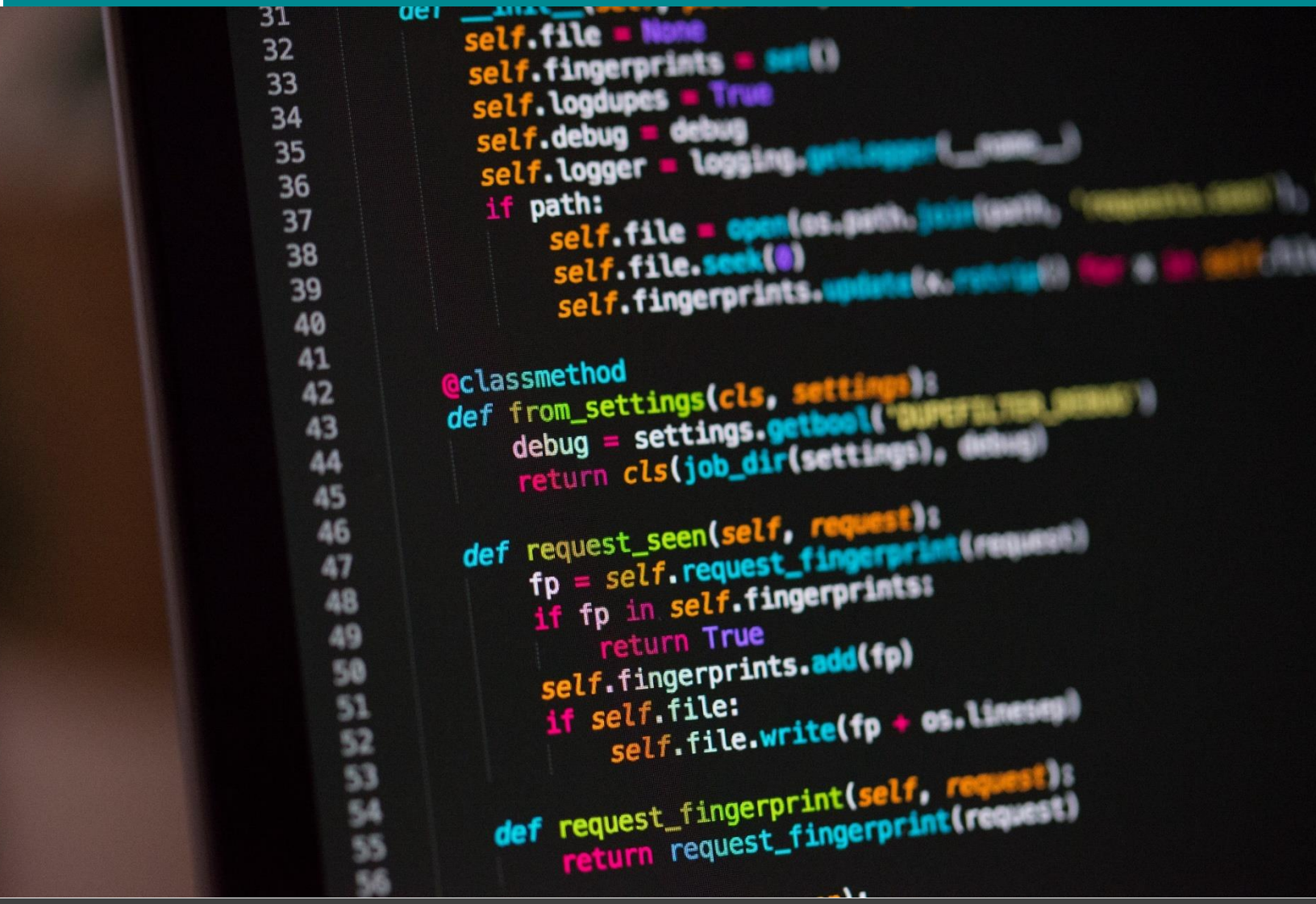
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
LENGUAJES FORMALES Y DE PROGRAMACIÓN
Sección B+



MANUAL TEORICO

PROYECTO 1

MANUAL TECNICO

DESCRIPCION

Este programa analiza una entrada de texto plano, su principal función es analizar lexica y sintácticamente un archivo de entrada para un menú de un restaurante, y un archivo de factura, en el cual generara un archivo html para los lexemas aceptados y otro archivo si existen errores.

FUNCIÓN MENÚ

Def main: en esta función llamamos a la presentacion() y dentro de un bucle while podremos seguir seleccionando las opciones hasta seleccionar salir. Utilizamos un paradigma de repetición al igual que estructurada.

```
def main():
    print('-------------------- Proyecto 1 --------------------')
    Presentacion()
    while True:
        opciones()
        print(">> Ingrese una Opcion:")
        print(">>", end="")
        entrada = input().lower()
        seleccion(entrada)
```

Def opciones: esta muestra las opciones que tenemos en el programa, solamente imprime.

```
def opciones():
    print('1. Cargar Menú')
    print('2. Cargar Orden')
    print('3. Generar Menú')
    print('4. Generar Factura')
    print('5. Generar Árbol')
    print('6. Salir')
```

Def Presentacion(): únicamente mostrara los datos del alumno, y de la clase

```
def Presentacion():
    print('-------------------------------------------------------\n'
          '| Lenguajes Formales y de Programacion Seccion B+    |\n'
          '| Jhonathan Daniel Tocay Cotzojay Carné: 201801268   |\n'
          '-------------------------------------------------------')
```

Def selección: en esta función se utilizo el paradigma de selección, dado que utilizamos un if, el cual nos ayudara a llamar a otras funciones según sea la selección

```
def seleccion(entrada):
    en = entrada
    if en == "1":
        print(" -------------------- Cargar Menu -------------------- ")
        openFile()

    elif en == "2":
        print(" -------------------- Cargar Orden -------------------- ")
        openFile_Factura()

    elif en == "3":
        print(" -------------------- Generar Menú --------------------")
        if Base !=[]:
            print(" ¿Desea Poner un Limite en los precios? Si/No")
            print(">>", end="")
            resp = input().lower()
            if resp == "si":
                try:
                    p_limite = float(input("Ingrese El Precio Limite:"))
                    Generar_Menu.reporte_Filtro(Base, p_limite)
                    os.system('Menu.html')
                except ValueError:
                    print("no ha ingresado un numero valido")
```

```
        elif resp == "no":
            Generar_Menu.reporte(Base)
            os.system('Menu.html')
        else:
            print(" ---------- Esta opcion No existe ----------")
    else:
        print("Ingrese un Archivo Valido en Cargar Menu para Generar Menu")
        print("------------------------------------------------------------")

elif en == "4":
    print(" -------------------- Generar Factura -------------------- ")
    if Datos_Facturas != []:
        Generar_Factura.reporte(Base, Datos_Facturas)
        os.system('Factura.html')
    else:
        print("Ingrese un archivo Valido en Cargar Orden para poder Generar la Factura")
        print("------------------------------------------------------------")
```

ABRIR ARCHIVO

La función openFile(): utiliza una librería llamada tkinter para generar una ventana emergente, en la cual se puede seleccionar un archivo de texto, para su posterior análisis, con la propiedad de el filedialog, **askopenfilename()** obtenemos la ruta del archivo. Agregaremos filetypes para que solo acepte archivos .lfp

```
def openFile():
    filepath = filedialog.askopenfilename(filetypes=(("Archivos de Texto","*.lfp"),("Todos Los Archivos","*.*")))

    if filepath == '':
        print("no se ha seleccionado un archivo")
        print("------------------------------------------------")
    else:
        archi(filepath)
```

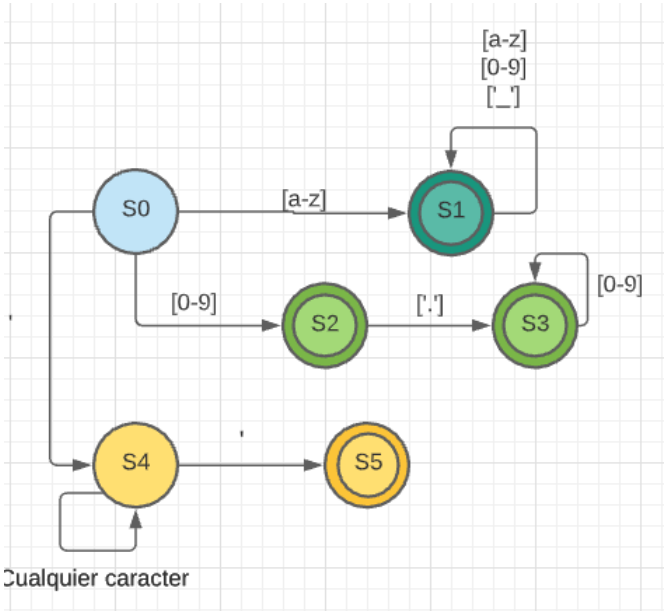
Def archi: en esta función se verifica si el archivo es .lfp por lo cual se utiliza el paradigma estructurado, específicamente uno condicional.

```
def archi(filepath):
    archivo = open(filepath)
    file = archivo.read()
    # ---------------------------- Datos Del Archivo -----------------------------
    #-----------------------------------------------------------------------------
    nombre_archivo = os.path.basename(filepath)
    extension = nombre_archivo.split('.')

    if (extension[-1].lower() == "lfp"):
        try:
            # ------------------ Analizador Lexico Menu
            Analizador.analizador(file, lexema, lexema_error, lexema_scan)
            # ------------------ Analizador Sintactico Menu
            sintactico.analizador_sintactico(lexema_scan, Base, lexema_error, Lexemas_aceptados)
            #Impresion()
        except:
            print("Opss!", sys.exc_info()[0], "ocured.")
    else:
        print("La extension del Archivo no es LFP")
```

ANALIZADOR LEXICO

TEORICO: el siguiente autómata mostrara gráficamente como se empleara la construcción de patrones para los lexemas.



PRACTICO: el siguiente autómata mostrara gráficamente como se empleara la construcción de patrones para los lexemas.

El archivo se recorrerá carácter por carácter para armar los patrones propuestas en el enunciado, claramente se utilizara el **paradigma estructurado.**

```
while i < len(cadena):
    if estado == 0:
        if cadena[i] == '[':
            columna += 1
            lexema.append({"fila": fila, "columna": columna, "token": "tk_Abrir_Corchete",
"valor": "["})
            lexema_scan.append({"fila": fila, "columna": columna, "token": "tk_Abrir_Corchete",
"valor": "["})
        elif cadena[i] == ']':
            columna += 1
            lexema.append({"fila": fila, "columna": columna, "token": "tk_Cerrar_Corchete",
"valor": "]"})
            lexema_scan.append({"fila": fila, "columna": columna, "token": "tk_Cerrar_Corchete",
"valor": "]"})
        elif cadena[i] == ':':
            columna += 1
            lexema.append({"fila": fila, "columna": columna, "token": "tk_Dos_Puntos", "valor":
":"})
            lexema_scan.append({"fila": fila, "columna": columna, "token": "tk_Dos_Puntos",
"valor": ":"})
        elif cadena[i] == "'":
            columna += 1
            lexema.append({"fila": fila, "columna": columna, "token": "tk_comilla", "valor":
"'"})
            lexema_scan.append({"fila": fila, "columna": columna, "token": "tk_comilla", "valor":
"'"})
            estado = 4
        elif cadena[i] == "=":
            columna += 1
            lexema.append({"fila": fila, "columna": columna, "token": "tk_igual", "valor": "="})
            lexema_scan.append({"fila": fila, "columna": columna, "token": "tk_igual", "valor":
"="})
        elif cadena[i] == ";":
            columna += 1
            lexema.append({"fila": fila, "columna": columna, "token": "tk_Punto_Coma", "valor":
";"})
            lexema_scan.append({"fila": fila, "columna": columna, "token": "tk_Punto_Coma",
"valor": ";"})
        elif (ord(cadena[i]) >= 65 and ord(cadena[i]) <= 90) or (ord(cadena[i]) >= 97 and
ord(cadena[i]) <= 122):
            auxReservada = auxReservada + cadena[i]
            estado = 1
        elif ord(cadena[i]) >= 48 and ord(cadena[i]) <= 57:
            num = num + cadena[i]
            estado = 2
        elif ord(cadena[i]) == 32 or cadena[i]=='':
            columna += 1
        elif cadena[i] == '\t':
```

```
            columna += 1
        elif cadena[i] == '\n':
            fila = fila + 1
        else:
            columna += 1
            lexema_error.append({"fila": fila, "columna": columna, "valor": cadena[i],
"descripcion": "[Error Lexico] Caracter Desconocido"})
            lexema_scan.append({"fila": fila, "columna": columna, "token": "tk_desconocido",
"valor": cadena[i]})
```

El estado para encontrar los identificadores será

```
# ------------------------------------ Identificador ------------------------------------
--------------------
    elif (estado == 1):
        if (ord(cadena[i]) >= 65 and ord(cadena[i]) <= 90) or (ord(cadena[i]) >= 97 and
ord(cadena[i]) <= 122) or (ord(cadena[i]) >= 48 and ord(cadena[i]) <= 57) or (cadena[i] == "_"):
            auxReservada = auxReservada + cadena[i]
            estado = 1
        else:
            columna += 1
            if (auxReservada.lower() == 'restaurante'):
                lexema.append({"fila": fila, "columna": columna, "token": "tk_" + auxReservada,
"valor": auxReservada})
                lexema_scan.append({"fila": fila, "columna": columna, "token": "tk_" +
auxReservada, "valor": auxReservada})
                auxReservada = ""
                estado = 0
                i = i-1
            else:
                lexema.append({"fila": fila, "columna": columna, "token": "tk_identificador",
"valor": auxReservada})
                lexema_scan.append({"fila": fila, "columna": columna, "token":
"tk_identificador", "valor": auxReservada})
                auxReservada = ""
                estado = 0
                i = i-1
```

Para los números

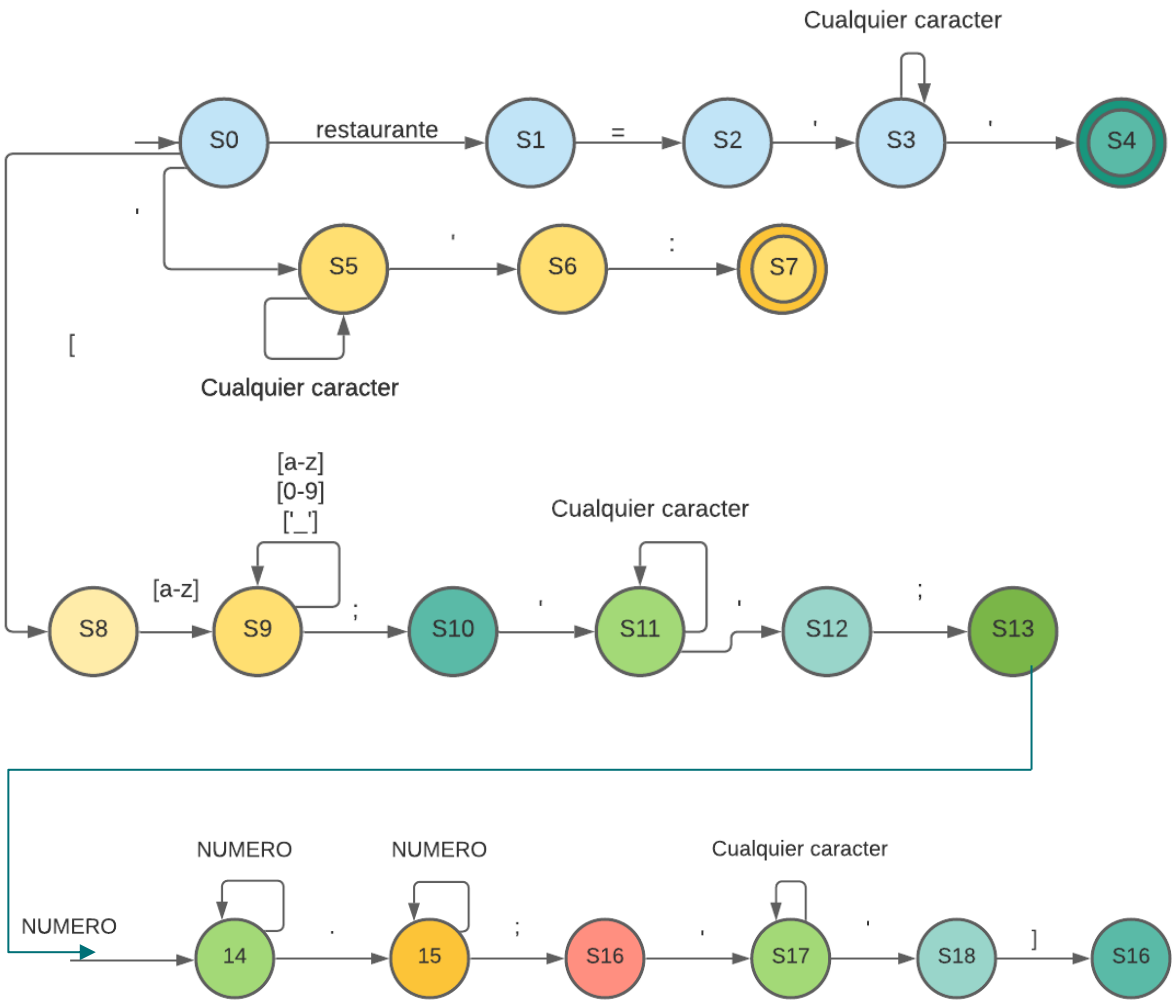
```
# ------------------------------------------- Numero ------------------------------
----------------------------------
    elif(estado == 2):
        if ord(cadena[i]) >= 48 and ord(cadena[i]) <= 57:
            num = num + cadena[i]
            estado = 2
        elif cadena[i] == '.' or ord(cadena[i]) == 46:
            num = num + cadena[i]
            estado = 3
        else:
            columna += 1
            lexema.append({"fila": fila, "columna": columna, "token": "tk_numero",
"valor": num})
            lexema_scan.append({"fila": fila, "columna": columna, "token":
"tk_numero", "valor": num})
            num = ""
            i = i - 1
            estado = 0
# ---------------------------------------  Numero Decimal  ------------------------
----------------------------------
    elif estado == 3:
        if ord(cadena[i]) >= 48 and ord(cadena[i]) <= 57:
            num = num + cadena[i]
        else:
            columna += 1
            lexema.append({"fila": fila, "columna": columna, "token": "tk_numero",
"valor": num})
            lexema_scan.append({"fila": fila, "columna": columna, "token":
"tk_numero", "valor": num})
            num = ""
            i = i-1
            estado = 0
```

Para las cadenas

```
# ------------------------------------------- Cadena ------------------------------------------
---
    elif estado == 4:
        if cadena[i] == "'":
            columna += 1
            lexema.append({"fila": fila, "columna": columna, "token": "tk_cadena", "valor":
cad2})
            lexema_scan.append({"fila": fila, "columna": columna, "token": "tk_cadena", "valor":
cad2})
            columna += 1
            lexema.append({"fila": fila, "columna": columna, "token": "tk_comilla", "valor":
"'"})
            lexema_scan.append({"fila": fila, "columna": columna, "token": "tk_comilla", "valor":
"'"})
            cad2 = ""
            estado = 0
        else:
            cad2 = cad2 + cadena[i]
```

ANALIZADOR SINTACTICO

TEORICO: el siguiente autómata mostrara gráficamente como se empleara



Aquí se programara recorrera lexema por lexema comparando los tokens para verificar el orden de este archivo .

```
while i < len(lexema_scan):
# ------------------------------------ Estado Inicial ------------------------------------
----
    if estado == 0:
        if(lexema_scan[i]['token'] == 'tk_restaurante'):
            if cont_res == 1:
                Lexemas_aceptados.append(lexema_scan[i])
                cont_res += 1
                estado = 1
            else:
                lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'], "descripcion": "[Error Sintactico]
Se repite la palabra reservada Restaurante"})
                estado = 0
        elif(lexema_scan[i]['token'] == "tk_comilla"):
            estado = 5

        elif(lexema_scan[i]['token'] == "tk_Abrir_Corchete"):
            estado = 8
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'], "descripcion": "[Error Sintactico
Desconocido]"})
            estado = 0
# ------------------------------------------------------------------------------------------
--
    # ------------------------------ Restaurante -------------------------------------
    elif estado == 1:
        if(lexema_scan[i]['token'] == 'tk_igual'):
            estado = 2
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'], "descripcion": "[Error Sintactico]
se esperaba un signo ="})
            estado = 0

    elif estado == 2:
        if (lexema_scan[i]['token'] == "tk_comilla"):
            estado = 3
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'], "descripcion": "[Error Sintactico]
se esperaba un signo '"})
            estado = 0

    elif estado == 3:
        if (lexema_scan[i]['token'] == "tk_cadena"):
            nombre_del_Restaurante = lexema_scan[i]['valor']
            lexema_scan[i]['token'] = 'tk_Nombre_Restaurante'
            Lexemas_aceptados.append(lexema_scan[i])
            estado = 4
        else:
```

```
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'], "descripcion": "[Error Sintactico]
Se Esperaba el Nombre del Restaurante"})
            estado = 0

    elif estado == 4:
        if (lexema_scan[i]['token'] == "tk_comilla"):
            estado = 0
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'], "descripcion": "[Error Sintactico]
se esperaba un signo '"})
            estado = 0
# ------------------------------------------------------------------------------------------
--
    # ------------------------------ Seccion ---------------------------------------------
    elif estado == 5:
        if (lexema_scan[i]['token'] == "tk_cadena"):
            Nombre_Seccion.append(lexema_scan[i]['valor'])
            auxNS = lexema_scan[i]['valor']
            lexema_scan[i]['token'] = 'tk_Seccion'
            Lexemas_aceptados.append(lexema_scan[i])
            estado = 6
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'], "descripcion": "[Error Sintactico]
Se esperaba el Nombre de una Seccion'"})
            estado = 0

    elif estado == 6:
        if (lexema_scan[i]['token'] == "tk_comilla"):
            estado = 7
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'], "descripcion": "[Error Sintactico]
se esperaba un signo '"})
            estado = 0

    elif estado == 7:
        if (lexema_scan[i]['token'] == "tk_Dos_Puntos"):
            estado = 0
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'], "descripcion": "[Error Sintactico]
Se esperaba 2 puntos"})
            estado = 0
    # ------------------------------ Producto ---------------------------------------------
    elif estado == 8:
        if (lexema_scan[i]['token'] == "tk_identificador"):
            auxID = lexema_scan[i]['valor']
            Lexemas_aceptados.append(lexema_scan[i])
            estado = 9
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'], "descripcion": "[Error Sintactico]
se esperaba un identificador"})
            estado = 0
```

```
    elif estado == 9:
        if (lexema_scan[i]['token'] == "tk_Punto_Coma"):
            estado = 10
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'],"valor": lexema_scan[i]['valor'],"descripcion": "[Error Sintactico] se
esperaba un punto y coma"})
            estado = 0

    elif estado == 10:
        if (lexema_scan[i]['token'] == "tk_comilla"):
            estado = 11
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'],"descripcion": "[Error Sintactico] se
esperaba un signo '"})
            estado = 0

    elif estado == 11:
        if (lexema_scan[i]['token'] == "tk_cadena"):
            lexema_scan[i]['token'] = 'tk_Nombre_identificador'
            auxNombre_ID = lexema_scan[i]['valor']
            Lexemas_aceptados.append(lexema_scan[i])
            estado = 12
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'],"valor": lexema_scan[i]['valor'],"descripcion": "[Error Sintactico] se
esperaba el nombre del identificador"})
            estado = 0

    elif estado == 12:
        if (lexema_scan[i]['token'] == "tk_comilla"):
            estado = 13
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'], "descripcion": "[Error Sintactico]
se esperaba un signo '"})
            estado = 0

    elif estado == 13:
        if (lexema_scan[i]['token'] == "tk_Punto_Coma"):
            estado = 14
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'], "descripcion": "[Error Sintactico]
se esperaba un punto y coma "})
            estado = 0

    elif estado == 14:
        if (lexema_scan[i]['token'] == "tk_numero"):
            auxPrecio = lexema_scan[i]['valor']
            Lexemas_aceptados.append(lexema_scan[i])
            estado = 15
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'], "descripcion": "[Error Sintactico]
```

```
se esperaba un Numero"})
            estado = 0

    elif estado == 15:
        if (lexema_scan[i]['token'] == "tk_Punto_Coma"):
            estado = 16
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'], "descripcion": "[Error Sintactico]
se esperaba un punto y coma"})
            estado = 0

    elif estado == 16:
        if (lexema_scan[i]['token'] == "tk_comilla"):
            estado = 17
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'], "descripcion": "[Error Sintactico]
se esperaba un signo '"})
            estado = 0

    elif estado == 17:
        if (lexema_scan[i]['token'] == "tk_cadena"):
            lexema_scan[i]['token'] = 'tk_Descripcion'
            auxDescripcion = lexema_scan[i]['valor']
            Lexemas_aceptados.append(lexema_scan[i])
            estado = 18
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'], "descripcion": "[Error Sintactico]
se esperaba una Descripcion"})
            estado = 0

    elif estado == 18:
        if (lexema_scan[i]['token'] == "tk_comilla"):
            estado = 19
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'], "descripcion": "[Error Sintactico]
se esperaba un signo '"})
            estado = 0

    elif estado == 19:
        if (lexema_scan[i]['token'] == "tk_Cerrar_Corchete"):
            Lista_Seccion.append({"seccion": auxNS,"id": auxID, "nombre": auxNombre_ID, "numero":
float(auxPrecio), "descripcion": auxDescripcion})
            auxID = ''
            auxNombre_ID = ''
            auxPrecio = ''
            auxDescripcion = ''
            estado = 0
        else:
            lexema_error.append({"fila": lexema_scan[i]['fila'], "columna":
lexema_scan[i]['columna'], "valor": lexema_scan[i]['valor'], "descripcion": "[Error Sintactico]
se esperaba ]"})
            estado = 0
    i += 1
```

GENERAR ARBOL

Para generar esta Grafica se utilizo la librería Digraph, y graphviz y dándole un nobre a los nodos he escribiéndole los datos especificados, además de las aristas o transiciones.

```
from graphviz import Digraph
import operator
def un(Base):
    # ------------------------------------------------
    nombre_del_Restaurante = Base[0]
    Nombre_Seccion = Base[1]
    Lista_Seccion = Base[2]
    # ------------------------------------------------
    #print('-----------------Generar Grafica-------------------------')
    dot = Digraph(comment='Grafica Restaurante')
    dot  # doctest: +ELLIPSIS
    #print(nombre_del_Restaurante)
    dot.node('A', str(nombre_del_Restaurante))
    # ----------------------------------------

    Lista_Seccion.sort(key=lambda p: p['numero'])
    #for m in Lista_Seccion:
    #    print(m)
    # ----------------------------------------
    i=0
    o=0
```

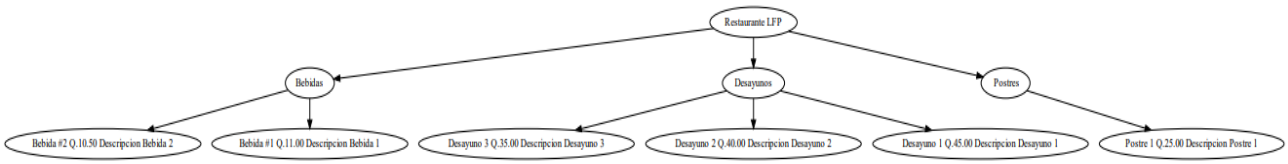
```
for x in Nombre_Seccion:
    i = i + 1
    aux1 = chr(66 + i)
    dot.node(str(aux1), label=r''+str(x)+'')
    dot.edges(['A'+str(aux1)])
    #print(x)
    for z in Lista_Seccion:
        if z['seccion'] == x:
            o =o+1
            Numero = float(z['numero'])
            aux2 = chr(96+o)
            #des = z['nombre']+" Q."+str("{:.2f}".format(Numero)+" \n"+z['descripcion'])
            dot.node(str(aux2),label=r''+z['nombre']+" Q."+str("{:.2f}".format(Numero)+ ' '+z['descripcion']+''))

            #dot.node(str(aux2),label= r'centro\nSal')
            dot.edges([str(aux1)+str(aux2)])
# -----------------------------------------------
#print(dot.source)
#print("-------------------------------------------------------")
dot.render('Grafica/Grafica_Salida.dot', view=True)  # doctest: +SKIP
'Grafica/Grafica_Salida.dot.pdf'
```

El Archivo Dot que se genera es el siguiente:

```
// Grafica Restaurante
digraph {
      A [label=”Restaurante LFP”]
      C [label=Bebidas]
      A -> C
      a [label=”Bebida #2 Q.10.50 Descripcion Bebida 2”]
      C -> a
      b [label=”Bebida #1 Q.11.00 Descripcion Bebida 1”]
      C -> b
      D [label=Desayunos]
      A -> D
      c [label=”Desayuno 3 Q.35.00 Descripcion Desayuno 3”]
      D -> c
      d [label=”Desayuno 2 Q.40.00 Descripcion Desayuno 2”]
      D -> d
      e [label=”Desayuno 1 Q.45.00 Descripcion Desayuno 1”]
      D -> e
      E [label=” Postres”]
      A -> E
      f [label=”Postre 1 Q.25.00 Descripcion Postre 1”]
      E -> f
                                                }
```

La Grafica


Restaurante LFP
Bebidas
Desayunos
Postres
Bebida #2 Q.10.50 Descripcion Bebida 2
Bebida #1 Q.11.00 Descripcion Bebida 1
Desayuno 3 Q.35.00 Descripcion Desayuno 3
Desayuno 2 Q.40.00 Descripcion Desayuno 2
Desayuno 1 Q.45.00 Descripcion Desayuno 1
Postre 1 Q.25.00 Descripcion Postre 1