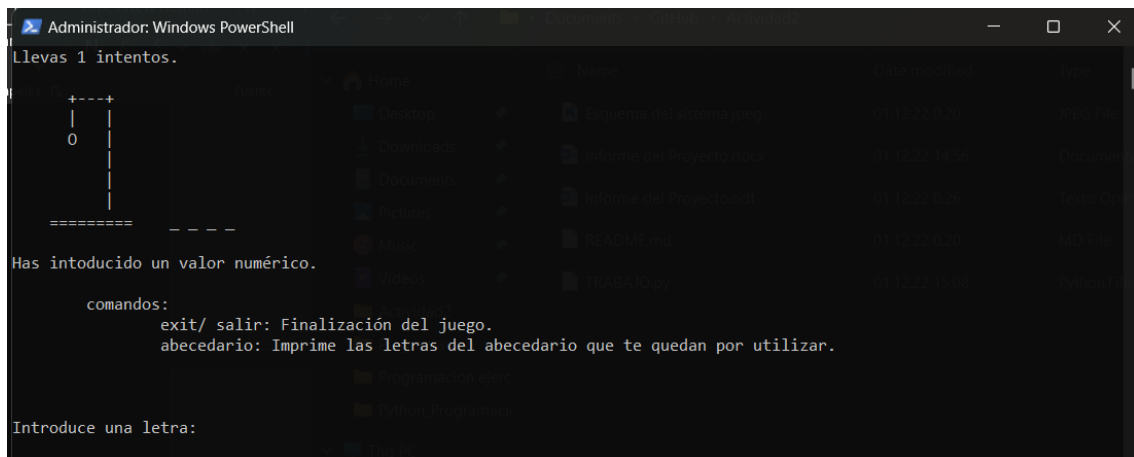
2. la frase contiene caracteres que no son pertenecientes al abecedario



3. Una vez se inicialice el juego ingresas valores que no son una letra o la frase a adivinar.



Por ejemplo, en este caso se ingresa el número 5, te sale un mensaje y se te suma 1 al número de intentos ya que te has equivocado como aparece en la siguiente imagen:



```
Administrador: Windows PowerShell
Llevas 1 intentos.

+---+
|   |
0
|
+---+

=====

Has introducido un valor numérico.

comandos:
  exit/ salir: Finalización del juego.
  abecedario: Imprime las letras del abecedario que te quedan por utilizar.

Introduce una letra:
```

Al igual si metes un carácter o un valor vacío.

Obteniendo así una tasa de éxito del 100%.

Dificultades encontradas.

No hay dificultades a objetar ya que, si ha habido alguna, se ha ido resolviendo durante el proceso de creación del código.

Por ejemplo: Para lograr ver los diferentes fallos del programa la mejor forma de encontrarlos es que una persona independiente juegue y nos dé su opinión y de esta manera lograr que nuestro juego este a su pleno rendimiento.

De entre los fallos encontramos nuestro primer “Bug”, consistía en si introduces una cadena de caracteres erróneo no te sumaba un intento.

Y a base de prueba y error que es como podemos definir este trabajo, hemos logrado replicar el juego del ahorcado.

Valores que nos ha aportado el trabajo:

Nico: Personalmente, este trabajo me ha ayudado a asentar las bases de los conocimientos que he adquirido en estas unidades, además de darles un valor útil y poder ver que tiene un sentido práctico. Además, hemos desarrollado el trabajo en equipo, que es algo que me parece clave hoy en día y ha sido muy divertido realizar el trabajo juntos. También quería destacar el valor de superación ya que, si en algún momento surgía alguna dificultad, hemos luchado para arreglar el error o sacar el proyecto adelante, y verlo terminado ahora es una gran satisfacción.

Aura: Este trabajo ha supuesto un gran reto para mí porque a la mínima en el momento en el que no me funciona algo me desespero. Y me ha ayudado mediante los commits a la resolución de problemas de una manera rápida y eficaz. Adquiriendo una gran capacidad de resolución de problemas. Pero lo que me ha enseñado es que constancia y dedicación que hemos puesto tanto Nico como yo ha hecho que podamos recrear un juego tan conocido como es el ahorcado, y espero poder decir sin fallos ni bugs.

Conclusiones.

Como conclusión, podríamos llegar a decir que hemos cumplido nuestros objetivos iniciales, creando un juego muy similar al que tomábamos de referencia, y que disponga de una interfaz fácil y sencilla que pueda ser fácilmente ejecutada por cualquier persona, a pesar de que no se tenga idea sobre programar.

Además, ambos nos hemos dado cuenta de que con un poco de práctica y siguiendo todas las indicaciones y directrices que se nos han dado, se pueden crear grandes programas a partir de la base tan simple que se obtiene con la teoría.

Anexo al repositorio

Por diferentes motivos tenemos 2 uno publico y uno privado

PUBLICO

<https://github.com/aurittamora/Actividad2.git>

PRIVADO

<https://github.com/ntimomar/TRABAJO.git>