Práctica 4: Programación modular usando funciones

1. Objetivos

- Comprender los conceptos básicos sobre la codificación de funciones en Python.
- Desarrollar programas que involucren programación modular mediante el uso de funciones.
- Aprender a manipular archivos para los datos de entrada y salida de un programa.

2. Marco Teórico

Las funciones, presentes en la gran mayoría de lenguajes de programación, son la manera como un lenguaje le permite al programador crear sub-programas. Con las funciones, el programador puede encapsular un segmento de código que en la mayoría de los casos va a ser utilizado múltiples veces, o simplemente, le permite organizar un programa de manera modular. Al crear una función se debe dejar claro, por medio de un docstring, la especificación que describe las condiciones de los argumentos (datos de entrada) y las garantías del trabajo que hace la función. Las funciones deben ser probadas, una por una, a medida que se van integrando al programa principal.

3. Tareas a Realizar

Esta práctica consiste en la implementación de un programa para el juego clásico de **ahorcado**. El programa debe ser desarrollado de manera modular mediante el uso de funciones.

3.1. Resumen del juego

El ahorcado es un popular juego cuyo objetivo consiste en adivinar una palabra o frase conociendo su longitud. Al comenzar se dibuja una palabra o frase reemplazando cada letra por un guion y dejando los espacios que separan las palabras cuando es una frase. Para adivinar la palabra o frase, el participante tendrá un número máximo de intentos. En cada intento se le solicitará una letra de tal manera que si ésta aparece dentro de la frase, los lugares ocultos (guion) correspondientes a la letra serán reemplazados por la letra ingresada. Por otro lado, si la letra no pertenece a la frase o palabra, el usuario recibirá una penalización disminuyendo el número de intentos disponibles. El juego culmina cuando el usuario adivina la palabra o cuando gasta el número máximo de intentos sin poder adivinar la palabra. En la siguiente página: www.hangman.no puede experimentar con el juego (en inglés) para que comprenda su funcionamiento.

3.2. Funcionamiento del programa

A continuación se dan los requerimientos que deberá cumplir el juego a implementar:

- Cuando el programa inicia por primera vez deberá tener un anuncio de bienvenida con el nombre del juego de manera semi-gráfica.
- Una vez se inicie el juego, éste deberá permitir seleccionar el modo de juego. Para esto habrá 2 posibilidades: selección aleatoria de una lista de palabras o que otro usuario permita ingresar la palabra a descubrir.
- Después de que la frase secreta sea seleccionada o ingresada, el programa deberá mostrar en pantalla el número apropiado de rayas y espacios que representan la frase.
- El juego deberá solicitar entonces al usuario una letra por ronda con el fin de ir descubriendo la palabra secreta. Cada vez que el usuario acierte en la letra elegida, el programa deberá reemplazar las rayas de la salida secreta por dicha letra. Por otro lado, si el usuario falla, el programa deberá informar el intento incorrecto y disminuir los intentos restantes.
- En cada turno, el programa deberá desplegar la palabra o frase parcialmente adivinada por el usuario además de las letras que el usuario aún no ha empleado.
- El juego culmina cuando el usuario adivina la palabra o cuando después del número máximo de intentos no logra adivinarla (en este caso se revelará la palabra oculta). Cuando el juego termina, deberá preguntar al usuario si quiere empezar un nuevo juego o si prefiere salir del programa.

Reglas adicionales del juego

- Asuma que el usuario solo ingresa un carácter a la vez.
- El número máximo de intentos permitidos es de 8. El programa deberá ir llevando la cuenta del número de intentos restantes después de cada ronda.
- El número de intentos disponibles solo disminuye cuando la letra elegida por el usuario no pertenece a la palabra.
- Si el usuario introduce una misma letra dos veces, el número de intentos disponibles no disminuye, simplemente se informa al usuario y se le solicita una nueva letra.
- El programa deberá llevar un conteo de las palabras o frases intentadas por el usuario y el total de frases o palabras adivinadas.

Diagrama de flujo principal

Este problema es un poco más complejo que los que se han venido desarrollando hasta el momento, por lo tanto lo ideal antes de codificarlo es planificar su estructura de funcionamiento general. Una herramienta bastante útil para ello consiste en el diagrama de flujo, que en nuestro caso, serán diagramas de flujos simplificados y abstractos pues el detalle debe ser

desarrollado por el estudiante. Iniciaremos describiendo el diagrama de flujo general del juego, y luego, una vez definido este, se definirán todos aquellos detalles internos necesarios para la implementación de cada proceso del diagrama de flujo. Aunque el desarrollo del diagrama de flujo no suele ser algo obligatorio cuando se concibe un programa, recomendamos que se tome su tiempo para esto y no se apresure con iniciar la codificación. A continuación se muestra el diagrama de flujo general del programa:

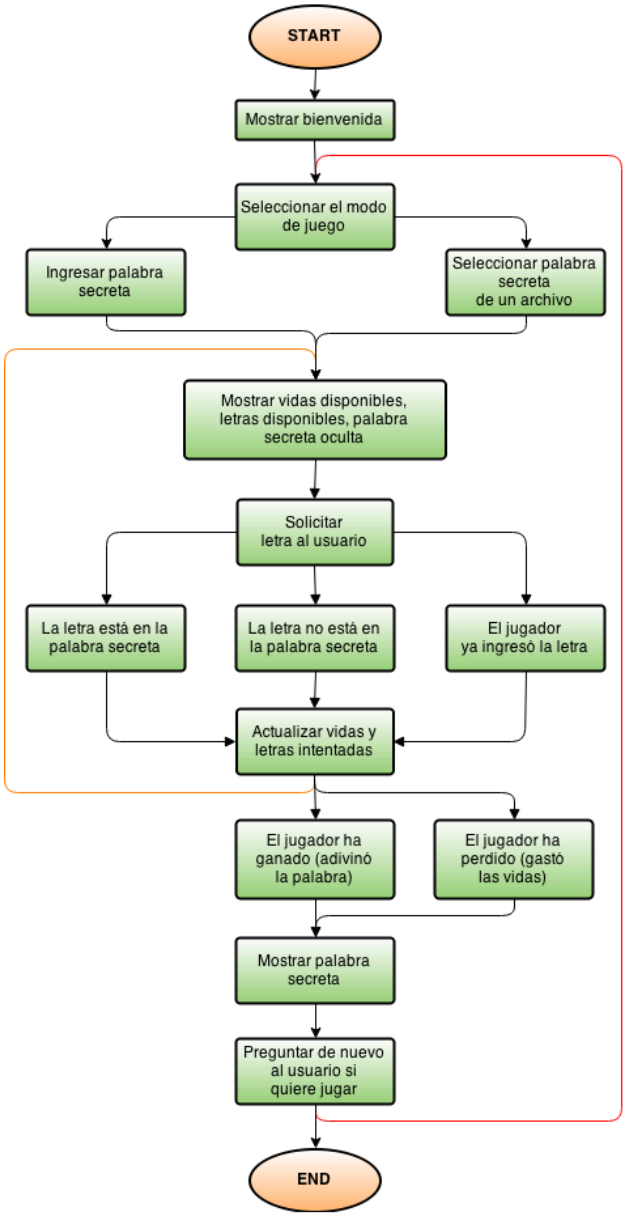

Figura 1. Diagrama de flujo principal del juego.

La figura anterior muestra el funcionamiento del programa de modo resumido. El propósito de este laboratorio es que usted implemente todos estos módulos haciendo uso de funciones.

3.3. Desarrollo del programa

Para el desarrollo de este laboratorio se suministran los siguientes archivos los cuales deberán ser descargados de la página del curso:

- **`ahorcado.py`**: Archivo donde va la implementación de todas las funciones que será necesario completar de manera guiada a lo largo del laboratorio para el desarrollo del juego del ahorcado.
- **`main.py`**: Archivo en el que se implementará el programa principal haciendo uso de las funciones implementadas en el archivo **`ahorcado.py`**.
- **`superHeroes.txt`**: Archivo con palabras y frases nombres de súper héroes y que serán empleadas en el juego como palabras ocultas.
- **`intro.txt`**: Archivo que contiene el nombre del juego de manera amigable.

Anuncio de bienvenida

El anuncio a desplegar se encuentra almacenado en el archivo **`intro.txt`**. La función llamada **`printIntro()`** deberá desarrollarse (en el archivo **`ahorcado.py`**) para permitir desplegar el contenido de cualquier archivo de texto cuyo nombre es tomado como argumento de entrada. Pruebe la función con el archivo intro.txt suministrado.

Una vez la función **`printIntro()`** esté completa, el siguiente paso consistirá en probar que funcione bien. En los comentarios de la función en la plantilla (en la sección de ejemplos de uso), se muestra la salida tentativa una vez se hace el llamado a la función. Si todo está bien, la salida deberá ser algo como se muestra en la siguiente Figura 2.

Figura 2. Resultado de la invocación de la función printIntro().

Comentarios y tips:

- Recuerde el tema de manipulación de archivos. http://docs.python.org.ar/tutorial/3/inputoutput.html#leyendo-y-escribiendo-archivos
- Por si desea explorar más a fondo, el generador ascii empleado en el caso para generar el mensaje de ahorcado semi-gráfico está en:
- Tenga en cuenta que el archivo intro.txt le será entregado y no será necesario que lo modifique o cree nuevos archivos para desplegar la entrada semi-gráfica. Sin embargo, si desea experimentar, consulte la el sitio web http://patorjk.com/software/taag/#p=display&f=Doom&t=Ahorcado

Selección del modo de juego

Recordemos que el programa tendrá dos formas de juego. Una en la cual el usuario podrá elegir entre ingresar una palabra secreta o seleccionar una palabra secreta de un archivo.

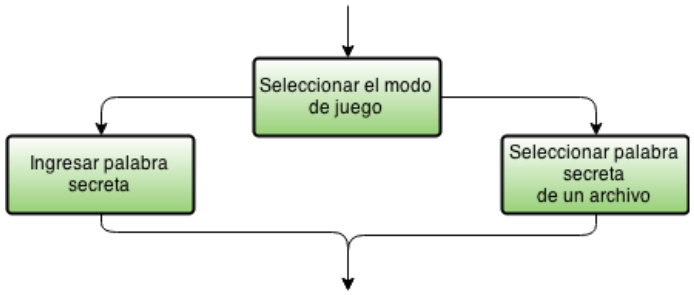

Figura 3. Parte asociada a la selección del modo de juego.

Ingreso de la palabra secreta

Este modo de juego está ideado para 2 personas. Aquí uno de los participantes ingresará la palabra o frase desconocida y su oponente, sin conocer lo ingresado, será quien intentará adivinar la palabra oculta.

Para implementar esta parte complete la función **`inputSecret()`** (ver el archivo **`ahorcado.py`**) la cual se encarga de solicitar al usuario una palabra o frase cualquiera. Esta función no tendrá argumentos de entrada y deberá retornar la palabra secreta.

Selección de la palabra secreta de un archivo de texto

La otra forma de juego es mediante la selección de una palabra secreta de un archivo de texto. En el archivo **superHeroes.txt** las palabras secretas están separadas por comas, tenga esto en cuenta al momento de implementar el programa. Abra manualmente el archivo para que observe cómo están almacenadas las palabras.

Las siguientes funciones deben ser implementadas por usted para llevar a cabo esta parte del programa:

- **`loadWords()`**: Función que solicita el nombre de un archivo y retorna su contenido en una cadena de caracteres. Los detalles para la implementación de la función se muestran en sus comentarios.
- **`countWords()`**: Esta función se encargará de contar el número de palabras o frases secretas almacenadas en una cadena de caracteres. Para distinguir una palabra o frase de otra dentro de la cadena de caracteres es necesario saber el carácter separador de las palabras.
- **`pickWord()`**: Esta función retornará una palabra o frase al azar de una secuencia de palabras almacenadas en una cadena de caracteres. Dentro de la secuencia, cada una de las palabras va separada por un mismo carácter.

En ejemplos de uso se muestran las respectivas invocaciones y resultados esperados cuando se invoca la función.

Comentarios y tips:

- Dentro del cuerpo de **`pickWord()`** es necesario utilizar la función **`countWords()`**, la cual fue anteriormente implementada.
- Para este ejercicio necesitará generar un número aleatorio entre 0 y la cantidad de palabras que hay en el archivo de palabras. Para ello use la función random.randint(a,b) la cual retorna un entero aleatorio N entre a y b incluidos estos (a <= N <= b).
- Para poder hacer uso de la función anterior no olvidar importar la librería random dentro de la función.

Actualización de la palabra secreta

La Figura 4 muestra el diagrama de flujo de la parte del programa que se encarga de actualizar el estado del juego a medida que éste avanza, es decir, el ciclo principal interno. Además de imprimir el estado del juego, se debe capturar cada intento de adivinar una letra por parte del usuario. Intentar adivinar la palabra es una tarea repetitiva por lo que cada vez que se lleva a cabo el ingreso de una letra se deberán hacer cosas como: imprimir la palabra secreta oculta,

actualizar y mostrar el contador de intentos que quedan y actualizar y mostrar las letras disponibles.

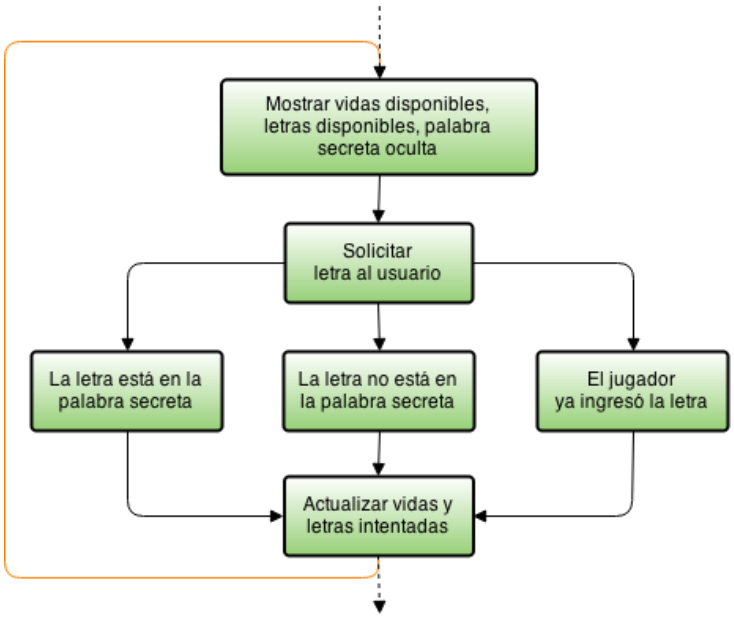

Figura 4. Ciclo principal interno del juego.

A continuación vamos a proceder a implementar cada una de estas partes.

Haciendo énfasis en imprimir la palabra oculta, cada vez que se ingresa una letra, la palabra o frase oculta deberá actualizarse. La siguiente tabla muestra el proceso de actualización suponiendo que la palabra desconocida es **gato**:

El objetivo ahora es implementar una función que permita realizar lo anterior, es decir, imprimir la palabra a adivinar con guion bajo en aquellos lugares donde no se ha adivinado aún la letra correspondiente.

Impresión en pantalla	Letras probadas
_ _ _ _	x
_ a _ _	a
_ a _ _	i
g a _ _	g
g a _ _	e
g a t _	t
g a t _	u
g a t o	o

`obtenerParteAdivinada()`: Esta función realiza lo descrito anteriormente desplegando la parte de la palabra que se conoce. Esta función, tomará dos parámetros: un string con la palabra secreta y una lista de letras ya intentadas. La función retornará una cadena compuesta por letras y guiones bajos como se muestra en el siguiente ejemplo:

```
>>> palabraSecreta = 'perro'
>>> letrasIntentadas = ['a', 'e', 'i', 'o', 'u', 's', 'p']
>>> print obtenerParteAdivinada(palabraSecreta, letrasIntentadas)
'p e _ _ o'
```

Obtención de las letras disponibles

En cada turno del juego lo que se hace es seleccionar una letra del alfabeto. A medida que avanza el juego la letra seleccionada se va quitando de la lista de letras disponibles. En esta sección, el propósito es la implementación de una función encargada de ésto, la cual se describe a continuación.

`obtenerLetrasDisponibles()`: Esta función toma como entrada una lista con cada una de las letras intentadas y retorna una cadena compuesta por letras (en minúscula) que contienen las letras del alfabeto que aún no han sido usadas.

Comentarios y tips:

- La función **`string.ascii_lowercase`** lista todo el alfabeto en minúsculas. Esta puede serle de utilidad en el desarrollo de la función.

Verificación de letras ingresadas

Cuando el jugador ingresa una letra que ya ha ingresado antes, no habrá penalización en el número de intentos disponibles sino que se solicitará nuevamente al usuario el ingreso de una letra diferente. Pensando en esto, implemente la siguiente función:

`verificarLetraIngresada()`: Esta función toma como entrada una letra y determina si se encuentra dentro de la lista de letras intentadas, retornando True en caso positivo o False en caso negativo.

Verificación de juego terminado

La parte final consistirá en determinar si el juego ha terminado o no. El juego continúa mientras el usuario aún no haya adivinado la palabra y disponga de vidas (intentos disponibles). Si se determina que el juego ha finalizado, se debe mostrar el resultado del juego y preguntar al usuario si desea volver a jugar. Esta parte se muestra en la

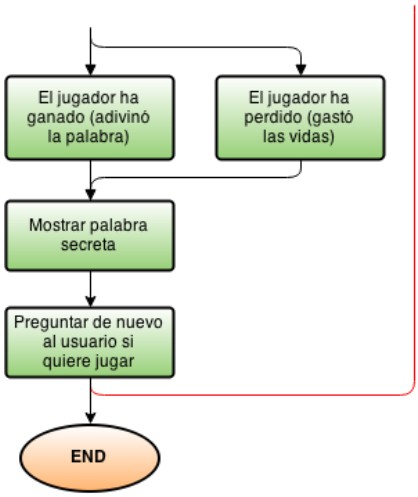

Figura 5. Parte final del juego.

El usuario adivinó la palabra

El jugador adivina la palabra cuando las letras ingresadas permiten descubrir completamente la frase oculta. Para esta parte implemente la siguiente función:

`palabraAdivinada()`: Esta función toma como entrada una palabra o frase y la lista de letras intentadas. La función devolverá True si con las letras de dicha lista se puede formar la frase o False en caso contrario.

El usuario no adivinó la palabra

Esta situación se da cuando el usuario no logra descubrir la palabra después de haber agotado el número máximo de intentos. Esta parte no será implementada como una función propiamente sino que se dejará como parte del programa principal.

3.4. Integración de las funciones para implementar el juego completo

Antes de iniciar esta fase de desarrollo, asegúrese de haber verificado el correcto funcionamiento de cada una de las funciones. Ahora lo que resta es integrarlos en el programa que implementará el juego como tal. Para esto se proporcionará el esqueleto del archivo principal (**`main.py`**). Sin embargo describiremos antes algunas variables importantes que le serán dadas y que deberá emplear en la implementación del juego; estas son:

- **`letrasIntentadas`**: Esta variable es una lista (inicialmente vacía) que almacena cada una de las letras probadas por el usuario en cada turno (sin repeticiones).
- **`numeroIntentos`**: Variable que almacena el número de intentos (vidas) máximo que el usuario tiene para adivinar la palabra o frase oculta. El valor inicial de esta variable es 8.
- **`palabraSecreta`**: Cadena de caracteres que contiene la palabra o frase secreta que deberá ser adivinada por el usuario a lo largo del juego.

- **`otraVez`**: Variable empleada para indicar si se desea jugar de nuevo.
- **`ban`**: Bandera que indica la terminación de una tanda de turnos. Su valor será 1 mientras haya vidas y se pondrá en 0 cuando la tanda culmine ya sea porque el usuario ganó o perdió.

Guíese por el código y los comentarios que hay en la plantilla del programa principal y el diagrama de flujo de la Figura 1.

Opcional

Agregue una función que, en cada intento del jugador, le muestre de manera semi-gráfica el muñequito que se va completando para ser ahorcado a medida que se incrementan los intentos fallidos.

A continuación se muestran ejemplos de varias sesiones del programa ante diferentes situaciones. Obsérvelas bien pues le ayudaran a implementar el programa según los lineamientos.

```
** SELECCION DEL MODO DE JUEGO **
1. Ingresar palabra secreta
2. Seleccionar de un archivo

Seleccione una opcion: 1
Ingrese la palabra o frase oculta: pepe
--------------------
Usted tiene 8 disponibles
Letras disponibles: abcdefghijklmnopqrstuvwxyz
Palabra secreta: - - - -
Por Favor ingrese una letra: p
Letra acertada
--------------------
Usted tiene 8 disponibles
Letras disponibles: abcdefghijklmnoqrstuvwxyz
Palabra secreta: p - p -
Por Favor ingrese una letra: e
Letra acertada
--------------------
Usted tiene 8 disponibles
Letras disponibles: abcdfghijklmnoqrstuvwxyz
Palabra secreta: p e p e

Felicitaciones, la palabra secreta es: pepe has ganado un viaje a Cuba
Desea jugar otra vez (y/n): n
```

Figura 6. Caso en el cual el usuario logra adivinar la palabra secreta y no desea continuar.

```
------------------
Usted tiene 1 disponibles
Letras disponibles: dhijkmnopquwxyz
Palabra secreta: l a   a v - s - a
Por Favor ingrese una letra: q
Letra fallada
------------------
Usted tiene 0 disponibles
Letras disponibles: dhijkmnopuwxyz
Palabra secreta: l a   a v - s - a

Lo lamento, has perdido, la palabra secreta era: la avispa. Animo
Desea jugar otra vez (y/n):
```

Figura 7. Caso en el cual el usuario pierde el juego al no adivinar la palabra después de gastar el número máximo de intentos (vidas).

```
Seleccione una opcion: 1
Ingrese la palabra o frase oculta: pepe
------------------
Usted tiene 8 disponibles
Letras disponibles: abcdefghijklmnopqrstuvwxyz
Palabra secreta: - - - -
Por Favor ingrese una letra: p
Letra acertada
------------------
Usted tiene 8 disponibles
Letras disponibles: abcdefghijklmnoqrstuvwxyz
Palabra secreta: p - p -
Por Favor ingrese una letra: p
La letra ya fue ingresada, por favor intente con otra letra
```

Figura 8. Escenario en el cual el usuario introduce una letra que ya ha sido usada.

```
Desea jugar otra vez (y/n): y


** SELECCION DEL MODO DE JUEGO **
1. Ingresar palabra secreta
2. Seleccionar de un archivo

Seleccione una opcion: 2
Cargando una lista de Super heroes desde el archivo superHeroes.txt...
------------------
Usted tiene 8 disponibles
Letras disponibles: abcdefghijklmnopqrstuvwxyz
Palabra secreta: - - - - - -
Por Favor ingrese una letra: a
Letra acertada
------------------
Usted tiene 8 disponibles
Letras disponibles: bcdefghijklmnopqrstuvwxyz
Palabra secreta: - - - - - a
Por Favor ingrese una letra: e
Letra acertada
------------------
Usted tiene 8 disponibles
Letras disponibles: bcdfghijklmnopqrstuvwxyz
Palabra secreta: - e - e - a
Por Favor ingrese una letra: r
Letra fallada
------------------
Usted tiene 7 disponibles
Letras disponibles: bcdfghijklmnopqstuvwxyz
Palabra secreta: - e - e - a
Por Favor ingrese una letra: p
Letra fallada
```

Figura 9. Caso en el cual la palabra secreta fue seleccionada desde un archivo después de que el usuario decidiera intentar de nuevo otra ronda.

4. Evaluación

La evaluación se basará en los códigos enviados y un quiz sobre los temas de la práctica. Además, se tendrá en cuenta una bonificación para quien haga la parte opcional.