

Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Lenguajes Formales y de Programación

Inga. Zulma Aguirre

Aux. Luis Yela



Elder Anibal Pum Rojas

201700761

Guatemala, 2 de Mayo del 2020

## DICCIONARIO DE CLASES UTILIZADAS

Las clases utilizadas dentro del archivo principal.py fueron las siguientes:

- `BorrarPantalla()`
- `menu()`
- `menuAFD()`
- `menuGramatica()`
- `crearGramaticaDelAFDGenerado(gramática, transicionPrimario, estadoAceptacion)`
- `creacionAFD()`
- `creacionGramatica()`
- `menuCadena()`
- `validarCadena()`
- `rutaAFD()`
- `expansionGramatica()`
- `menuReportes()`
- `reporteDetalle()`
- `generarPDF()`
- `menuCarga()`
- `cargarAFD()`
- `cargaGramatica()`
- `menuGuardar()`
- `guardarArchivo()`

Acá se pueden apreciar imágenes de los diferentes métodos:

```
principal.py x afd.py x
1 import os
2 import time
3 import sys
4 from graphviz import Digraph
5 from afd import AFD
6 from reportlab.pdfgen import canvas
7 from PIL import Image
8 from reportlab.lib.pagesizes import A4
9 automataGeneral = dict()
10 cadenasPrincipal = dict()
11 cadenasSecundario = dict()
12 ruta = os.path.join(os.sep, "Users", "Elder", "Desktop", "Repositorio", "AFD")
13 rutaGuardar = os.path.join(os.sep, "Users", "Elder", "Desktop", "Repositorio", "Guardados", ".")
14 print(" ")
15 print("-----Datos del estudiante-----")
16 print("#")
17 print("# Lenguajes formales y de programacion #")
18 print("# Seccion: B- #")
19 print("# Carne: 201700761 #")
20 print("#")
21 print("-----")
22 print(" ")
23 input("Presione enter para continuar")
24
25
26 def BorrarPantalla():...
31
32 def menu():...
77
```

```
principal.py x afd.py x
31
32 def menu():...
77
78 def menuAFD():...
108
109 def menuGramatica():...
140
141 def crearGramaticaDelAFDGenerado(gramatica, transicionPrimario, estadosAceptacion):...
155
156 def creacionAFD():...
387
388 def creacionGramatica():...
571
572 def menuCadenas():...
603
604 def validarCadena():...
674
675 def rutaAFD():...
747
748 def expansionGramatica():...
820
821 def menuReportes():...
849
850 def reporteDetalle():...
896
897 def generarPDF():...
977
978 def menuCarga():...
1006
```

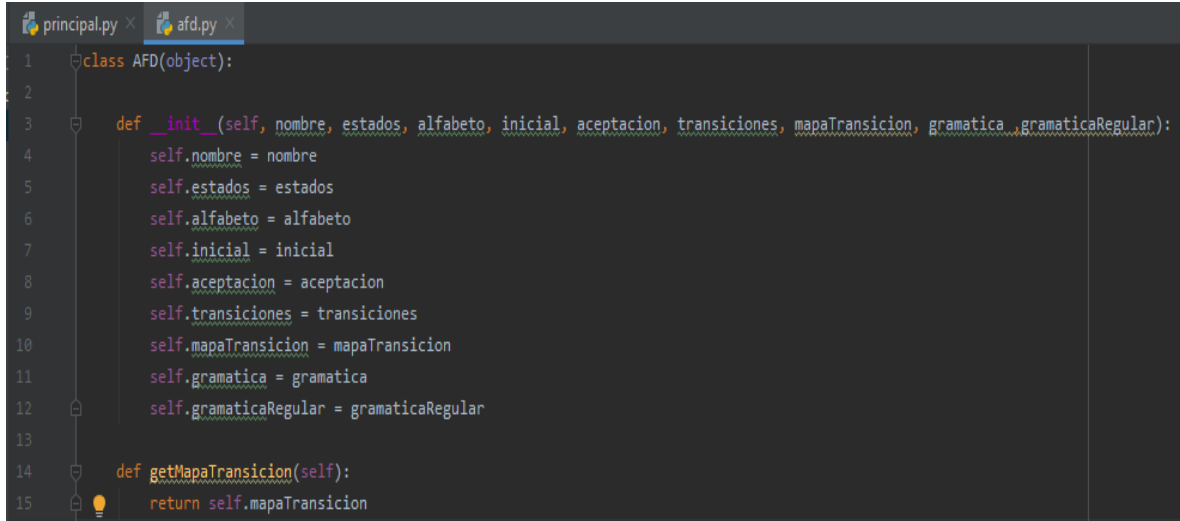
```
principal.py x afd.py x
572 def menuCadenas():...
603
604 def validarCadena():...
674
675 def rutaAFD():...
747
748 def expansionGramatica():...
820
821 def menuReportes():...
849
850 def reporteDetalle():...
896
897 def generarPDF():...
977
978 def menuCarga():...
1006
1007 def cargarAFD():...
1100
1101 def cargaGramatica():...
1233
1234 def menuGuardar():...
1260
1261 def guardarArchivo():...
1346 menu()
```

- **BorrarPantalla():** Se encarga de limpiar la pantalla de la consola independientemente del sistema operativo donde se ejecute el proyecto.
- **menu():** Contiene el menú interactivo donde el usuario se desplazará para navegar en el proyecto. Contiene a “Crear AFD”, “Crear Gramática”, “Evaluar Cadenas”, “Generar reportes”, “Cargar Archivo”, “Guardar Archivo” y “Salir”.
- **menuAFD():** Menú interactivo para poder desplazarse sobre el menú de construcción del AFD.
- **menuGramatica:** Menú interactivo para poder desplazarse sobre el menú de construcción de las gramáticas.
- **crearGramaticaDelAFDGenerado(gramática, transicionPrimario, estadoAceptacion):** Básicamente recibe una lista gramática, un diccionario primario que contiene como llave el nombre el estado inicial y como valores tanto la transición como el estado final, además del estado de aceptación. Esto devuelve la gramática ya generada.

- **creacionAFD():** Esta clase sirve para crear ya directamente el AFD. Solicita cada parte del autómata, crea la imagen en graphviz y encima genera su propia gramática regular.
- **creacionGramatica():** Esta clase sirve para crear ya directamente las gramáticas. Además de que puede resolver gramáticas recursivas por la izquierda. De una vez genera el AFD en graphviz para su posterior uso.
- **menuCadenas():** Menú interactivo para la evaluación de las cadenas.
- **validarCadena():** Valida que la cadena ingresada para el AFD/Gramática sea válida o no, mostrando un único mensaje.
- **rutaAFD():** Misma función que validarCadena() solamente que muestra el recorrido dentro del AFD para una mejor comprensión.
- **expansionGramatica():** Igual que rutaAFD() solamente que este muestra el recorrido pero en la gramática generada e indica si la cadena es válida o inválida.
- **menuReportes():** Menú interactivo para la creación de diferentes reportes.
- **reporteDetalle():** Método que se utiliza para saber cada una de las partes que compone un AFD o gramática con únicamente ingresar el nombre de este.
- **generarPDF():** Método que genera un archivo de extensión “.pdf” con el nombre del AFD/Gramática ingresado. Muestra cada parte del AFD y gramática generada, así como cadenas evaluadas, saber si estas son válidas e inválidas, además de mostrar la imagen del AFD.
- **menuCarga():** Menú interactivo para la carga de archivos con extensión “.afd” o “.grm”.
- **cargarAFD():** Método que sirve para cargar un archivo con extensión “.afd” y guardarlo dentro del sistema para su posterior uso.
- **cargarGramatica():** Igual que cargarAFD() solo que este carga un archivo con extensión “.grm” para su posterior uso.
- **menuGuardar():** Menú interactivo para manejar el guardado de archivos.
- **guardarArchivo():** Método que sirve para guardar un archivo con extensión “.afd” o “.grm” con únicamente escribir su nombre.



Para el archivo afd.py se obtuvo lo siguiente:



```
1 class AFD(object):
2
3     def __init__(self, nombre, estados, alfabeto, inicial, aceptacion, transiciones, mapaTransicion, gramatica, gramaticaRegular):
4         self.nombre = nombre
5         self.estados = estados
6         self.alfabeto = alfabeto
7         self.inicial = inicial
8         self.aceptacion = aceptacion
9         self.transiciones = transiciones
10        self.mapaTransicion = mapaTransicion
11        self.gramatica = gramatica
12        self.gramaticaRegular = gramaticaRegular
13
14    def getMapaTransicion(self):
15        return self.mapaTransicion
```

- **\_\_init\_\_():** El constructor de la clase AFD del archivo afd. Nos sirve para guardar todas las cosas que se nos piden tanto en el módulo de creación de AFD como en gramática. Incluso sirve para realizar las validaciones de las cadenas.
- **getMapaTransicion(self):** Devuelve el diccionario que se utiliza para la validación de las cadenas.

## DICCIONARIO DE CLASES UTILIZADAS PROYECTO #2

- menuPrincipal()
- menuProyecto2()
- crearGramaticaT2()
- generarAP()
- validarCadenaAP()

Acá se pueden apreciar imágenes de los métodos antes mencionados:

```
1354
1355 def menuPrincipal():...
1381
1382 def menuProyecto2():...
1412
1413 def crearGramaticaT2():...
1741
1742 def generarAP():...
1805
1806 def validarCadenaAP():...
```

- **menuPrincipal():** Es un menú creado exclusivamente como puente entre las funcionalidades desarrolladas en el proyecto #1 (descritas anteriormente) y las nuevas funcionalidades desarrolladas en el proyecto 2.
- **menuProyecto2():** Es un menú desarrollado con el fin de poder navegar entre las nuevas funcionalidades del proyecto 2, entre ellas están “Crear / Modificar Gramática”, “Generar AP” y “Validar Cadenas”.
- **crearGramaticaT2():** La principal función es crear una gramática de tipo 2 desde el inicio, solicitando un nombre, luego los terminales, luego los no terminales y sus producciones. Como bonus cabe resaltar que ahora incluye un proceso de borrado de producciones por si alguna no coincide con la que queríamos ingresar. Además, permite ingresar el no terminal inicial. Si se repite el proceso y se ingresa un nombre ya registrado, se inicia el proceso de modificación.
- **generarAP():** La única utilidad de este módulo es generar el archivo .png del autómatas de pila con los datos ingresados. Si en caso no existiera, simplemente muestra error y se sale del módulo.
- **validarCadenaAP():** Igualmente que el módulo anterior, valida una cadena para una gramática ya creada. Si esta cadena es válida, entonces genera dos cosas, genera el reporte en archivo .csv donde describe el movimiento de la pila hasta llegar a la aceptación y también genera el árbol sintáctico para esa expresión.



Para el archivo ap.py (utilizado para el proyecto #2) se tiene lo siguiente:

```
principal.py x ap.py x afd.py x
1 class AP(object):
2
3     def __init__(self, nombre, terminales, noTerminales, producciones, inicial):
4         self.nombre = nombre
5         self.terminales = terminales
6         self.noTerminales = noTerminales
7         self.producciones = producciones
8         self.inicial = inicial
```

- **\_\_init\_\_():** El constructor de la clase AP del archivo ap. Nos sirve para guardar todas las cosas que se nos piden tanto en el módulo de creación de la gramática de tipo 2 y es el eje de los datos para los demás módulos, ya que utilizando esta clase es como se obtienen los datos.

## HERRAMIENTAS A UTILIZAR

- Python V2.X ó V3.X, preferible la versión 3 en adelante, que servirá para realizar la práctica correspondiente.
- El lenguaje utilizado para desarrollar dicho proyecto es Python.
- Se utilizó como IDE Pycharm para trabajar de forma óptima Python.
- Se utilizó de forma extra el programa Graphviz para poder obtener los grafos de los AFD.
- Se utilizó en su mayoría paradigma orientado a objetos para la realización del proyecto.

## PLATAFORMA DE EJECUCIÓN

