

Logica proposicional enfocada a juegos

Nicolás Suquillo^[L00400639], Toapanta Antoni^[L00384245]

Universidad de las Fuerzas Armadas
artoapanta3@espe.edu.ec, fnsuquillo@espe.edu.ec

Abstract.

En este documento podremos observar la aplicación de la lógica proposicional en los juegos tradicionales que jugábamos de niño, entendiendo así que hasta en lo más simple encontramos la lógica proposicional y pudiendo comprender también de manera más didáctica como funciona en la vida cotidiana. Para esto usaremos el lenguaje de Programación Python el cual es fácil de aprender y programar, además que es uno de los lenguajes de alto nivel más usados hoy en día.

1 Introducción

Durante los últimos años Python se ha convertido en lenguaje muy utilizado en el área tecnológica y se puede decir que se ha convertido en un estándar para la investigación y la exploración en el ámbito de las computadoras [2]. El mismo es utilizado más para áreas como las de el desarrollo web y aplicaciones informáticas, es un lenguaje simple por lo que su curva de aprendizaje no es tan compleja, lo que causa que sea más fácil implementar en el desarrollo de proyectos de software.

Por lo tanto, mediante la utilización de Python como nuestro lenguaje de desarrollo principal, se creará un juego que nos permitirá mejorar nuestros conocimientos en la materia de la lógica proposicional. El juego elegido fue “el teléfono descompuesto” que tiene como objetivo enviar un mensaje, tratando de evitar que este se modifique.

La lógica proposicional es de suma importancia en el área de la programación, ya que, al momento de desarrollar, en este caso un juego, la parte fundamental para poder comenzar a desarrollarlo es realizar un análisis del problema a resolver, donde se aplica muchos de estos conectores, los cuales permitirán que se facilite la implementación. También cabe mencionar que para poder resolver el problema se aplicaran los siguientes pasos: 1) Comprender el problema, 2) Formular un modelo, 3) Desarrollar un algoritmo, 4) Escribir el programa, 5) Probar el programa y 6) Evaluar el programa.

2 Desarrollo

Como se menciono anteriormente para comenzar a realizar el programa primero se deben tener en cuenta los 6 pasos. Por lo cual se implementaron para la resolucion de este problema quedando de la siguiente manera.

3 Proyecto N1 ”Nicolas Suquillo”

3.1 Análisis

Mediante este programa se busca poder comprender de una mejor manera el uso de la lógica proposicional, mediante el desarrollo del juego del teléfono descompuesto, utilizando el lenguaje de programación Python.

Gracias al análisis se podra tener una idea de como se va a desarrollar el programa en base a las necesidades del mismo.

Por lo tanto el análisis se lo realizo de la siguiente manera(vea Figura 1).:

Análisis.	
1.-	Bienvenida al usuario por parte del programa
2.-	Indicaciones al usuario.
3.-	Ingreso de un numero para escoger la frase.
4.-	La frase elegida se muestra por unos segundos en la pantalla.
5.-	Cuando la frase desaparezca se le pedira al usuario que ingrese la frase.
6.-	El usuario ingresara lo que piense que decia la frase.
7.-	Se guardara la frase ingresada por el usuario y se comparara con la frase elegida.
8.-	Si las frases coinciden, el programa mostrara un mensaje de felicitaciones y mostrara la frase con un audio.
9.-	Si las frases no coinciden, el programa mostrara un mensaje indicando que perdio y se mostrara cual era la frase.
10.-	El programa culminara.

Fig. 1. Análisis programa.

1. Lo primero que el programa debe realizar es mostrar un mensaje de bienvenida al Usuario, el cual permitirá que un mejor ambiente visual.
2. Lo segundo que realizara el programa será mostrar las instrucciones de como se juega al usuario para que este se familiarice y entienda el funcionamiento del mismo.
3. Al usuario se le pedirá un número cualquiera que se encontrará relacionado con una frase y al momento del usuario ingresar su numero se le asignará una frase.
4. Cuando el Usuario presiones una tecla para poder seguir con el funcionamiento del mismo, se mostrará inmediatamente la frase que se le asigno al número anteriormente ingresado. La frase solo se mostrará durante unos segundos para que no pueda ser tan entendible para el usuario.
5. Cuando el tiempo de aparición de la frase elegida acabe, el programa le pedirá al usuario que ingrese la frase por teclado, de manera que el usuario deberá visualizarla bien para poder ingresarla.
6. El usuario en caso de no haber visualizado bien la frase, ingresara lo que el crea que es la correcta.
7. La frase elegida y la que se ingresó por el usuario se compararan dando como resultado si se pudo llegar a la misma o no se lo logro.
8. Después de realizar la comparación el programa mostrara un mensaje de felicitación si el usuario llego a visualizar bien la frase.
9. En caso de que se ingresó una completamente diferente, se le indicara cual era la frase inicial.
10. Posteriormente de realizar todos estos procesos el programa después de que el usuario presione una tecla se cerrara.

3.2 Formular un modelo

Modelo.		
P: Frase elegida	q: Frase ingresada por el usuario	
P ^ q	P ^ q	P ^ q
V V	V V	V
V F	F	
F V	F	
F F	F	

• Solo en caso de que las dos proposiciones sean verdaderas el jugador gana.

Fig. 2. Modelo.

3.3 Desarrollar un algoritmo

El pseudocódigo del programa quedaría de la siguiente manera:

Algorithm 1 Pseudocódigo teléfono descompuesto

Require: frases.txt: lista de frases las cuales se guardarán en el programa

numero= Se ingresa un numero del 1 al 20

frase= Frase elegida

Se mostrara la frase al Usuario

adivinanza= Se ingresa la frase del usuario

if adivinanza = frase **then**

Se mostrara un mensaje de felicitacion al usuario

else

Se le indicara al usuario que no acerto en la frase

break

end if

Al usuario se le mostrara un audio con la frase

Fin

Se realizó el pseudocódigo en base a lo que se quiere lograr realizar con el programa y mediante el mismo se comenzará a elaborar el código.

3.4 Digrama de flujo

El diagrama de flujo quedaria de la siguiente manera:

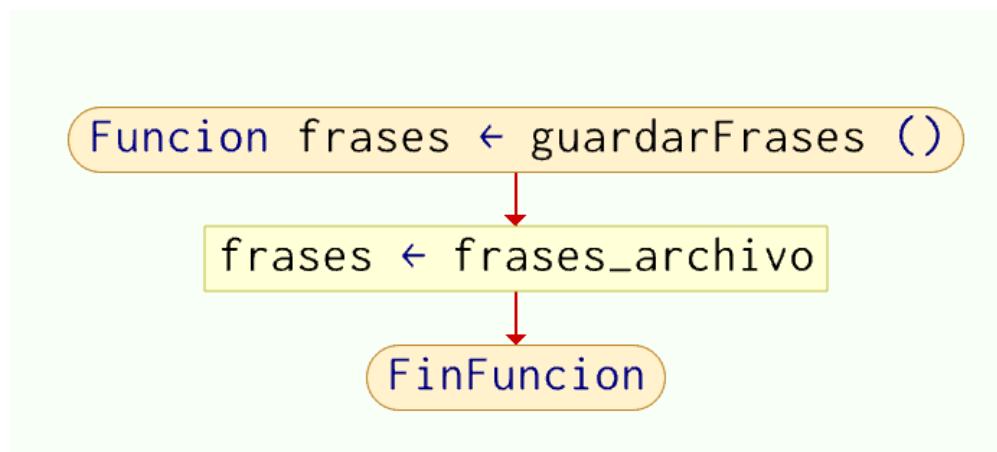


Fig. 3. Funcion para guardar las frases.

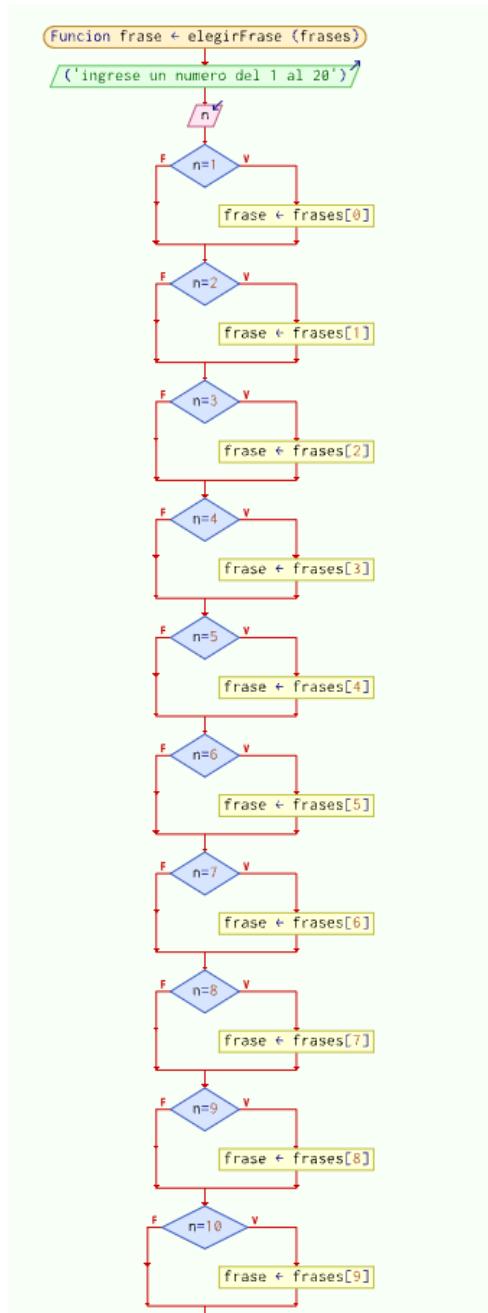


Fig. 4. Funcion para elegir las frases.

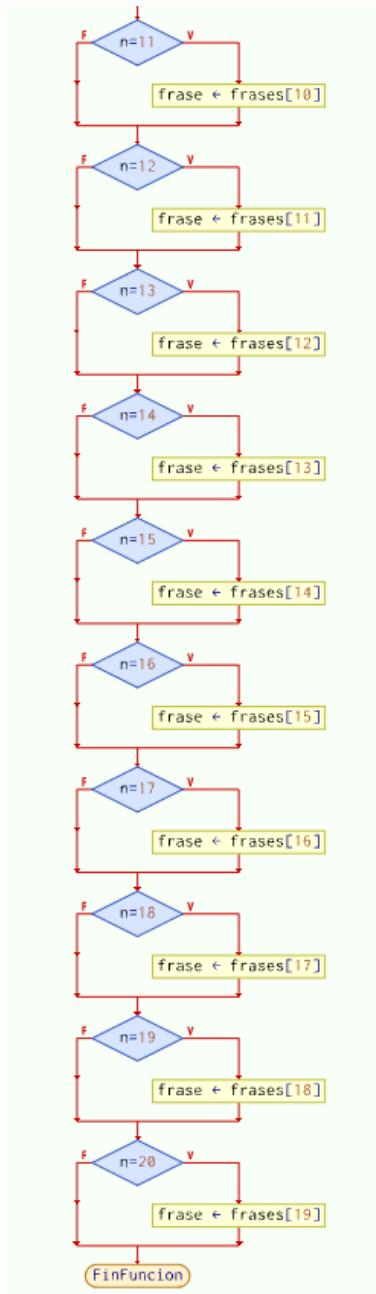
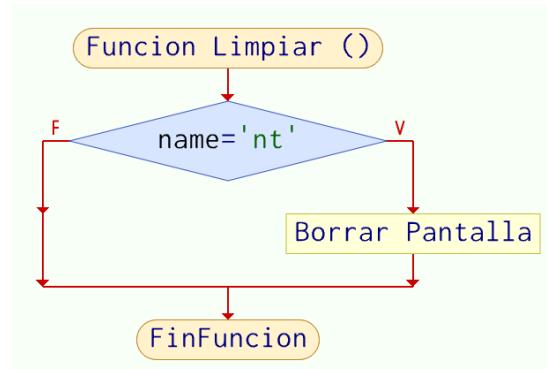
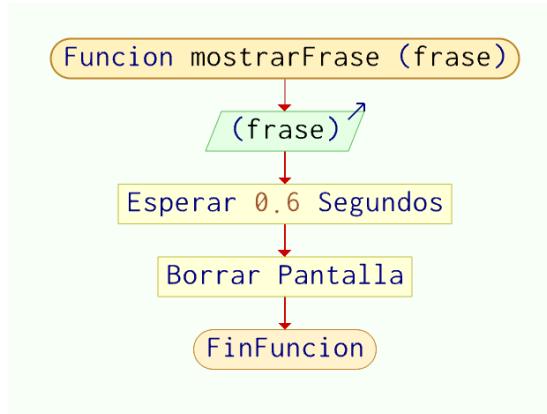
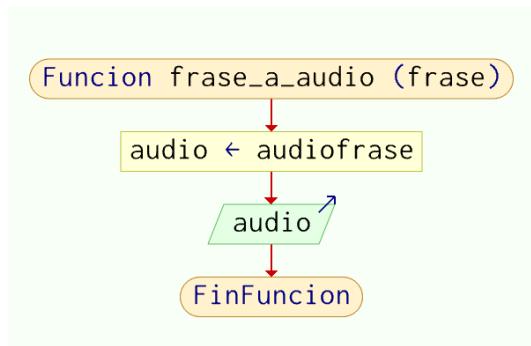


Fig. 5. Funcion para elegir las frases (segunda parte).

**Fig. 6.** Funcion para limpiar la pantalla.**Fig. 7.** Funcion para mostrar y limpiar la frase.**Fig. 8.** Funcion para convertir la frase a audio.

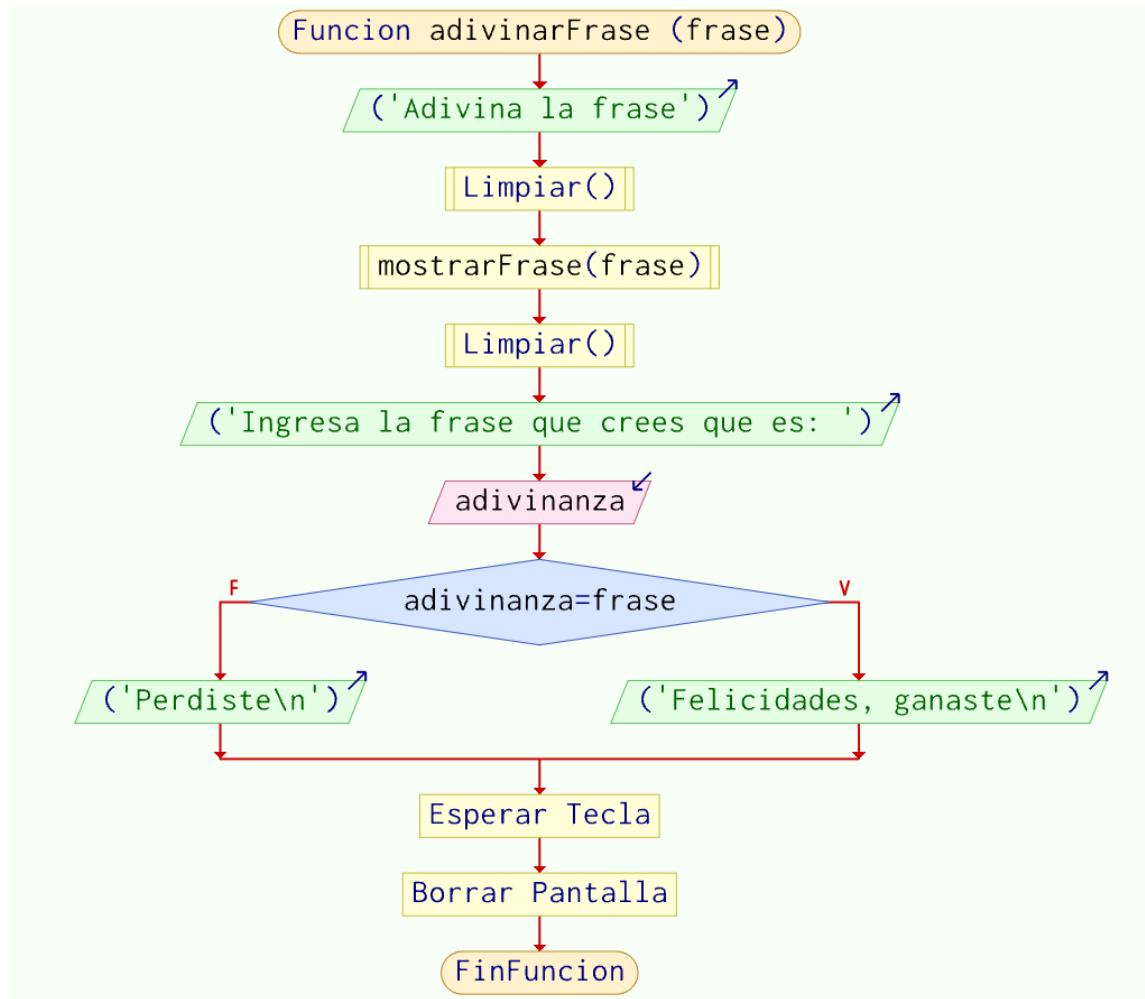


Fig. 9. Funcion para adivinar la frase.

3.5 Escribir el programa

```

1 system("cls")
2 print("\t\t\t-----")
3 print("\t\t\tBienvenido al juego del telefono")
3 print("\t\t\tdescompuesto|")
4 print("\t\t\t-----\n")
5 print("\t\t\tReglas")
5 print("\t\t\t-----")
  
```

```

6 print("\t\t\t|"
      |")
7 print("\t\t\t|El juego consiste en que se te mostrara una
      frase y deberas adivinarla |")
8 print("\t\t\t|La frase se mostrara durante un tiempo y luego
      desaparecera           |")
9 print("\t\t\t|Posteriormente se tendra que ingresar la frase
      que se cree es         |")
10 print("\t\t\t|"
      |")
11 print("\t\t\t-----\n")
12 print("\t\t\t-----")
13 print("\t\t\t|Si aciertas ganas, si no pierdes|")
14 print("\t\t\t-----\n")
15 print("\t\t\tBuena suerte!!!!\n\n\n\n")
16 system("pause")
17 system("cls")

```

Codigo 1.1. Bienvenida del sistema

```

1 #Se leen las frases de un txt y se guardan en una lista
2 def guardarFrases():
3     """
4         Es un proceso que guarda las frases que se encuentran en
5         archivo txt en una lista
6         Parametros:
7             -----
8             No tiene parametros de entrada
9
10        Retorna:
11            -----
12            frases: lista
13            Lista que contiene las frases del archivo txt
14        """
15    #guarda las frases en una lista
16    frases = []
17    #Se abre el archivo
18    archivo = open("frases.txt", "r")
19    #Se lee el archivo
20    with open("frases.txt", "r") as archivo:
21        #Se guarda cada linea en una lista
22        for linea in archivo:
23            #Se guarda la linea en la lista, se le quita el
24            #salto de linea y espacios
25            frases.append(linea.lower().strip())
26    #Se cierra el archivo
27    archivo.close()
28    #Se regresa la lista

```

```
27     return frases
```

Codigo 1.2. Funcion para guardar las frases

```
1 #El usuario ingresa un numero y se le asigna una frase
2 def elegirFrase(frases):
3     """
4         Es un proceso que se encarga de elegir una frase de la
5             lista de frases ,
6             posteriormente de haber pedido al usuario que ingrese un
7             numero
8             Parametros:
9                 -----
10                frases: lista
11                    Lista que contiene las frases del archivo txt
12
13                Retorna:
14                -----
15                frase: str
16                    Frase que se eligio de la lista de frases
17
18    #Se le pide al usuario que ingrese un numero
19    frase = ""
20    #Se valida que el numero este en el rango de la lista
21    while frase == "":
22        #Se le pide al usuario que ingrese un numero
23        numero = input("Ingresa un numero del 1 al 20: ")
24        #Se valida que el numero este en el rango de la lista
25        if numero == "1":
26            #Se guarda la frase en la variable
27            frase = frases[0]
28        #Se valida que el numero este en el rango de la lista
29        elif numero == "2":
30            #Se guarda la frase en la variable
31            frase = frases[1]
32        #Se valida que el numero este en el rango de la lista
33        elif numero == "3":
34            #Se guarda la frase en la variable
35            frase = frases[2]
36        #Se valida que el numero este en el rango de la lista
37        elif numero == "4":
38            #Se guarda la frase en la variable
39            frase = frases[3]
40        #Se valida que el numero este en el rango de la lista
41        elif numero == "5":
42            #Se guarda la frase en la variable
43            frase = frases[4]
44        #Se valida que el numero este en el rango de la lista
45        elif numero == "6":
46            #Se guarda la frase en la variable
```

```
45         frase = frases[5]
46     #Se valida que el numero este en el rango de la lista
47     elif numero == "7":
48         #Se guarda la frase en la variable
49         frase = frases[6]
50     #Se valida que el numero este en el rango de la lista
51     elif numero == "8":
52         #Se guarda la frase en la variable
53         frase = frases[7]
54     #Se valida que el numero este en el rango de la lista
55     elif numero == "9":
56         #Se guarda la frase en la variable
57         frase = frases[8]
58     #Se valida que el numero este en el rango de la lista
59     elif numero == "10":
60         #Se guarda la frase en la variable
61         frase = frases[9]
62     #Se valida que el numero este en el rango de la lista
63     elif numero == "11":
64         #Se guarda la frase en la variable
65         frase = frases[10]
66     #Se valida que el numero este en el rango de la lista
67     elif numero == "12":
68         #Se guarda la frase en la variable
69         frase = frases[11]
70     #Se valida que el numero este en el rango de la lista
71     elif numero == "13":
72         #Se guarda la frase en la variable
73         frase = frases[12]
74     #Se valida que el numero este en el rango de la lista
75     elif numero == "14":
76         #Se guarda la frase en la variable
77         frase = frases[13]
78     #Se valida que el numero este en el rango de la lista
79     elif numero == "15":
80         #Se guarda la frase en la variable
81         frase = frases[14]
82     #Se valida que el numero este en el rango de la lista
83     elif numero == "16":
84         #Se guarda la frase en la variable
85         frase = frases[15]
86     #Se valida que el numero este en el rango de la lista
87     elif numero == "17":
88         #Se guarda la frase en la variable
89         frase = frases[16]
90     #Se valida que el numero este en el rango de la lista
91     elif numero == "18":
92         #Se guarda la frase en la variable
93         frase = frases[17]
94     #Se valida que el numero este en el rango de la lista
```

```

95     elif numero == "19":
96         #Se guarda la frase en la variable
97         frase = frases[18]
98     #Se valida que el numero este en el rango de la lista
99     elif numero == "20":
100        #Se guarda la frase en la variable
101        frase = frases[19]
102    #Si el numero no esta en el rango
103    else:
104        #Se le indica al usuario que ingrese un numero
105        valido
106        print("Numero invalido")
107    #Se regresa la frase
108    return frase

```

Codigo 1.3. Funcion para elegir frase

```

1 #Se limpia la pantalla
2 def limpiar():
3     """
4     Es un proceso que se encarga de limpiar la pantalla
5     Parametros:
6     -----
7         No tiene parametros de entrada
8
9     Retorna:
10    -----
11        No tiene parametros de salida
12    """
13
14    # for windows
15    if name == 'nt':
16        #Se limpia la pantalla
17        _ = system('cls')

```

Codigo 1.4. Funcion para limpiar la pantalla

```

1 #Mostrar por un tiempo la frase elegida
2 def mostrarFrase(frase):
3     """
4     Es un proceso que se encarga de mostrar la frase elegida
5     por el usuario durante unos segundos
6     Parametros:
7     -----
8         frase: str
9             Frase que se eligio de la lista de frases
10
11     Retorna:
12     -----
13         No tiene parametros de salida
14     """
15     #Se imprime la frase

```

```

15     print(frase)
16     #Se espera 0.6 segundos
17     sleep(0.6)
18     #Se limpia la pantalla
19     limpiar()

```

Codigo 1.5. Funcion para mostrar la frase

```

1 #Se convierte la frase a audio
2 def frase_a_audio(frase):
3     """
4         Es un proceso que se encarga de convertir la frase
5             elegida por el usuario a un archivo de audio .mp3 y lo
6                 guarda en la carpeta del proyecto
7                     para poder reproducirlo
8                         Parametros:
9                             -----
10                            frase: str
11                                Frase que se eligio de la lista de frases
12
13        Retorna:
14            -----
15                No tiene parametros de salida
16                """
17 #Guarda la frase en un archivo de audio
18     audio = gTTS(frase, lang='es')
19 #Se guarda el audio
20     audio.save("frase.mp3")
21 #Se reproduce el audio
22     system("frase.mp3")

```

Codigo 1.6. Funcion para convertir a audio

```

1 #Se adivina la frase
2 def adivinarFrase(frase):
3     """
4         Es un proceso que se encarga de ver si la frase que
5             ingreso el usuario es igual a la frase que se eligio de
6                 la lista de frases
7                     Parametros:
8                         -----
9                            frase: str
10                               Frase que se eligio de la lista de frases
11
12        Retorna:
13            -----
14                No tiene parametros de salida
15                """
16 #Se imprime adivina la frase
17     print("Adivina la frase")
18 #Se limpia la pantalla

```

```

17     limpiar()
18     #Se muestra la frase
19     mostrarFrase(frase)
20     #Se limpia la pantalla
21     limpiar()
22     #Se le pide al usuario que adivine la frase
23     adivinanza = input("Ingresa la frase que crees que es: ")
24     #Se compara la frase con la adivinanza
25     if adivinanza.lower().strip() == frase:
26         #Se imprime que la adivinanza es correcta
27         print("Felicitaciones, ganaste\n")
28     else:
29         #Se imprime que la adivinanza es incorrecta
30         print("Perdiste\n")
31     #Se pausa hasta que el usuario presione una tecla
32     system("pause")
33     #Se limpia la pantalla
34     limpiar()

```

Codigo 1.7. Funcion para adivinar la frase

```

1 #Se ejecuta el juego
2 def main():
3     """
4     Es un proceso que se encarga de correr el juego
5     Parametros:
6     -----
7         No tiene parametros de entrada
8
9     Retorna:
10    -----
11        No tiene parametros de salida
12    """
13    #Se guarda las frases en la variable
14    frases = guardarFrases()
15    #Se guarda la frase elegida en la variable
16    frase = elegirFrase(frases)
17    #Se envia la frase para que el usuario la adivine
18    adivinarFrase(frase)
19    #Se muestra cual era la frase
20    print("La frase era: ", frase)
21    #Se convierte la frase a audio
22    frase_a_audio(frase)
23    #Se finaliza el programa
24    print("Gracias por jugar\n\n")
25    #Se pausa hasta que el usuario presione una tecla
26    system("pause")

```

Codigo 1.8. Funcion main

3.6 Pruebas de escritorio

Las pruebas de escritorio quedarían de la siguiente manera:

Prueba de Escritorio		
n	Frase	Jugador Frase
1	Tres tristes tigres	Tres triste tigres
2	Cuando Cuento cuentos	Cuento cuentos cuentos
3	Juan tuvo un tubo	Juan tubo
4	A Cuesta le cuesta	A la Cuesta le cuesta
5	Ana tiene una banana	Ana tiene una ban
6	El cielo esta enladrillado	El cielo esta azul
7	Cubre la cebra su cuerpo	Cubre la cebra su cuerpo
8	El pato tiene dos patas	El pato tiene dos patas
9	Come la hierba la cabra	Come la vaca la hierba
10	Paca se llama la vaca	Laca se llama la vaca

Fig. 10. Pruebas de escritorio.

3.7 Casos de prueba

Las pruebas de prueba quedarian de la siguiente manera:

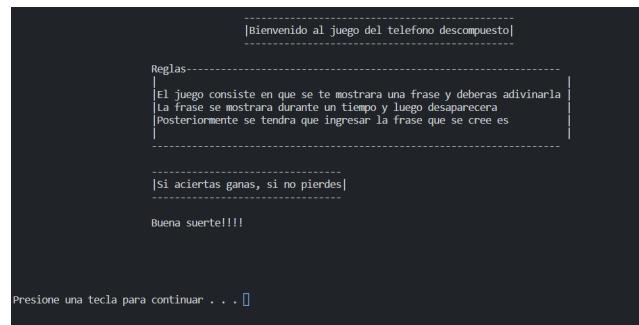


Fig. 11. Caso de prueba 1.

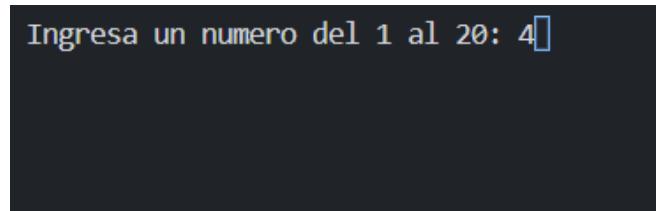


Fig. 12. Caso de prueba 2.

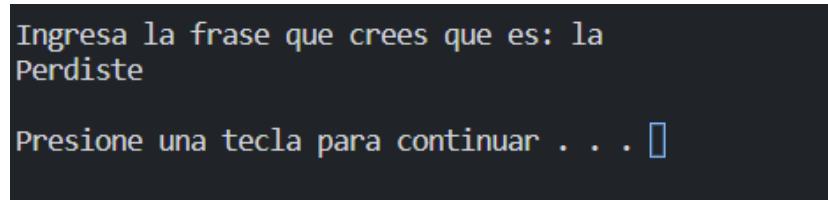


Fig. 13. Caso de prueba 3.

La frase era: juan tuvo un tubo
Gracias por jugar

Presione una tecla para continuar . . . □

Fig. 14. Caso de prueba 4.

Ingresá la frase que crees que es: juan tuvo un tubo
Felicitaciones, ganaste

Presione una tecla para continuar . . . □

Fig. 15. Caso de prueba 5.

Ingresá la frase que crees que es: cuando cuentas cuentos
Perdiste

Presione una tecla para continuar . . . □

Fig. 16. Caso de prueba 6.

La frase era: cuando cuentes cuentos
Gracias por jugar

Presione una tecla para continuar . . .

Fig. 17. Caso de prueba 7.

Ingresá la frase que crees que es: nunca piedra pica piedra
Perdiste

Presione una tecla para continuar . . . □

Fig. 18. Caso de prueba 8.

La frase era: nadie pica piedra como pedro pica piedra
Gracias por jugar

Presione una tecla para continuar . . . □

Fig. 19. Caso de prueba 9.

4 Proyecto N2 ”Toapanta Antoni”

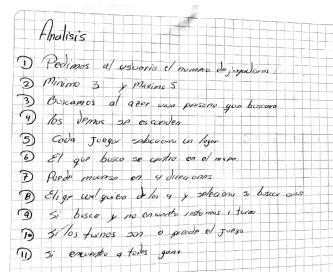
El **juego de las escondidas** es un juego que muchos de nosotros jugábamos de pequeños, acompañados de todos nuestros amigos, consiste de una persona que busca y los demás se esconden. El objetivo principal es encontrar a las demás personas en el menor tiempo posible en este caso se lo hará por turnos, y si encuentra a todos gana el juego caso contrario perdería.

Comprendión del problema: Para pasar de un problema de la vida cotidiana a código tenemos que entender como funciona el juego de las escondidas así logrando encontrar las variables a usar como los procesos que tendrá que seguir para lograr crear algo lo mas parecido al juego, en este caso tendremos que tener máximo 5 jugadores y mínimo 3, de los cuales 1 de ellos se encargara de buscar a los demás y los restantes en esconderse ahora bien el número de intentos será calculado por el número de jugadores por 2, todo ellos se encontrar en un escenario denominado parque que por lo general es donde se juega, las persona que se tienen que esconder tendrán una serie de escondites por elegir y la persona a buscar tendrá que moverse en el escenario del parque en 4 direcciones que son Norte, Sur, Este, Oeste buscando a los demás jugadores.

Analisis:

1. Ingresar el número de jugadores.
2. Seleccionamos al azar un jugador para que sea la persona que busque.
3. Descartamos de nuestros jugadores la persona que va a buscar.
4. Calculamos el número de intentos que tendrá la persona que va a buscar.
5. Los jugadores empezar a seleccionar sus escondites.
6. La persona que busca se centrara en el parque.
7. La persona que busca podrá moverse entre norte, sur, este, oeste.

8. Dependiendo donde se mueva tendrá varios lugares para buscar o regresar al parque central.
9. Si selecciona un lugar para buscar perderá un turno si no se encuentra algún jugador.
10. Si desea regresar al parque central no perderá nada.
11. Si el número de intentos es 0 pierde el juego
12. Si el número de personas por buscar es 0 gana el juego

**Fig. 20.** Analysis

Formular un Modelo El juego tendrá que funcionar primero generando un numero aleatorio entre 0 y el número de jugadores sin contar a la persona que buscara, luego usaremos sentencia **and** para controlar si el número de intentos y personas a buscar sean diferente a 0.

Modelo		
P	Q	$P \wedge Q$
V	V	V
V	F	F
F	V	F
F	F	F

Fig. 21. Modelo del Juego del Ahorcado

Desarrollo del Algoritmo Despues de Analizar todo acerca de cómo funcionara el juego crearemos un algoritmo con el cual poseerá una serie de instrucciones con los debidos procesos el cual demostrara la funcionalidad del juego de manera simple y entendible para cualquier persona.

Algorithm 2 Juego las Escondidas

Require: *listaDeEscondites* :generar lista de escondites

numAleatorio=Generar numero aleatorio

listaDeJugadores=Creamos una lista de jugadores

Ensure:

numAleatorio \leftarrow numeroAleatorio

buscador \leftarrow listaEnlaPosicionAleatorio

Mostrara quien es la persona que busca

for *i* \leftarrow 0 to jugadore **do**

if *jugadoresEnLaPosicion*[*i*] \neq *buscador* **then**

 Selecciona un lugar para esconderse

Colocamos al busacador en el mapa

while *turnos* \neq 0 \wedge *jugadores* \neq 0 **do**

 Buscador se mueve por el mapa pasando por Norte, Sur, Este, Oeste

if *buscador* encuentra a la persona **then**

 Mostar que encontro

else

 no se encontro nadie resto *turnos*

Si turnos es igual a 0 pierde el juego

Si jugadores el igual a 0 gana =0

Digrama de Flujo Creamos el diagrama de flujo de procesos que ira realizando el programa

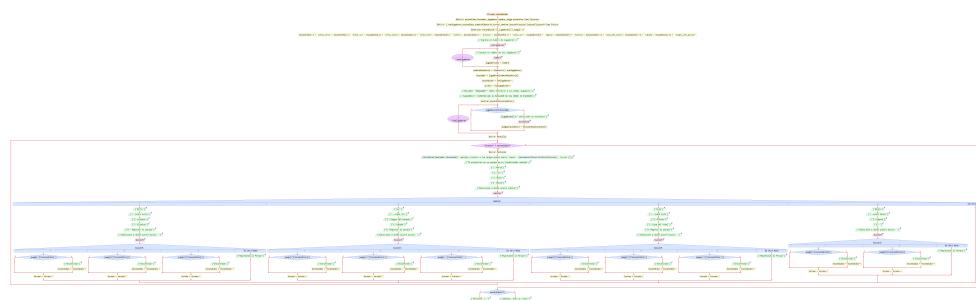


Fig. 22. Flujograma del Juego de las Escondidas

Escribir el Programa: Empezamos con la creación de nuestro programa

Creación de una Función para mostrar todos los escondites que posee el jugador

```
1 def mostarEscondites(escondites):
```

```

2      """
3      Funcion para mostrar escondites
4      Parametros:
5      -----
6      array de escondites
7
8      Retorna
9      -----
10     no retorna nada
11     """
12     #Empieza la iteracion de i en un rango igual a la
13     #longitud de los escondites
14     for i in range(len(escondites)):
15         #mostramos los escondites
16         print("{}: {}".format(i,escondites[i]))

```

Codigo 1.9. Funcion para mostrar los escondites

Creamos los jugadores que poseerá nuestro juego

```

1  def crearJugadores(numJugadores):
2      """
3          Funcion para crear un arreglo de jugadores
4          Parametros:
5          -----
6          numero de jugadores
7
8          Retorna
9          -----
10         retorna un arreglo de jugadores
11         """
12         #creacion de arreglo que se devolvera al finalizar la
13         #funcion
14         jugadoresArreglo=[]
15         #Creacion de un bucle con un rango igual al numero de
16         #jugadores
17         for i in range(0,numJugadores):
18             #agregamos al arreglo los jugadores que participaran
19             en el juego
20             jugadoresArreglo.append(input("Ingrese el nombre de
21             jugador {}: ".format(i+1)))
22             #Retornamos el arreglo
23         return jugadoresArreglo

```

Codigo 1.10. Funcion para crear jugadores

Creamos un número randomico que representara la posición de la persona que buscara

```

1  def numAleatorio(numJugadores):
2      """
3          Funcion crear un numero aleatorio

```

```

4     Parametros:
5     -----
6     numero de jugadores
7
8     Retorna
9     -----
10    retorna un randomico
11    """
12    #Usamos la funcion randomica para crear numeros
13    #aleatorios en un rango de 0 al numero de jugadores
14    num=random.randint(0,numJugadores-1)
15    #retornamos el numero aleatorio
        return num

```

Codigo 1.11. Funcion para crear un randomico

Realizamos una función con la que sabremos si existe o no el jugador

```

1 def seEncuentra(lugar,diccionarioJuego,escondidos):
2 """
3     Funcion para saber si se encuentra el jugador y devolver
4     el numero de personas por encontrar
5     Parametros:
6     -----
7     lugar a buscar, lugare donde se encuebran escondidos y
8     numero de personas por busacar
9
10    Retorna
11    -----
12    no retorna en numero de escondidos de encontrar
13    """
14    #conprobamos si el lugar existe en mi diccionario de
15    #juego
16    if lugar in diccionarioJuego:
17        print("Encontraste a {}".format(diccionarioJuego[
lugar]))
18        #restamos el numero de personas por encontrar
19        escondidos-=1
20    else:
21        #retornamos las personas que faltan encontrar
22        print("No se encuentra nadie")
23    return escondidos

```

Codigo 1.12. Funcion para comprobar si existe un jugador o no

Desarrollo del Programa Principal

```

1 if __name__ == "__main__":
2     print("Numero de jugadores minimo 3 y maximo 5")
3     # creacion de la colección donde se encontrara guardado
4     # el lugar y la persona escondida
5     juego = {}
6     # validamos el numero de jugadores que pueden participar

```

```
6     while True:
7         # ingresamos el numero de jugadores
8         numJugadores = int(input("Ingrese el numero de
9             jugadores: "))
10        # comprobamos si existe el numero de jugadores en el
11        # rango
12        if numJugadores < 3 or numJugadores > 5:
13            # Informacion de jugadores fuera de rango
14            print("Numeros de jugadores superior/inferior
15            recuerde minimo 3 y maximo 5")
16        else:
17            break
18    # para calcular turnos sumamos el numero de jugadores mas
19    # 2
20    turnos = numJugadores+3
21    # Creamos los jugadores
22    jugadores = crearJugadores(numJugadores)
23    # Seleccionamos a la persona que buscara
24    buscador = jugadores.pop(numAleatorio(numJugadores))
25    # el el numero de personas escodidas tomando de la lista
26    # de jugadores sin contar al que busca
27    escondidos = len(jugadores)
28    # funcion para limpiar pantalla
29    system("cls")
30
31    # Los demas Jugadores empiezan a buscar donde esconderse
32
33    # mostramos la informacion del juego
34    print("Buscador: {}, Jugadores: {}".format(buscador,
35        jugadores))
36    # alertar a los demas jugadores
37    print("JUGADORES!!! INTENTEN QUE EL BUSCADOR NO VEA DONDE
38        SE ESCONDEN")
39    # pausa hasta que presione una tecla
40    system("pause")
41    # limpia la pantalla
42    system("cls")
43    # Seleccion de escondite de los jugadores
44    for i in range(0, len(jugadores)):
45        # Muestra todos los lugares donde se pueden encontrar
46        mostarEscondites(escondites)
```

```
47         system("cls")
48         print("Escondite no existente")
49     else:
50         juego[escondites[escondite]] = jugadores[i]
51         system("cls")
52         break
53     # limpia la pantalla
54     system("cls")
55
56     # Empieza a buscar a los demas jugadores
57     # si el numero del turnos es diferente de 0 y si las
58     # personas escondidas son diferente de 0 salimos de nuestro
59     # bucle
60     while turnos != 0 and escondidos != 0:
61         system("cls")
62         # muestra las personas por buscar y las oportunidades
63         # que tiene para encontrales
64         print("{} empieza a buscar a tus amigos, buen suerte
65             tienes {} oportunidades y tienes {} por buscar".format(
66                 buscador, turnos, escondidos))
67         # empeiza a caminar en el parque se pude eleguir
68         # entre 4 opciones
69         caminar = int(input(
70             "Te encuentras en un parque de tu ciudad puedes
71             caminar al \n1.-Norte\n2.-Este\n3.-Sur\n4.-Oeste\
72             nSeleccione:"))
73         system("cls")
74         # Camina Hacia el norte del parque
75         if caminar == 1:
76             # informacion donde nos encontramos ubicados
77             print("NORTE")
78             # si selecciona alguna de las opciones y si no se
79             # encuentra pierde 1 turno
80             buscarN = int(input(
81                 "Seleccione donde quiere buscar:\n1.-Arbol
82                     norte\n2.-Arbusto\n3.-Estatua\n0.-Regresar Al Parque\
83                     nSeleccione:"))
84             # si selecciona la opcion de arbol norte
85             if buscarN == 1:
86                 # comprueba si existe alguien ahi
87                 escondidos = seEncuentra("arbol_norte", juego,
88                 , escondidos)
89                 # se resta un turno
90                 turnos -= 1
91             # si selecciona la opcion de arbusto
92             elif buscarN == 2:
93                 # comprueba si existe alguien ahi
94                 escondidos = seEncuentra("arbusto", juego,
95                 , escondidos)
96                 # se resta un turno
```

```
85         turnos -= 1
86         # si selecciona la opcion de estatua
87     elif buscarN == 3:
88         # comprueba si existe alguien ahi
89         escondidos = seEncuentra("estatua", juego,
90         escondidos)
91         # se resta un turno
92         turnos -= 1
93         # si desea regresar al parque
94     elif buscarN == 0:
95         # regresa al parque principal
96         print("Regresando Al Parque")
97         # Camina Hacia el Este del parque
98     elif caminar == 2:
99         # informacion donde nos encontramos ubicados
100        print("ESTE")
101        # si selecciona alguna de las opciones y si no se
102        # encuetra pierde 1 turno
103        buscarE = int(input(
104            "Seleccione donde quiere buscar:\n1.-Arbol
105            este\n2.-Pileta\n3.-Casa del Arbol\n0.-Regresar Al Parque
106            \nSeleccione:"))
107        # si selecciona la opcion arbol este
108        if buscarE == 1:
109            # comprueba si existe alguien ahi
110            escondidos = seEncuentra("arbol_este", juego,
111            escondidos)
112            # se resta un turno
113            turnos -= 1
114            # si selecciona la opcion pileta
115        elif buscarE == 2:
116            # comprueba si existe alguien ahi
117            escondidos = seEncuentra("pileta", juego,
118            escondidos)
119            # se resta un turno
120            turnos -= 1
121            # si selecciona la opcion casa del arbol
122        elif buscarE == 3:
123            # comprueba si existe alguien ahi
124            escondidos = seEncuentra("casa_del_arbol",
125            juego, escondidos)
126            # se resta un turno
127            turnos -= 1
128            # si desea regresar al parque
129        elif buscarE == 0:
130            # regresa al parque principal
131            print("Regresando Al Parque")
132            # Camina Hacia el Sur del parque
133        elif caminar == 3:
134            # informacion donde nos encontramos ubicados
```

```
128         print("SUR")
129         # si selecciona alguna de las opciones y si no se
130         #encuentra pierde 1 turno
131         buscarS = int(input(
132             "Seleccione donde quiere buscar:\n1.-Arbol
133             sur\n2.-Juegos del parque sur\n3.-Caba a\n0.-Regresar Al
134             Parque\nSeleccione:"))
135         # si selecciona la opcion arbol sur
136         if buscarS == 1:
137             # comprueba si existe alguien ahi
138             escondidos = seEncuentra("arbol_sur", juego,
139             escondidos)
140             # se resta un turno
141             turnos -= 1
142             # si selecciona la opcion juegos del parque
143             elif buscarS == 2:
144                 # comprueba si existe alguien ahi
145                 escondidos = seEncuentra("juegos_del_parque",
146                 juego, escondidos)
147                 # se resta un turno
148                 turnos -= 1
149                 # si selecciona la opcion caba a
150                 elif buscarS == 3:
151                     # comprueba si existe alguien ahi
152                     escondidos = seEncuentra("caba a", juego,
153                     escondidos)
154                     # se resta un turno
155                     turnos -= 1
156                     # si desea regresar al parque
157                     elif buscarS == 0:
158                         # regresa al parque principal
159                         print("Regresando Al Parque")
160                         # Camina Hacia el Oeste del parque
161                         elif caminar == 4:
162                             #informacion donde nos encontramos ubicados
163                             ("OESTE")
164                             #si selecciona alguna de las opciones y si no se
165                             #encuentra pierde 1 turno
166                             buscar0 = int(input(
167                                 "Seleccione donde quiere buscar:\n1.-Arbol
168                                 oeste\n2.-Laguna\n0.-Regresar Al Parque\nSeleccione:"))
169                             # si selecciona la opcion arbol oeste
170                             if buscar0 == 1:
171                                 # comprueba si existe alguien ahi
172                                 escondidos = seEncuentra("arbol_oeste", juego
173 , escondidos)
174                                 # se resta un turno
175                                 turnos -= 1
176                                 # si selecciona la opcion laguna
177                                 elif buscar0 == 2:
```

```

169      # comprueba si existe alguien ahí
170      escondidos = seEncuentra("laguna", juego,
171      escondidos)
172      # se resta un turno
173      turnos -= 1
174      # si desea regresar al parque
175      elif buscar0 == 0:
176          # regresamos al parque principal
177          print("Regresando Al Parque")
178          system("pause")
179          # si el jugador principal encontro a todos el numero
180          # de escodidos sera 0 y ganara el juego
181          if (escondidos == 0):
182              # informacion de que gano la partida
183              print("Ganaste!!! eres un Capo :)")
184              # si no encuentra a todas las personas y su turno
185              son 0 pierde
186          else:
187              # informacion que dice que perdio el game
188              print("Perdiste :c ")

```

Código 1.13. Programa Principal

Para revisar el código desde su computador se adjunta el link del GitHub donde se ubica el proyecto: https://github.com/Antoni30/Juego_Tradicional_Las_Escondidas

https://github.com/NicolasSuquillo/modelos_Discretos.git

Pruebas de Escritorio La prueba de escritorio es una herramienta que nos servirsa saber si todos los procesos funcionan correctos

Prueba de escritorio					
numAlcantar	numJugadores	buscador	escocidos	turno	camino
1	5	—	4	10	—
3	5	carlos	4	10	—
3	5	carlos	4	10	—
3	5	carlos	4	9	1
3	5	carlos	3	8	4
3	5	carlos	2	7	2
3	5	carlos	1	6	3
3	5	carlos	0	5	4
carlos gana por que encontro a todos					

Fig. 23. Prueba de Escritorio

Casos de Prueba Probamos nuestro programa para saber como funciona

```

Buscador: Antoni, Jugadores: ['Gabriela', 'Mirian', 'Mishell', 'Rebeca']
JUGADORES!!! INTENTEN QUE EL BUSCADOR NO VEA DONDE SE ESCONDEN
Presione una tecla para continuar . . .

Antoni empieza a buscar a tus amigos, buen suerte tienes 10 oportunidades y tienes 4 por buscar
Te encuentras en un parque de tu ciudad puedes caminar al
1.-Norte
2.-Este
3.-Sur
4.-Oeste
Selecciona: 1

Antoni empieza a buscar a tus amigos, buen suerte tienes 8 oportunidades y tienes 3 por buscar
Te encuentras en un parque de tu ciudad puedes caminar al
1.-Norte
2.-Este
3.-Sur
4.-Oeste
Selecciona: 2

Selecione donde quiere buscar:
1.-Arbol norte
2.-Arbusto
3.-Plataforma
4.-Retrovisor
5.-Estatua
6.-Rearview
7.-Regresar Al Parque
Selecciona: 2

Encuentra a Gabriela
Presione una tecla para continuar . . .

NORTE
Selecione donde quiere buscar:
1.-Arbol norte
2.-Arbusto
3.-Plataforma
4.-Retrovisor
5.-Estatua
6.-Rearview
7.-Regresar Al Parque
Selecciona: 1

No se encuentra nadie
Presione una tecla para continuar . . .

Antoni empieza a buscar a tus amigos, buen suerte tienes 5 oportunidades y tienes 1 por buscar
Te encuentras en un parque de tu ciudad puedes caminar al
1.-Norte
2.-Este
3.-Sur
4.-Oeste
ESTE
Selecciona donde quiere buscar:
1.-Arbol este
2.-Piletta
3.-Plataforma del Arbol
4.-Retrovisor
5.-Estatua
6.-Rearview
7.-Regresar Al Parque
Selecciona: 2

Encuentra a Rebeca
Presione una tecla para continuar . . .
Ganaste!! eres un Capo :)
```

Fig. 24. Caso 1

The screenshot shows a terminal window with the following text:

```

Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda ← →
PROBLEMAS SALIDA CONSOLA DE DÉPUDACIÓN TERMINAL JUPYTER

Buscador: Juan, Jugadores: ['Antoni', 'Pedro']
JUGADORES!!! INTENTEN QUE EL BUSCADOR NO VEA DONDE SE ESCONDEN
Presione una tecla para continuar . . .

Juan empieza a buscar a tus amigos, buen suerte tienes 5 oportunidades y tienes 2 por buscar
Te encuentras en un parque de tu ciudad puedes caminar al
1.-Norte
2.-Este
3.-Sur
4.-Oeste
Selecciona: 1

Selecione donde quiere buscar:
1.-Arbol este
2.-Plataforma
3.-Plataforma
4.-Retrovisor
5.-Estatua
6.-Rearview
7.-Regresar Al Parque
Selecciona: 2

No se encuentra nadie
Presione una tecla para continuar . . .
Perdiste :)
```

Fig. 25. Caso 2

5 Discusión

Para la elaboración del juego se pudo notar que es muy importante, realizar un buen análisis del problema para poder elaborar el programa y así como también que el uso de las proposiciones lógicas se encuentra muy presente, siendo casi fundamentales para el desarrollo del mismo. A la hora de la elaboración del código



Fig. 26. Caso 3



Fig. 27. Caso 4

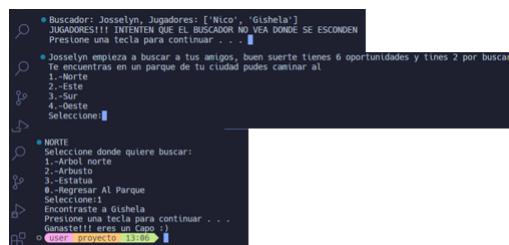


Fig. 28. Caso 5



Fig. 29. Caso 6

mediante Python se pudo ir viendo como es un lenguaje de fácil entendimiento que permite al desarrollador realizar los procesos necesarios sin la necesidad de muchas líneas de código y así llevando a realizar un código mas limpio.

6 Conclusiones

En conclusión, el uso de Python para desarrollar el programa fue una excelente elección ya que facilito mucho el proceso de creación del juego y así como también, la idea que se propuso inicialmente fue implementada con éxito, permitiendo un mayor entendimiento de el tema principal el cual es la lógica proporcional.

También se puede concluir que los condicionales de for y else son casi imprescindibles al momento de desarrollar un programa ya que pueden servir como la base y el generador de nuevas opciones

El uso de funciones permitió desarrollar un código más limpio y entendible. Se cumplió con el objetivo propuesto, con el menor porcentaje de errores.

References

1. L. Festinger and J. Carlsmith. Cognitive consequences of forced compliance. *The Journal of Abnormal and Social Psychology*, 1959.
2. K. Jarrod Millman and Michael Aivazis. Python for scientists and engineers. *Computing in Science & Engineering*, 13(2):9–12, March 2011.
3. Noé Fernández Pozo. Lenguajes de programación para la bioinformática. <http://www.encuentros.uma.es/encuentros134/lenguajes.pdf>. [Accessed: Nov 11, 2021].