INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
Reporte Programa Autómata Pila

Teoría de la computación

Profesor: Genaro Juárez Martínez
Realizado por: Kevin Atilano Gutiérrez

December 29, 2023

Contents

1 Introducción

Este código implementa una interfaz gráfica usando Tkinter para evaluar cadenas en el contexto de un lenguaje libre de contexto.

El programa utiliza una pila para realizar la verificación de la cadena.

Elementos del programa:

Interfaz Gráfica (GUI): Utiliza la biblioteca Tkinter para crear una ventana principal y un lienzo (canvas) en el que se mostrarán elementos gráficos.

Clase Pila: Implementa una pila básica con operaciones de push, pop y top. La pila se utiliza para realizar la verificación de la cadena.

Imágenes: Se utilizan dos imágenes (aceptada y no aceptada) para indicar si la cadena es válida o no.

Ejecución del Programa: El programa solicita al usuario que elija entre el modo manual (1) y el modo automático (2). En el modo manual, el usuario ingresa una cadena. En el modo automático, se genera una cadena aleatoria.

Evaluación de la Cadena: La cadena se evalúa caracter por caracter utilizando una pila. La interfaz gráfica se actualiza dinámicamente para mostrar la cadena actual y el estado de la pila.

Resultados en la GUI: Después de evaluar la cadena, se muestra en la interfaz gráfica si la cadena es aceptada o no, junto con una imagen correspondiente.

Limpiar la Interfaz: Después de cada ejecución, se ofrece al usuario la opción de ejecutar el programa nuevamente.

La función limpiar etiquetas se utiliza para limpiar la interfaz gráfica y el lienzo antes de ejecutar el programa nuevamente.

El código combina lógica de programación, manipulación de pilas y una interfaz gráfica para proporcionar una experiencia interactiva en la evaluación de cadenas según un lenguaje libre de contexto específico.

2 Código en Python

```
import sys
import time
import random
from tkinter import *
from tkinter import Label, PhotoImage
from PIL import Image, ImageTk

imagen_aceptada = Image.open("t.png")
imagen_aceptada = imagen_aceptada.resize((150, 150),
 Image.LANCZOS)

imagen_noaceptada = Image.open("p.png")
imagen_noaceptada = imagen_noaceptada.resize((150, 150),
 Image.LANCZOS)

class Pila(object):
    def __init__(self):
        super(Pila, self).__init__()
        self.arreglo = []
        self.cnt = 0

    def push(self, objeto):
        self.cnt += 1
        self.arreglo.append(objeto)

    def pop(self):
        if self.cnt == 0:
            return False
        else:
            self.cnt -= 1
            del self.arreglo[self.cnt]
            return True

    def top(self):
        return self.arreglo[self.cnt-1]
```

```
def ejecutar_programa(ventana, canv):
    archivo = open("Descripción Instantánea.txt", "w")

    xspeed = 5
    yspeed = 0

    print("\n")
    print("Lenguaje libre de contexto {0^n 1^n |
     n >= 1}\n")
    print("1)MODO MANUAL. 2)MODO AUTOMÁTICO\n")
    opc = input()

    if opc == "1":
        cad = input("CADENA INGRESADA:\n")
        archivo.write("CADENA INGRESADA: \n\n"
         + cad + "\n")
    else:
        rand = random.randrange(100000)
        cad = str(bin(rand)[2:])
        print("CADENA GENERADA:", cad)
        archivo.write("CADENA GENERADA:\n\n"+cad + "\n")

    cadc = cad[::-1]

    p = Pila()

    e = "a"
    aux1 = 0

    o = Label(ventana, text="CADENA A EVALUAR. \n\n"
     + cad).place(x=10, y=10)

    cont = 0

    conta = 0
    contb = 0
    conterr = 0
    cuenta = 0
    canv.create_rectangle(300, 50, 400, 550, width=0,
```

```
 fill='white')

if len(cad) <= 10:
    for i in cadc:
        u = Label(ventana, text="CARACTER A VERIFICAR
        \n\n "+cad[0:len(cad)-cont]).place
        (x=10, y=100)
        ventana.update()
        time.sleep(.5)
        cont = cont + 1

        if cadc[cuenta] == "1":
            cuenta = cuenta + 1
            p.push("X")
            print(p.arreglo)
            archivo.write("\n"+str(p.arreglo))
            aux1 = aux1 + 1
            e = "c"

            canv.create_rectangle(300, 500-conterr*
            25,
             400, 550-conterr*25, width=1, fill=
             'orange',
              outline='white')
            conterr = conterr+1
            time.sleep(.5)
            conterr = conterr + 1
            continue

        if i == "0" and e == "a":
            p.push(i)
            print(p.arreglo)
            archivo.write("\n"+str(p.arreglo))
            canv.create_rectangle(300, 500-conta*25,
            400,
             550-conta*25, width=1, fill='blue',
              outline='white')
            conta = conta+1
            time.sleep(.5)
```

```
        conta = conta + 1
        continue

    if i == "0" and e == "b":
        e = "c"
        p.push("X")
        print(p.arreglo)
        archivo.write("\n"+str(p.arreglo))
        continue

    if i == "0" and e == "c":
        p.push("X")
        print(p.arreglo)
        archivo.write("\n"+str(p.arreglo))
        continue

    if i == "1" and e == "c":
        p.push("X")
        print(p.arreglo)
        archivo.write("\n"+str(p.arreglo))
        canv.create_rectangle(300, 500-conterr*25,
         400,
         550-conterr*25, width=1, fill='orange',
          outline='white')
        conterr = conterr+1
        time.sleep(.5)
        conterr = conterr + 1
        continue

    if i == "1" and e == "a":
        e = "b"
        p.pop()
        print(p.arreglo)
        archivo.write("\n"+str(p.arreglo))
        aux1 = aux1 + 1
        canv.create_rectangle(300, 550-((conta)*25),
         400,
         600-(conta)*25, width=1, fill='white',
          outline='white')
```

```
                contb = contb+1
                time.sleep(.5)
                continue

            if i == "1" and len(p.arreglo) < aux1:
                p.push("X")
                print(p.arreglo)
                archivo.write("\n"+str(p.arreglo))
                aux1 = aux1 + 1
                canv.create_rectangle(300, 500-conterr*25,
                 400,
                550-conterr*25, width=1, fill='orange',
                outline='white')
                conterr = conterr+1
                time.sleep(.5)
                conterr = conterr + 1
                continue

            if i == "1" and e == "b":
                p.pop()
                print(p.arreglo)
                archivo.write("\n"+str(p.arreglo))
                canv.create_rectangle(300, 550-
                ((conta)*25)+contb*50,
                400, 600-(conta)*25+contb*50, width=1,
                fill='white', outline='white')
                time.sleep(.5)
                contb = contb + 1
                continue
    else:

        for i in cadc:
            if cadc[cuenta] == "1":
                cuenta = cuenta + 1
                p.push("X")
                print(p.arreglo)
                archivo.write("\n"+str(p.arreglo))
                aux1 = aux1 + 1
                e = "c"
```

```
        continue

    if i == "0" and e == "a":
        p.push(i)
        print(p.arreglo)
        archivo.write("\n"+str(p.arreglo))
        continue

    if i == "0" and e == "b":
        e = "c"
        p.push("X")
        print(p.arreglo)
        archivo.write("\n"+str(p.arreglo))
        continue

    if i == "0" and e == "c":
        p.push("X")
        print(p.arreglo)
        archivo.write("\n"+str(p.arreglo))
        continue

    if i == "1" and e == "c":
        p.push("X")
        print(p.arreglo)
        archivo.write("\n"+str(p.arreglo))
        continue

    if i == "1" and e == "a":
        e = "b"
        p.pop()
        print(p.arreglo)
        archivo.write("\n"+str(p.arreglo))
        aux1 = aux1 + 1
        continue

    if i == "1" and len(p.arreglo) < aux1:
        p.push("X")
        print(p.arreglo)
        archivo.write("\n"+str(p.arreglo))
```

```
            aux1 = aux1 + 1
            continue

        if i == "1" and e == "b":
            p.pop()
            print(p.arreglo)
            archivo.write("\n"+str(p.arreglo))
            continue

a = len(p.arreglo)
if a == 0:
    print("CADENA ACEPTADA")
    archivo.write("\n\nCADENA ACEPTADA")
    imagen_aceptada_tk = ImageTk.PhotoImage
    (imagen_aceptada)
    o = Label(ventana, image=imagen_aceptada_tk)
    o.image = imagen_aceptada_tk
    o.place(x=500, y=300)
    canv.create_rectangle(300, 50, 400, 550,
    width=0, fill='green')
else:
    print("CADENA INVALIDA")
    archivo.write("\n\nCADENA INVALIDA")
    imagen_noaceptada_tk = ImageTk.PhotoImage
    (imagen_noaceptada)
    o = Label(ventana, image=imagen_noaceptada_tk)
    o.image = imagen_noaceptada_tk
    o.place(x=500, y=300)
    canv.create_rectangle(300, 50, 400, 550,
     width=0, fill='red')

def limpiar_etiquetas(ventana):
    for widget in ventana.winfo_children():
        if isinstance(widget, Label):
            widget.destroy()
    canv.delete(ALL)

archivo.close()
```

```
    respuesta = input("¿DESEA EJECUTAR NUEVAMENTE
    (S/N): ")
    if respuesta.upper() != 'S':
        limpiar_etiquetas(ventana)
        ventana.destroy()
        sys.exit()
    else:
         limpiar_etiquetas(ventana)

def main():
    ventana = Tk()
    ventana.title("Lenguaje libre de contexto
    {0^n 1^n | n >= 1}")
    canv = Canvas(ventana, width=800, height=600)
    ventana.geometry("800x600")
    canv.pack()

    while True:
        ejecutar_programa(ventana, canv)

if __name__ == "__main__":
    main()
```

3 Análisis de código en Pyhton

Análisis de código.

- Importa las clases Image y ImageTk del módulo PIL (Pillow). Estas clases se utilizan para manipular imágenes, y ImageTk se utiliza para mostrar imágenes en la interfaz gráfica de Tkinter.

```
from PIL import Image, ImageTk
```

- Abre la imagen llamada "t.png", la redimensiona a 150x150 píxeles utilizando el método resize y el algoritmo de interpolación LANCZOS, y almacena la imagen resultante en la variable imagen aceptada.

```
imagen_aceptada = Image.open("t.png")
imagen_aceptada = imagen_aceptada.resize((150, 150),
Image.LANCZOS)
```

- Similar al bloque anterior, abre la imagen llamada "p.png", la redimensiona a 150x150 píxeles utilizando el método resize y el algoritmo de interpolación LANCZOS, y almacena la imagen resultante en la variable imagen noaceptada.

```
imagen_noaceptada = Image.open("p.png")
imagen_noaceptada = imagen_noaceptada.resize((150, 150),
Image.LANCZOS)
```

- Define la clase Pila.

```
class Pila(object):
```

- Inicializa una instancia de la clase Pila con un arreglo vacío (self.arreglo) y un contador en cero (self.cnt).

```
def __init__(self):
    super(Pila, self).__init__()
    self.arreglo = []
    self.cnt = 0
```

- Inicializa una instancia de la clase Pila con un arreglo vacío (self.arreglo) y un contador en cero (self.cnt).

```
def __init__(self):
super(Pila, self).__init__()
self.arreglo = []
self.cnt = 0
```

- Define un método push que agrega un objeto al final del arreglo y aumenta el contador en uno.

```
def push(self, objeto):
    self.cnt += 1
    self.arreglo.append(objeto)
```

- Define un método pop que elimina el último elemento del arreglo si la pila no está vacía.

```
def pop(self):
    if self.cnt == 0:
        return False
    else:
        self.cnt -= 1
        del self.arreglo[self.cnt]
        return True
```

- Define un método top que devuelve el elemento en la cima de la pila.

```
def top(self):
    return self.arreglo[self.cnt-1]
```

- Define la función ejecutar programa que toma dos parámetros, ventana y canv.

```
def ejecutar_programa(ventana, canv):
```

- Abre un archivo llamado "Descripción Instantánea.txt" en modo escritura ("w").

```
archivo = open("Descripción Instantánea.txt", "w")
```

- Inicializa las variables xspeed e yspeed con valores específicos.

```
xspeed = 5
yspeed = 0
```

- Muestra mensajes en la consola y solicita al usuario que ingrese una opción ("1" para modo manual, "2" para modo automático).

```
print("\n")
print("Lenguaje libre de contexto {0^n 1^n | n >= 1}\n")
print("1)MODO MANUAL. 2)MODO AUTOMÁTICO\n")
opc = input()
```

- Dependiendo de la opción ingresada, solicita al usuario que ingrese una cadena manualmente o genera una cadena aleatoria y la escribe en el archivo.

```
if opc == "1":
    cad = input("CADENA INGRESADA:\n")
    archivo.write("CADENA INGRESADA: \n\n" + cad + "\n")
else:
    rand = random.randrange(100000)
    cad = str(bin(rand)[2:])
    print("CADENA GENERADA:", cad)
    archivo.write("CADENA GENERADA:\n\n"+cad + "\n")
```

- Crea una cadena invertida cadc a partir de la cadena cad.

```
cadc = cad[::-1]
```

- Crea una instancia de la clase Pila, inicializa la variable e con el valor "a" y la variable aux1 en cero.

```
p = Pila()
e = "a"
aux1 = 0
```

- Crea un objeto de la clase Label que muestra un texto en la ventana de la interfaz gráfica.

```
o = Label(ventana, text="CADENA A EVALUAR.
 \n\n" + cad).place
(x=10, y=10)
```

- Inicializa varias variables y crea un rectángulo blanco en el lienzo de la interfaz gráfica.

```
cont = 0
conta = 0
contb = 0
conterr = 0
cuenta = 0
canv.create_rectangle(300, 50, 400, 550,
 width=0, fill='white')
```

- Comprueba si la longitud de la cadena es menor o igual a 10.

```
if len(cad) <= 10:
```

- Inicia un bucle for para iterar sobre los caracteres de la cadena invertida.

```
for i in cadc:
```

- Crea un objeto de la clase Label que muestra información sobre el carácter actual en la ventana.

```
u = Label(ventana, text="CARACTER A
VERIFICAR \n\n "+
cad[0:len(cad)-cont]).place(x=10,
 y=100)
```

- Actualiza la ventana y espera medio segundo para proporcionar una animación.

```
ventana.update()
time.sleep(.5)
```

- Incrementa el contador en uno.

```
cont = cont + 1
```

- Comprueba si el carácter actual en cadc es ”1”.

```
if cadc[cuenta] == "1":
```

- Realiza operaciones si se cumple la condición anterior: incrementa cuenta, agrega ”X” a la pila, imprime la pila y actualiza aux1 y e.

```
cuenta = cuenta + 1
p.push("X")
print(p.arreglo)
archivo.write("\n"+str(p.arreglo))
aux1 = aux1 + 1
e = "c"
```

- Crea un rectángulo naranja en el lienzo, espera medio segundo y continua con la siguiente iteración.

```
canv.create_rectangle(300, 500-conterr*25,
 400, 550-conterr*25, width=1, fill='orange',
```

```
  outline='white')
conterr = conterr+1
time.sleep(.5)
conterr = conterr + 1
continue
```

- Este bloque de código verifica si el carácter actual es "0" y la variable e es "a". En ese caso, se realiza una serie de acciones, como hacer push en la pila, imprimir la pila, escribir en el archivo, crear un rectángulo azul en el lienzo, y hacer una pausa antes de continuar con la siguiente iteración.

```
if i == "0" and e == "a":
p.push(i)
print(p.arreglo)
archivo.write("\n"+str(p.arreglo))
canv.create_rectangle(300, 500-conta*25,
400, 550-conta*25, width=1, fill='blue',
 outline='white')
conta = conta+1
time.sleep(.5)
conta = conta + 1
continue
```

- Esta parte del código maneja el caso en que la longitud de la cadena es mayor que 10

```
else:
    for i in cadc:
```

- Estos bloques evalúan el estado final de la pila después de procesar la cadena. Si la pila está vacía, se considera que la cadena es aceptada y se muestra un mensaje en la consola, se escribe en el archivo y se muestra una imagen verde en la interfaz gráfica. En caso contrario, se considera que la cadena es inválida, se realiza la acción correspondiente y se muestra una imagen roja en la interfaz gráfica.

```
a = len(p.arreglo)
if a == 0:
    print("CADENA ACEPTADA")
    archivo.write("\n\nCADENA ACEPTADA")
```

```
        imagen_aceptada_tk = ImageTk.PhotoImage
        (imagen_aceptada)
        o = Label(ventana, image=imagen_aceptada_tk)
        o.image = imagen_aceptada_tk
        o.place(x=500, y=300)
        canv.create_rectangle(300, 50, 400, 550,
         width=0, fill='green')
    else:
        print("CADENA INVALIDA")
        archivo.write("\n\nCADENA INVALIDA")
        imagen_noaceptada_tk = ImageTk.PhotoImage
        (imagen_noaceptada)
        o = Label(ventana, image=imagen_noaceptada_tk)
        o.image = imagen_noaceptada_tk
        o.place(x=500, y=300)
        canv.create_rectangle(300, 50, 400, 550,
        width=0, fill='red')
```

- Define una función llamada limpiar etiquetas que elimina todos los widgets de tipo Label en la ventana y borra todo el contenido del lienzo.

```
def limpiar_etiquetas(ventana):
    for widget in ventana.winfo_children():
        if isinstance(widget, Label):
            widget.destroy()
    canv.delete(ALL)
```

- Cierra el archivo abierto anteriormente.

```
archivo.close()
```

- Pregunta al usuario si desea ejecutar el programa nuevamente. Si la respuesta no es 'S' (mayúscula), limpia la interfaz gráfica y termina el programa. Si la respuesta es 'S', limpia la interfaz gráfica.

```
respuesta = input("¿DESEA EJECUTAR NUEVAMENTE
(S/N): ")
if respuesta.upper() != 'S':
    limpiar_etiquetas(ventana)
    ventana.destroy()
```

```
        sys.exit()
    else:
         limpiar_etiquetas(ventana)
```

- Define la función principal main que crea una ventana y un lienzo, establece el título y las dimensiones, y ejecuta indefinidamente la función ejecutar programa dentro de un bucle while.

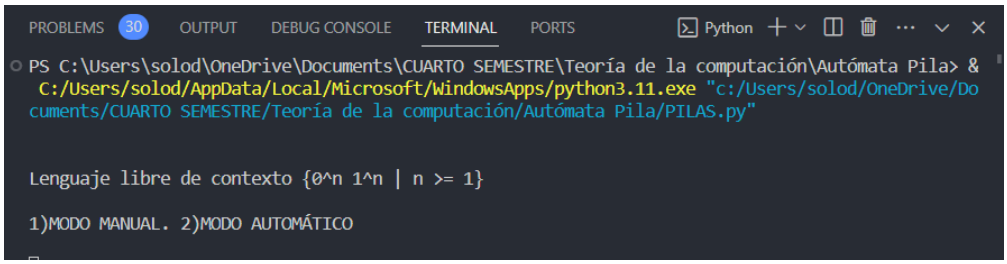
```
def main():
ventana = Tk()
ventana.title("Lenguaje libre de contexto
{0^n 1^n | n >= 1}")
canv = Canvas(ventana, width=800, height=600)
ventana.geometry("800x600")
canv.pack()

while True:
    ejecutar_programa(ventana, canv)
```

- Verifica si el script está siendo ejecutado como un programa independiente y, en ese caso, llama a la función principal main.

```
if __name__ == "__main__":
main()
```

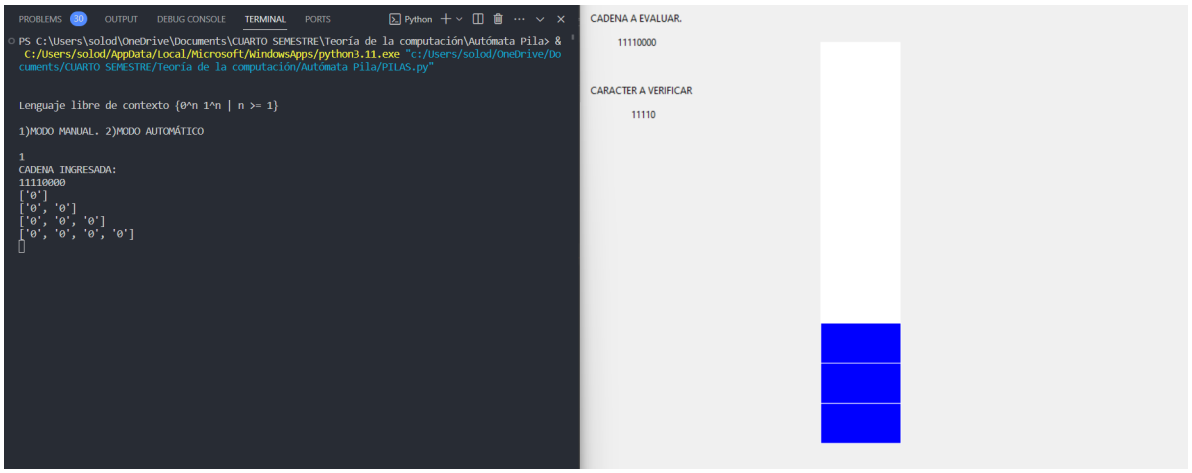
4 Ejecución

- Compilacion

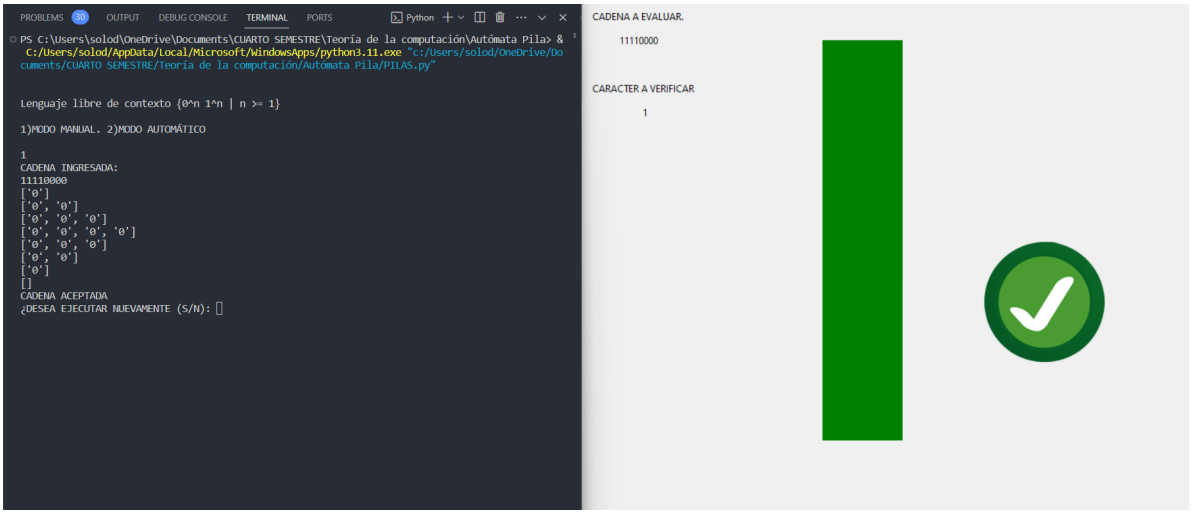


- Modo Manual

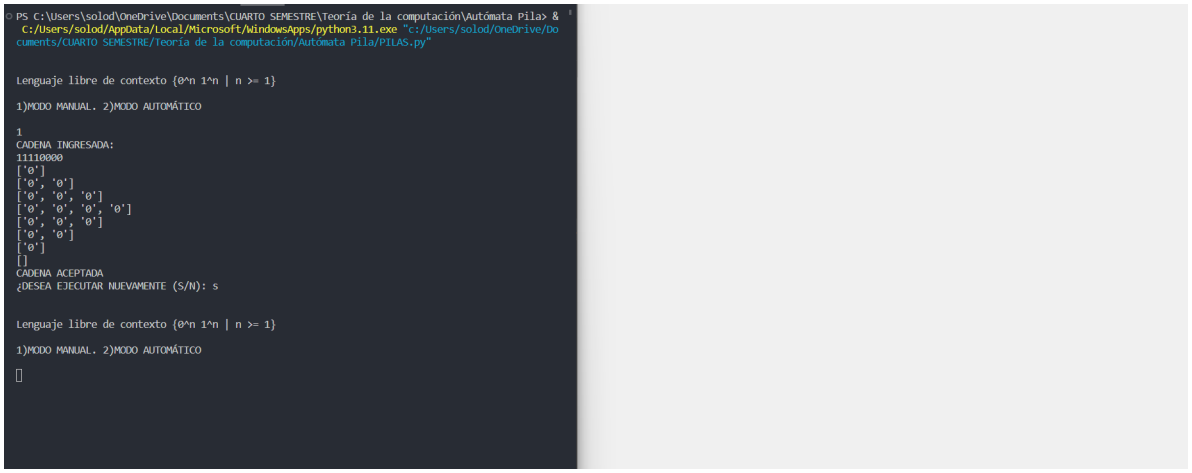
  Prueba con modo manual, se selecciona la opción 1 y se ingresa la cadena manualmente, en este caso se ingresó la cadena 11110000.



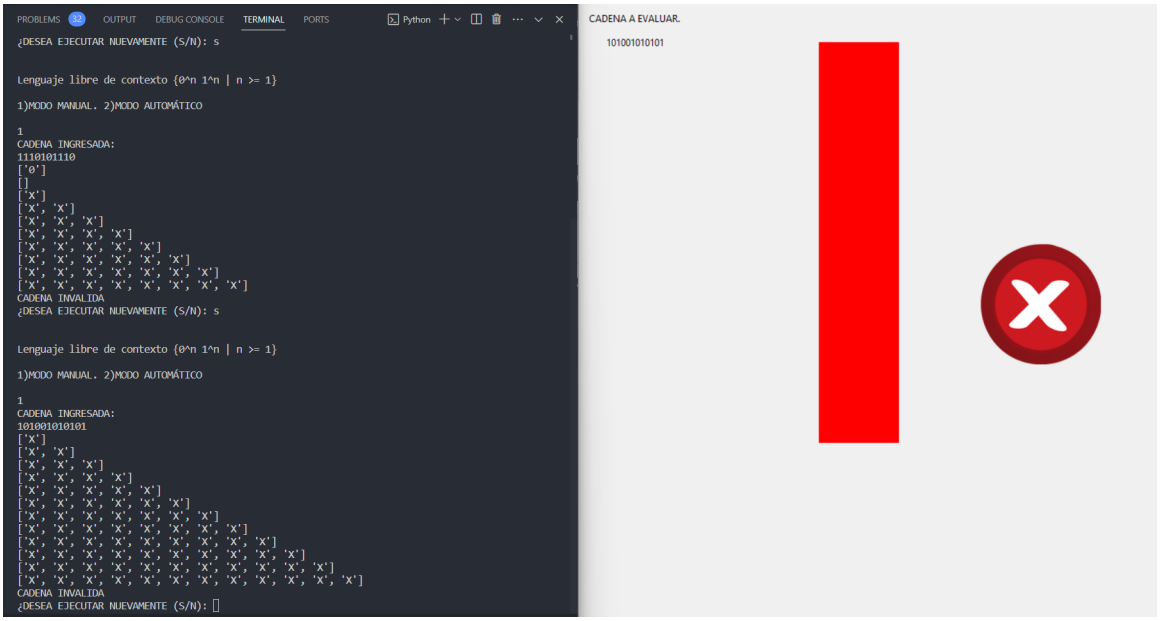
  La cadena respeta las condiciones de la pila, por tanto se valida y se colorea la pila de color verde, además de que se muestra una paloma dando a entender que es válida la cadena.



Al término de la ejecución se muestra un mensaje el cual permite volver a realizar una ejecución más del autómata, con tan solo escribir la letra "S", al ingresar la "S" se limpian los labels y el lienzo.



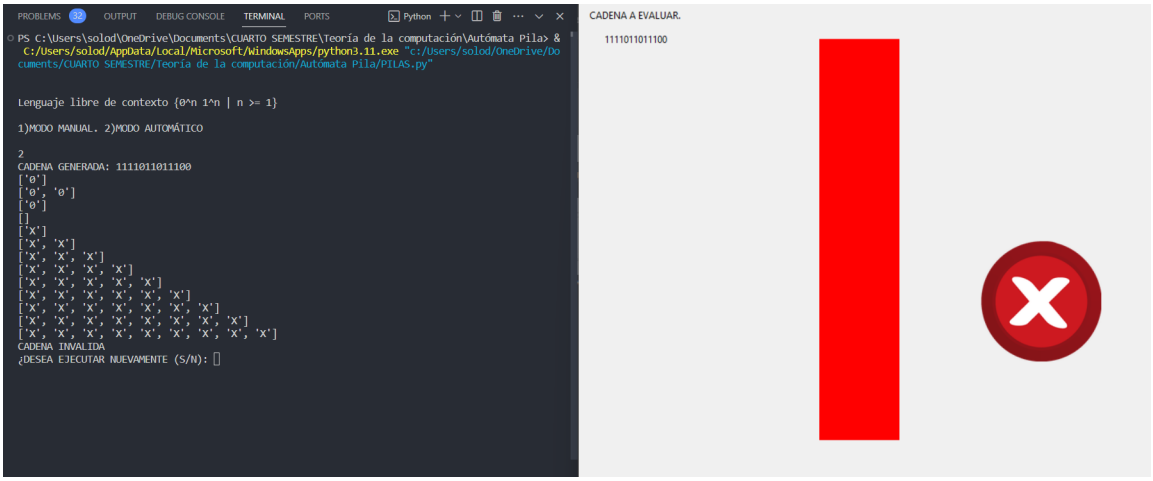
Ahora ingresaremos una cadena de más de 10 carácteres, al hacerlo podemos apreciar que no se va mostrando el proceso de la pila, si no que solo la descripción y se pinta del color correspondiente el lienzo. En este caso es de color rojo porque la cadena no es válida.



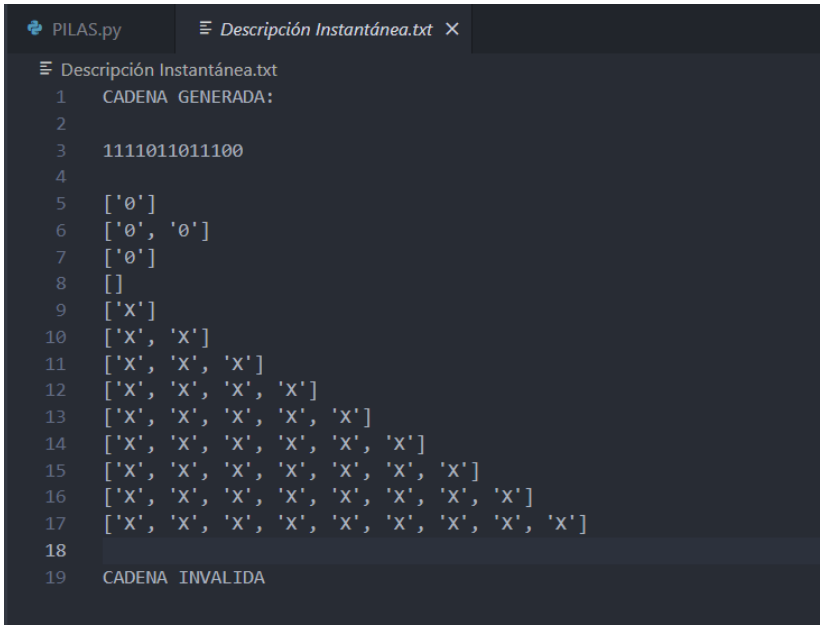
Se genera un archivo llamado ”Descripción Instantánea.txt”, el cual como se menciona contiene la descripción instantánea de la cadena.

- Modo Autómatico

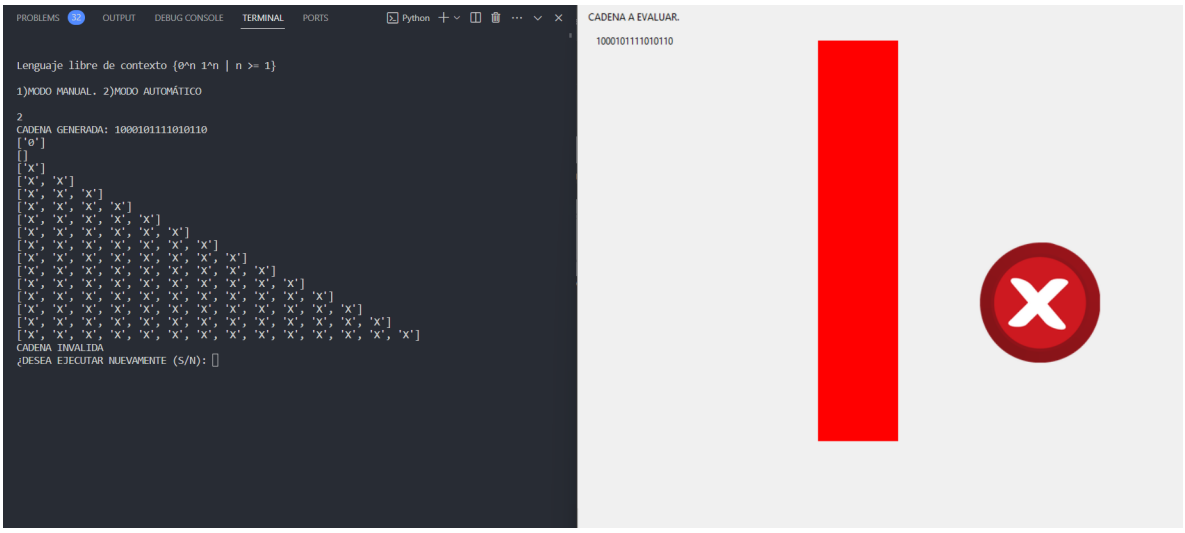
  La segunda opción es el Modo Autómatico y consiste en que el mismo programa ingresará una cadena aleatoria a evaluar.



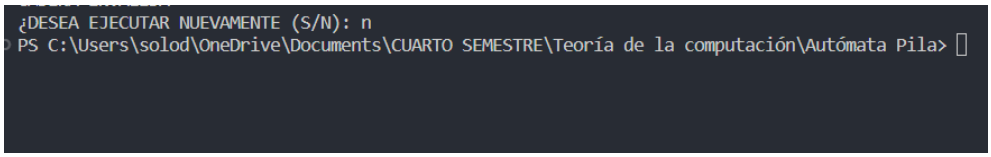
  En este caso el programa ingresó la cadena 1111011011100 y el txt también nos muestra la cadena y si es válida o no.



La cadena se evalúa y al no cumplir los requisitos, es decir no es válida esta se pinta la pila de rojo.



Al final si ya no se requiere realizar otra ejecución, solo se selecciona ”N” y el programa termina.



5 Código en Latex

```
    \documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage{graphicx}
\usepackage{verbatim}
\usepackage{hyperref}
\usepackage{geometry}

\geometry{a4paper, margin=1in}
\graphicspath{ {C:\Users\solod\OneDrive\Documents\CUARTO
SEMESTRE\Teoría de la computación\Autómata Pila} }
\begin{document}

\begin{titlepage}
    \centering
    \includegraphics[width=0.4\textwidth]{logo1.png}\par\vspace{1cm}
    \includegraphics[width=0.4\textwidth]{logo2.png}\par\vspace{1cm}
    {\scshape\Huge INSTITUTO POLITÉCNICO NACIONAL \par}
    {\scshape\Huge ESCUELA SUPERIOR DE CÓMPUTO \par}
    {\scshape\Huge Reporte Programa Autómata Pila \par}
    \vspace{2cm}
    {\huge Teoría de la computación \par}
    {\huge Profesor: Genaro Juárez Martínez \par}
    {\huge Realizado por: Kevin Atilano Gutiérrez \par}
    \vspace{2cm}
    {\large \today\par}
\end{titlepage}

{\Large\tableofcontents}
\newpage
\section{Introducción}
{\Large
Este código implementa una interfaz gráfica usando Tkinter
para evaluar cadenas en el contexto de un lenguaje libre
de contexto.\\
\\
El programa utiliza una pila para realiza
r la verificación de la cadena.
```

\\
\\
Elementos del programa:
\\
\\
Interfaz Gráfica (GUI): Utiliza la biblioteca Tkinter
para crear una ventana principal y un lienzo (canvas)
en el que se mostrarán elementos gráficos.
\\
\\
Clase Pila:
Implementa una pila básica con operaciones de push,
pop y top.
La pila se utiliza para realizar la verificación
de la cadena.
\\
\\
Imágenes:
Se utilizan dos imágenes (aceptada y no aceptada)
para indicar si la cadena es válida o no.
\\
\\
Ejecución del Programa:
El programa solicita al usuario que elija entre el
modo manual (1) y el modo automático (2).
En el modo manual, el usuario ingresa una cadena.
En el modo automático, se genera una cadena aleatoria.
\\
\\
Evaluación de la Cadena:
La cadena se evalúa caracter por caracter utilizando
una pila.
La interfaz gráfica se actualiza dinámicamente para
mostrar la cadena actual y el estado de la pila.
\\
\\
Resultados en la GUI:
Después de evaluar la cadena, se muestra en la interfaz
gráfica si la cadena es aceptada o no, junto con una

```
imagen correspondiente.
\\
\\
Limpiar la Interfaz:
Después de cada ejecución, se ofrece al usuario la
opción de ejecutar el programa nuevamente.
\\
\\
La función limpiar etiquetas se utiliza para limpiar la
interfaz gráfica y el lienzo antes de ejecutar el programa
nuevamente.
\\\\
El código combina lógica de programación, manipulación
 de pilas y una interfaz gráfica para proporcionar una
 experiencia interactiva en la evaluación de cadenas
 según un lenguaje libre de contexto específico.
}
\newpage
\section{Código en Python}
{\Large
\begin{verbatim}
    import sys
    import time
    import random
    from tkinter import *
    from tkinter import Label, PhotoImage
    from PIL import Image, ImageTk

    imagen_aceptada = Image.open("t.png")
    imagen_aceptada = imagen_aceptada.resize((150, 150),
     Image.LANCZOS)

    imagen_noaceptada = Image.open("p.png")
    imagen_noaceptada = imagen_noaceptada.resize((150, 150),
     Image.LANCZOS)

    class Pila(object):
        def __init__(self):
            super(Pila, self).__init__()
```

```
        self.arreglo = []
        self.cnt = 0

    def push(self, objeto):
        self.cnt += 1
        self.arreglo.append(objeto)

    def pop(self):
        if self.cnt == 0:
            return False
        else:
            self.cnt -= 1
            del self.arreglo[self.cnt]
            return True

    def top(self):
        return self.arreglo[self.cnt-1]


def ejecutar_programa(ventana, canv):
    archivo = open("Descripción Instantánea.txt", "w")

    xspeed = 5
    yspeed = 0

    print("\n")
    print("Lenguaje libre de contexto {0^n 1^n |
     n >= 1}\n")
    print("1)MODO MANUAL. 2)MODO AUTOMÁTICO\n")
    opc = input()

    if opc == "1":
        cad = input("CADENA INGRESADA:\n")
        archivo.write("CADENA INGRESADA: \n\n"
         + cad + "\n")
    else:
        rand = random.randrange(100000)
        cad = str(bin(rand)[2:])
        print("CADENA GENERADA:", cad)
```

```
    archivo.write("CADENA GENERADA:\n\n"+cad + "\n")

cadc = cad[::-1]

p = Pila()

e = "a"
aux1 = 0

o = Label(ventana, text="CADENA A EVALUAR. \n\n"
 + cad).place(x=10, y=10)

cont = 0

conta = 0
contb = 0
conterr = 0
cuenta = 0
canv.create_rectangle(300, 50, 400, 550, width=0,
 fill='white')

if len(cad) <= 10:
    for i in cadc:
        u = Label(ventana, text="CARACTER A VERIFICAR
        \n\n "+cad[0:len(cad)-cont]).place
        (x=10, y=100)
        ventana.update()
        time.sleep(.5)
        cont = cont + 1

        if cadc[cuenta] == "1":
            cuenta = cuenta + 1
            p.push("X")
            print(p.arreglo)
            archivo.write("\n"+str(p.arreglo))
            aux1 = aux1 + 1
            e = "c"

            canv.create_rectangle(300, 500-conterr*
```

```
        25,
         400, 550-conterr*25, width=1, fill=
         'orange',
          outline='white')
        conterr = conterr+1
        time.sleep(.5)
        conterr = conterr + 1
        continue

    if i == "0" and e == "a":
        p.push(i)
        print(p.arreglo)
        archivo.write("\n"+str(p.arreglo))
        canv.create_rectangle(300, 500-conta*25,
        400,
         550-conta*25, width=1, fill='blue',
          outline='white')
        conta = conta+1
        time.sleep(.5)
        conta = conta + 1
        continue

    if i == "0" and e == "b":
        e = "c"
        p.push("X")
        print(p.arreglo)
        archivo.write("\n"+str(p.arreglo))
        continue

    if i == "0" and e == "c":
        p.push("X")
        print(p.arreglo)
        archivo.write("\n"+str(p.arreglo))
        continue

    if i == "1" and e == "c":
        p.push("X")
        print(p.arreglo)
        archivo.write("\n"+str(p.arreglo))
```

```
            canv.create_rectangle(300, 500-conterr*25,
             400,
             550-conterr*25, width=1, fill='orange',
              outline='white')
            conterr = conterr+1
            time.sleep(.5)
            conterr = conterr + 1
            continue

        if i == "1" and e == "a":
            e = "b"
            p.pop()
            print(p.arreglo)
            archivo.write("\n"+str(p.arreglo))
            aux1 = aux1 + 1
            canv.create_rectangle(300, 550-((conta)*25),
             400,
             600-(conta)*25, width=1, fill='white',
              outline='white')
            contb = contb+1
            time.sleep(.5)
            continue

        if i == "1" and len(p.arreglo) < aux1:
            p.push("X")
            print(p.arreglo)
            archivo.write("\n"+str(p.arreglo))
            aux1 = aux1 + 1
            canv.create_rectangle(300, 500-conterr*25,
             400,
            550-conterr*25, width=1, fill='orange',
            outline='white')
            conterr = conterr+1
            time.sleep(.5)
            conterr = conterr + 1
            continue

        if i == "1" and e == "b":
            p.pop()
```

```
                print(p.arreglo)
                archivo.write("\n"+str(p.arreglo))
                canv.create_rectangle(300, 550-
                ((conta)*25)+contb*50,
                400, 600-(conta)*25+contb*50, width=1,
                fill='white', outline='white')
                time.sleep(.5)
                contb = contb + 1
                continue
    else:

        for i in cadc:
            if cadc[cuenta] == "1":
                cuenta = cuenta + 1
                p.push("X")
                print(p.arreglo)
                archivo.write("\n"+str(p.arreglo))
                aux1 = aux1 + 1
                e = "c"
                continue

            if i == "0" and e == "a":
                p.push(i)
                print(p.arreglo)
                archivo.write("\n"+str(p.arreglo))
                continue

            if i == "0" and e == "b":
                e = "c"
                p.push("X")
                print(p.arreglo)
                archivo.write("\n"+str(p.arreglo))
                continue

            if i == "0" and e == "c":
                p.push("X")
                print(p.arreglo)
                archivo.write("\n"+str(p.arreglo))
                continue
```

```
            if i == "1" and e == "c":
                p.push("X")
                print(p.arreglo)
                archivo.write("\n"+str(p.arreglo))
                continue

            if i == "1" and e == "a":
                e = "b"
                p.pop()
                print(p.arreglo)
                archivo.write("\n"+str(p.arreglo))
                aux1 = aux1 + 1
                continue

            if i == "1" and len(p.arreglo) < aux1:
                p.push("X")
                print(p.arreglo)
                archivo.write("\n"+str(p.arreglo))
                aux1 = aux1 + 1
                continue

            if i == "1" and e == "b":
                p.pop()
                print(p.arreglo)
                archivo.write("\n"+str(p.arreglo))
                continue

    a = len(p.arreglo)
    if a == 0:
        print("CADENA ACEPTADA")
        archivo.write("\n\nCADENA ACEPTADA")
        imagen_aceptada_tk = ImageTk.PhotoImage
        (imagen_aceptada)
        o = Label(ventana, image=imagen_aceptada_tk)
        o.image = imagen_aceptada_tk
        o.place(x=500, y=300)
        canv.create_rectangle(300, 50, 400, 550,
        width=0, fill='green')
```

```
    else:
        print("CADENA INVALIDA")
        archivo.write("\n\nCADENA INVALIDA")
        imagen_noaceptada_tk = ImageTk.PhotoImage
        (imagen_noaceptada)
        o = Label(ventana, image=imagen_noaceptada_tk)
        o.image = imagen_noaceptada_tk
        o.place(x=500, y=300)
        canv.create_rectangle(300, 50, 400, 550,
         width=0, fill='red')

    def limpiar_etiquetas(ventana):
        for widget in ventana.winfo_children():
            if isinstance(widget, Label):
                widget.destroy()
        canv.delete(ALL)

    archivo.close()

    respuesta = input("¿DESEA EJECUTAR NUEVAMENTE
    (S/N): ")
    if respuesta.upper() != 'S':
        limpiar_etiquetas(ventana)
        ventana.destroy()
        sys.exit()
    else:
         limpiar_etiquetas(ventana)

def main():
    ventana = Tk()
    ventana.title("Lenguaje libre de contexto
    {0^n 1^n | n >= 1}")
    canv = Canvas(ventana, width=800, height=600)
    ventana.geometry("800x600")
    canv.pack()

    while True:
        ejecutar_programa(ventana, canv)
```

```
    if __name__ == "__main__":
        main()



}


\newpage
\section{Análisis de código en Pyhton}
{\Large
Análisis de código.
\begin{itemize}
\item Importa las clases Image y ImageTk del módulo
 PIL (Pillow).
Estas clases se utilizan para manipular imágenes,
 y ImageTk
se utiliza para mostrar imágenes en la interfaz
gráfica de Tkinter.
\begin{verbatim}
    from PIL import Image, ImageTk

\item Abre la imagen llamada "t.png", la redimensiona
a 150x150 píxeles utilizando el método resize y
el algoritmo de interpolación LANCZOS, y
almacena la imagen resultante en la variable imagen aceptada.
\begin{verbatim}
    imagen_aceptada = Image.open("t.png")
    imagen_aceptada = imagen_aceptada.resize((150, 150),
    Image.LANCZOS)

\item Similar al bloque anterior, abre la imagen llamada
"p.png", la redimensiona a 150x150 píxeles
utilizando el método resize y el algoritmo de
interpolación LANCZOS,
y almacena la imagen resultante en la variable
 imagen noaceptada.
\begin{verbatim}
    imagen_noaceptada = Image.open("p.png")
```

```
    imagen_noaceptada = imagen_noaceptada.resize((150, 150),
    Image.LANCZOS)

\item  Define la clase Pila.
\begin{verbatim}
    class Pila(object):

\item  Inicializa una instancia de la clase Pila con
un arreglo vacío (self.arreglo) y
un contador en cero (self.cnt).
\begin{verbatim}
    def __init__(self):
        super(Pila, self).__init__()
        self.arreglo = []
        self.cnt = 0

\item  Inicializa una instancia de la clase Pila con un
arreglo vacío (self.arreglo) y
 un contador en cero (self.cnt).
\begin{verbatim}
    def __init__(self):
    super(Pila, self).__init__()
    self.arreglo = []
    self.cnt = 0

\item Define un método push que agrega
un objeto al final del arreglo y aumenta el contador
 en uno.
\begin{verbatim}
    def push(self, objeto):
        self.cnt += 1
        self.arreglo.append(objeto)
\item Define un método pop que elimina el
 último elemento del arreglo si la pila no está vacía.
\begin{verbatim}
    def pop(self):
        if self.cnt == 0:
            return False
        else:
```

```
            self.cnt -= 1
            del self.arreglo[self.cnt]
            return True

\item  Define un método top que devuelve el elemento
en la cima de la pila.
\begin{verbatim}
    def top(self):
        return self.arreglo[self.cnt-1]

\item  Define la función ejecutar programa
 que toma dos parámetros, ventana y canv.
\begin{verbatim}
    def ejecutar_programa(ventana, canv):

\item Abre un archivo llamado "Descripción Instantánea.txt"
 en modo escritura ("w").
\begin{verbatim}
    archivo = open("Descripción Instantánea.txt", "w")

\item  Inicializa las variables xspeed e yspeed con
valores específicos.
\begin{verbatim}
    xspeed = 5
    yspeed = 0

\item Muestra mensajes en la consola y solicita al
usuario que ingrese
 una opción ("1" para modo manual, "2"
 para modo automático).
\begin{verbatim}
    print("\n")
    print("Lenguaje libre de contexto {0^n 1^n | n >= 1}\n")
    print("1)MODO MANUAL. 2)MODO AUTOMÁTICO\n")
    opc = input()

\item Dependiendo de la opción ingresada, solicita
 al usuario que ingrese una cadena manualmente
 o genera una cadena aleatoria y la escribe en el archivo.
```

```
\begin{verbatim}
    if opc == "1":
        cad = input("CADENA INGRESADA:\n")
        archivo.write("CADENA INGRESADA: \n\n" + cad + "\n")
    else:
        rand = random.randrange(100000)
        cad = str(bin(rand)[2:])
        print("CADENA GENERADA:", cad)
        archivo.write("CADENA GENERADA:\n\n"+cad + "\n")

\item Crea una cadena invertida cadc a partir de la cadena cad.
\begin{verbatim}
    cadc = cad[::-1]

\item Crea una instancia de la clase Pila, inicializa
la variable e con el valor "a" y la variable aux1 en cero.
\begin{verbatim}
    p = Pila()
    e = "a"
    aux1 = 0

\item Crea un objeto de la clase
Label que muestra un texto en la ventana de la
interfaz gráfica.
\begin{verbatim}
    o = Label(ventana, text="CADENA A EVALUAR.
     \n\n" + cad).place
    (x=10, y=10)

\item  Inicializa varias variables y crea
un rectángulo blanco en el lienzo de la interfaz gráfica.
\begin{verbatim}
    cont = 0
    conta = 0
    contb = 0
    conterr = 0
    cuenta = 0
    canv.create_rectangle(300, 50, 400, 550,
     width=0, fill='white')
```

```
\item  Comprueba si la longitud de la cadena es
 menor o igual a 10.
\begin{verbatim}
    if len(cad) <= 10:

\item Inicia un bucle for para iterar sobre los
caracteres de la cadena invertida.
\begin{verbatim}
    for i in cadc:

\item Crea un objeto de la clase Label que
 muestra información sobre el carácter actual en
  la ventana.
\begin{verbatim}
    u = Label(ventana, text="CARACTER A
    VERIFICAR \n\n "+
    cad[0:len(cad)-cont]).place(x=10,
     y=100)

\item Actualiza la ventana y espera medio
 segundo para proporcionar una animación.
\begin{verbatim}
    ventana.update()
    time.sleep(.5)

\item Incrementa el contador en uno.
\begin{verbatim}
    cont = cont + 1

\item Comprueba si el carácter actual en cadc es "1".
\begin{verbatim}
    if cadc[cuenta] == "1":

\item Realiza operaciones si se cumple la
condición anterior:
incrementa cuenta,
agrega "X" a la pila, imprime la pila y
actualiza aux1 y e.
```

```
\begin{verbatim}
    cuenta = cuenta + 1
    p.push("X")
    print(p.arreglo)
    archivo.write("\n"+str(p.arreglo))
    aux1 = aux1 + 1
    e = "c"

\item  Crea un rectángulo naranja en el lienzo,
espera medio segundo y continua con la
 siguiente iteración.
\begin{verbatim}
    canv.create_rectangle(300, 500-conterr*25,
     400, 550-conterr*25, width=1, fill='orange',
      outline='white')
    conterr = conterr+1
    time.sleep(.5)
    conterr = conterr + 1
    continue

\item  Este bloque de código verifica si el carácter
actual es "0" y la variable e es "a".
En ese caso, se realiza una serie de acciones,
 como hacer push en la pila, imprimir la pila,
 escribir en el archivo, crear un rectángulo azul
  en el lienzo,
 y hacer una pausa antes de continuar con la
 siguiente iteración.
\begin{verbatim}
    if i == "0" and e == "a":
    p.push(i)
    print(p.arreglo)
    archivo.write("\n"+str(p.arreglo))
    canv.create_rectangle(300, 500-conta*25,
    400, 550-conta*25, width=1, fill='blue',
     outline='white')
    conta = conta+1
    time.sleep(.5)
    conta = conta + 1
```

```
    continue

\item Esta parte del código maneja el
caso en que la longitud de la cadena es mayor que 10
\begin{verbatim}
    else:
        for i in cadc:

\item Estos bloques evalúan el estado final de la
 pila después de procesar la cadena. Si la pila
 está vacía, se considera que la cadena es aceptada
 y se muestra un mensaje en la consola, se escribe
 en el archivo y se muestra una imagen verde en
  la interfaz gráfica. En caso contrario, se
   considera que la cadena es inválida, se
   realiza la acción
correspondiente y se muestra una imagen roja
en la interfaz gráfica.
\begin{verbatim}
    a = len(p.arreglo)
    if a == 0:
        print("CADENA ACEPTADA")
        archivo.write("\n\nCADENA ACEPTADA")
        imagen_aceptada_tk = ImageTk.PhotoImage
        (imagen_aceptada)
        o = Label(ventana, image=imagen_aceptada_tk)
        o.image = imagen_aceptada_tk
        o.place(x=500, y=300)
        canv.create_rectangle(300, 50, 400, 550,
         width=0, fill='green')
    else:
        print("CADENA INVALIDA")
        archivo.write("\n\nCADENA INVALIDA")
        imagen_noaceptada_tk = ImageTk.PhotoImage
        (imagen_noaceptada)
        o = Label(ventana, image=imagen_noaceptada_tk)
        o.image = imagen_noaceptada_tk
        o.place(x=500, y=300)
        canv.create_rectangle(300, 50, 400, 550,
```

```
        width=0, fill='red')

\item Define una función llamada limpiar etiquetas que
 elimina todos los widgets de tipo
 Label en la ventana y borra todo el contenido del lienzo.
\begin{verbatim}
    def limpiar_etiquetas(ventana):
        for widget in ventana.winfo_children():
            if isinstance(widget, Label):
                widget.destroy()
        canv.delete(ALL)

\item Cierra el archivo abierto anteriormente.
\begin{verbatim}
    archivo.close()

\item Pregunta al usuario si desea ejecutar el programa
 nuevamente. Si la respuesta no es 'S' (mayúscula),
  limpia la interfaz gráfica y termina
 el programa. Si la respuesta es 'S',
  limpia la interfaz gráfica.
\begin{verbatim}
    respuesta = input("¿DESEA EJECUTAR NUEVAMENTE
    (S/N): ")
    if respuesta.upper() != 'S':
        limpiar_etiquetas(ventana)
        ventana.destroy()
        sys.exit()
    else:
         limpiar_etiquetas(ventana)

\item  Define la función principal main que crea una
ventana
 y un lienzo, establece el título y las dimensiones,
  y ejecuta
indefinidamente la función ejecutar programa dentro
 de un bucle while.
\begin{verbatim}
    def main():
```

```
    ventana = Tk()
    ventana.title("Lenguaje libre de contexto
    {0^n 1^n | n >= 1}")
    canv = Canvas(ventana, width=800, height=600)
    ventana.geometry("800x600")
    canv.pack()

    while True:
        ejecutar_programa(ventana, canv)

\item Verifica si el script está siendo ejecutado
como un programa independiente y,
en ese caso, llama a la función principal main.
\begin{verbatim}
    if __name__ == "__main__":
    main()

\end{itemize}

}

\newpage
\section{Ejecución}
{\Large
\begin{itemize}
\item Compilacion
\\
\newline
\begin{figure}[h]
\centering
\includegraphics[scale=.7]{img1.png}
\end{figure}
\newline
\item Modo Manual
\\
\newline
Prueba con modo manual, se selecciona la opción 1
 y se
ingresa la cadena manualmente, en este caso se
```

```
ingresó la cadena
11110000.
\\
\\
\begin{figure}[h]
\centering
\includegraphics[scale=.4]{img2.png}
\end{figure}
\\
\\
La cadena respeta las condiciones de la pila,
 por tanto
se valida y se colorea la pila de color verde, además
 de que se
muestra una paloma dando a entender que es válida
la cadena.
\\
\\
\begin{figure}[h]
    \centering
    \includegraphics[scale=.4]{img3.png}
    \end{figure}
\\
\newpage
Al término de la ejecución se muestra un mensaje el cual
permite volver a realizar una ejecución más del autómata,
con tan solo
escribir la letra "S", al ingresar la "S" se limpian los
labels y el lienzo.
\\
\begin{figure}[h]
\centering
\includegraphics[scale=.4]{img4.png}
\end{figure}
\\
\\
Ahora ingresaremos una cadena de más de 10 carácteres,
al hacerlo podemos
apreciar que no se va mostrando el proceso de la pila,
```

```
 si no que solo
la descripción y se pinta del color correspondiente el
lienzo. En este caso
es de color rojo porque la cadena no es válida.
\\
\\
\begin{figure}[h]
\centering
\includegraphics[scale=.4]{img5.png}
\end{figure}
\newpage
Se genera un archivo llamado "Descripción
Instantánea.txt", el cual
como se menciona contiene la descripción
instantánea de la cadena.
\\
\begin{figure}[h]
\centering
\includegraphics[scale=.7]{img6.png}
\end{figure}
\\
\newpage
\item Modo Autómatico
\\
\newline
La segunda opción es el Modo Autómatico y
consiste en que el mismo
programa ingresará una cadena aleatoria a evaluar.
\begin{figure}[h]
\centering
\includegraphics[scale=.4]{img7.png}
\end{figure}
\\
En este caso el programa ingresó la cadena
1111011011100 y el txt también nos muestra la
 cadena y si
es válida o no.
\begin{figure}[h]
    \centering
```

```
    \includegraphics[scale=.7]{img8.png}
    \end{figure}
\\
\newpage
La cadena se evalúa y al no cumplir los requisitos,
es decir no es
válida esta se pinta la pila de rojo.
\\
\begin{figure}[h]
    \centering
    \includegraphics[scale=.4]{img9.png}
    \end{figure}
Al final si ya no se requiere realizar otra
 ejecución, solo se
selecciona "N" y el programa termina.
\begin{figure}[h]
    \centering
    \includegraphics[scale=.7]{img10.png}
    \end{figure}
\\
\end{itemize}
}
\newpage
\section{Código en Latex}
{\Large
\begin{verbatim}
}
\newpage
\section{Conclusión}
{\Large

}
\end{document}
}
```

6 Conclusión

El código implementa un verificador para el lenguaje libre de contexto.

La estructura de pila es esencial para manejar las reglas, este código simula un autómata de pila, un tipo de autómata que utiliza una pila para almacenar información y realizar transiciones basadas en la entrada y el contenido de la pila.

El código utiliza imágenes (aceptada y no aceptada) para indicar visualmente si la cadena ha sido aceptada o no y se utiliza la manipulación de archivos para guardar información sobre las cadenas ingresadas o generadas y los pasos de evaluación del autómata.