MANUAL TECNICO

PROYECTO #1 LFP+A

Diego Facundo Pérez Nicolau
CARNET 202106538

Python:

Python es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

Programación Orientada a Objetos:

La programación orientada a objetos (POO) es una manera de estructurar el código que le hace especialmente efectivo organizando y reutilizando código, aunque su naturaleza abstracta hace que no sea muy intuitivo cuando se empieza. En general, los objetos se pueden considerar como tipos de datos con características propias que también pueden tener funcionalidades propias. De forma similar podemos crear nuestros propios objetos con características (llamadas atributos) y métodos propios. Los objetos se definen usando clases (class()) y las variables que de definen en ella son propiedades comunes de ese objeto.

Abrir y Crear archivos:

En Python, para abrir un archivo usamos la función open, que recibe el nombre del archivo a abrir. Esta función intentará abrir el archivo con el nombre indicado. Si tiene éxito, devolverá una variable que nos permitirá manipular el archivo de diversas maneras.
El contenido de la ventana de Python se puede guardar en un archivo de Python o un archivo de texto. Haga clic con el botón derecho del ratón en la ventana de Python y seleccione Guardar como para guardar el código como un archivo de Python (. py) o un archivo de texto

Tkinter:

Tkinter es un binding de la biblioteca gráfica Tcl/Tk para el lenguaje de programación Python. Se considera un estándar para la interfaz gráfica de usuario (GUI) para Python y es el que viene por defecto con la instalación para Microsoft Windows. El paquete tkinter es la interfaz por defecto de Python para el kit de herramientas de GUI Tk. Viene integrado con Python y no es necesario instalarlo. Es muy buena opción para quienes van empezando.

Declaración de variables métodos y clases

Abrir Archivos:

Las variables y métodos se declararon emperezando y manteniéndolas en minúsculas y si incluían mas de una palabra para separar una palabra de otra la nueva palabra se mantenía pegada pero su primera letra era mayúscula:

```
aTecnico= "Manuales\MANUAL TECNICO.pdf"
aUsuario= 'Manuales\MANUAL USUARIO.pdf'

def TemasAyuda():
    import InterfazAyuda.TemasAyuda
    InterfazAyuda.TemasAyuda.abrirWindowTA()

def abrirWindowA():
    windowHelp = tk.Tk()
    windowHelp.title("Consultar Ayuda")
    windowHelp.columnconfigure([0,4], minsize=150)
    windowHelp.rowconfigure([0,8], minsize=100)
    windowHelp.configure(background="#10ac84")

    button1= tk.Button(windowHelp, text ="Manual de Usuario",command=lambda: subprocess.Popen([aUsuario], shell=True),  bg="#1dd1a1") #Abre el Manual de Usuario
    button1.grid(row=2,column=1, padx=10,pady=10)

    button2= tk.Button(windowHelp, text ="Manual Tecnico",command=lambda: subprocess.Popen([aTecnico], shell=True), bg="#1dd1a1") #Abre el Manual Tecnico
    button2.grid(row=4,column=1, padx=10,pady=10)
```

(ejemplo dónde también se resume una palabra en las variables)

Leer info de archivos:

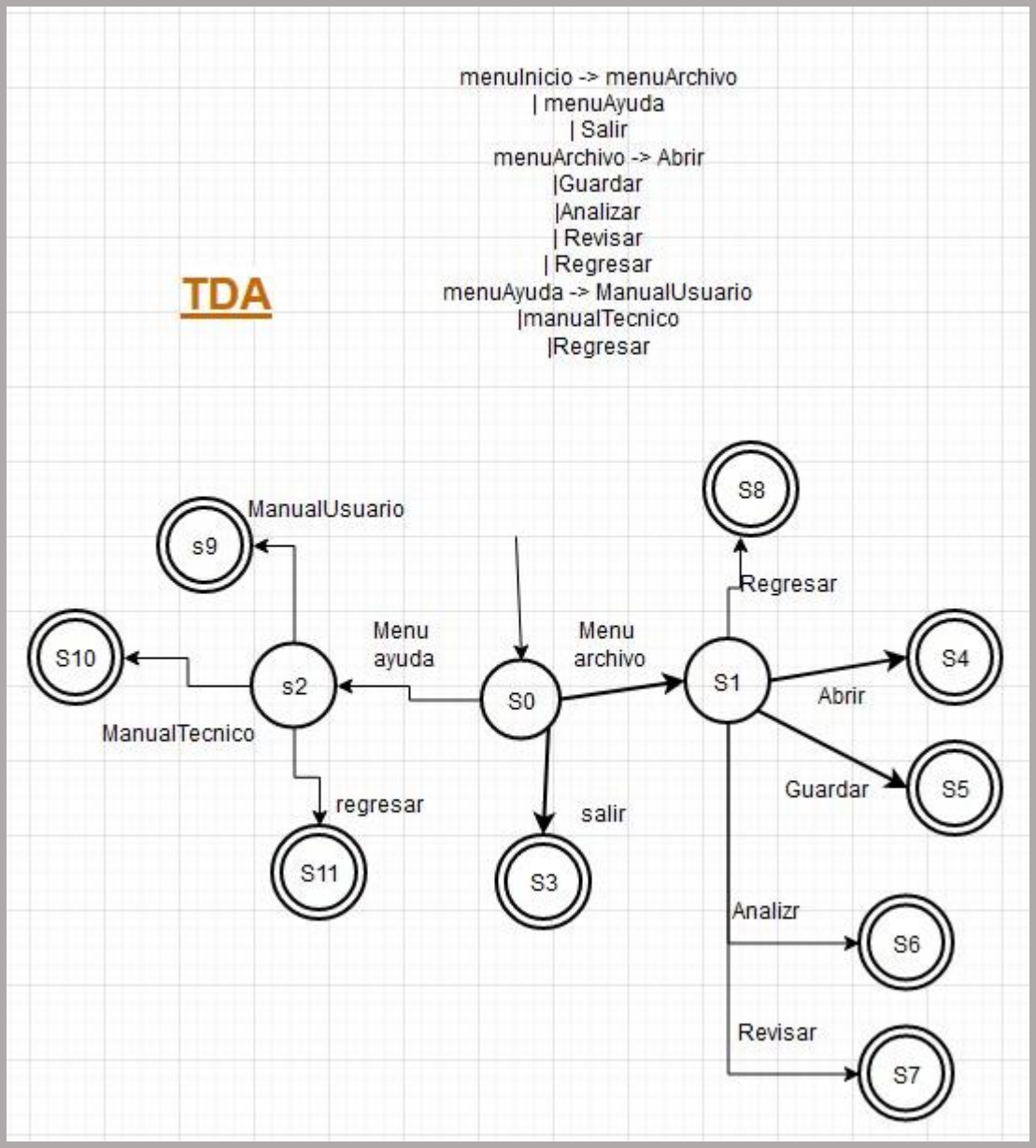
Las clases tienen que estar inicializadas con letra mayúscula seguidas de minúsculas hasta terminar la palabra y la siguiente palabra también empezarse por mayúscula.

```
def cargaMasiva(ruta=None):
        print('Subiendo datos') #Declaracion de que se estan subiendo
        print("Ruta: ", ruta)
        nombreArchivo = 'Archivo.txt'
        #nombreArchivo = ruta  tomar nombre y ruta del archivo de la barra de texto
        archivo = open (nombreArchivo,'r',encoding = "utf-8")   #Creando la variable archivo para python y diciendo qeu le lea con tildes y ñ
        datos = archivo.read()  #la variable datos toma el texto del archivo
        datos=datos.replace("\t", "")
        #print(datos)
        datos=datos.replace("\n", "")
        datos=datos.replace(" ", "")
        #print(datos)
        LeerEt(datos)
        archivo.close()     #cierra el archivo

cargaMasiva()
```

Guardar archivos:

Las clases tienen que estar inicializadas con letra mayúscula seguidas de minúsculas hasta terminar la palabra y la siguiente palabra también empezarse por mayúscula.

Diagrama


menuInicio -> menuArchivo
| menuAyuda
| Salir
menuArchivo -> Abrir
|Guardar
|Analizar
| Revisar
| Regresar
menuAyuda -> ManualUsuario
|manualTecnico
|Regresar
TDA
S8
ManualUsuario
s9
Regresar
S10
Menu
ayuda
Menu
archivo
S1
S4
s2
S0
Abrir
ManualTecnico
Guardar
S5
regresar
salir
S11
S3
Analizr
S6
Revisar
S7

Métodos

1. Abrir archivo: en este método se ingresa la ruta del archivo y mediante esta ruta la consola lo abre por la aplicación con la que se suele abrir este tipo de archivos.

```
def abrirArchivo(ruta):
    try:
        subprocess.Popen([ruta], shell=True)
    except:
        print("Hubo un fallo al abrir el archivo, revise su ruta")
```

2. Guardar Archivo / Guardar Archivo Como, en este método con 2 variantes se ingresa la información de la ruta del archivo y con esta se crea un nuevo archivo en la misma carpeta con el nombre escrito en Guardar Como, en Guardar el nombre del nuevo archivo esta preestablecido.

```
def guardar(ruta, nombreArchivo):

        archivo = open (ruta,'r')
        datos = archivo.read()
        archivo.close()

        newFile = open (nombreArchivo, "w")
        newFile.write(datos)
        newFile.close

        print(datos)
```

3. Revisar Errores: en este método se revisa el texto de un archivo leyendo lexema por lexema si es valido o no, si no lo es se guarda la información de este en una tabla.

```
def revisarErrores(datos):
        nFila=0
        nError=0

        lineas =datos.split("\n")
        for linea in lineas:
                nFila=nFila+1
                nColumna =0

                for token in linea:
                        nColumna=nColumna+1

                        if(token.isdigit()|token.isalpha() or token==" " or token == ">" or token=="<" or token == "/" or token==","  or token == "[" or token=="]" or token == "=" or toke
                                nError=nError
                        else:
                                nError=nError+1
                                print("Error # ", nError, ". ",token, "Fila: ",nFila, "  Columna: ",nColumna)
```

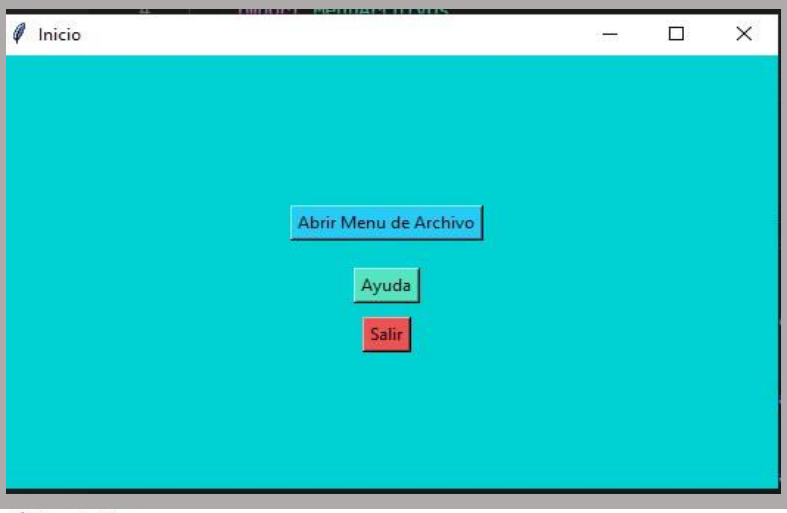
```
nFila=0
nError=0
lineas =datos.split("\n")
for linea in lineas:
    nFila=nFila+1
    nColumna =0

    for token in linea:
        nColumna=nColumna+1

        if(token.isdigit()|token.isalpha() or token==" " or token==">" or token=="<" or token=="/" or token==","  or token =="[" or token=="]" or token =="=" or token=="."):
            nError=nError
        else:
            nError=nError+1
            nuevaLinea=str("\n <tr> <td> "+ str(nError)+ "</td> <td> "+str(token)+ "</td> <td> Error </td> <td>"+str(nColumna)+ " </td> <td> "+str(nFila)+ "</td> </tr>")
            archivoHTMl.write(nuevaLinea)
```

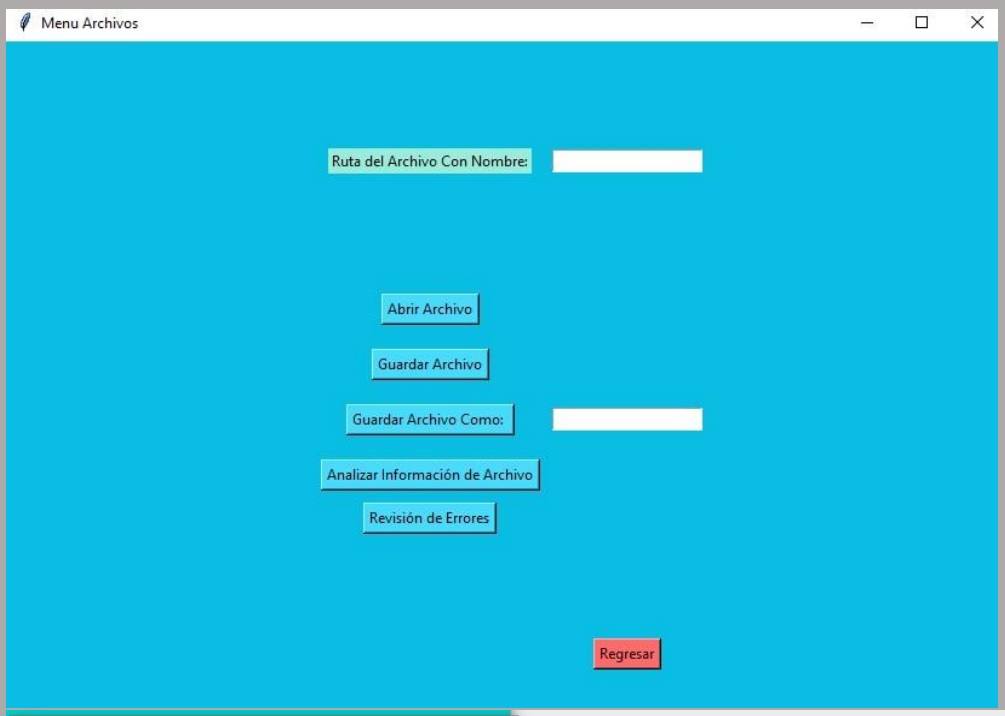
4. Analizar Información de un archivo, en este método se lee un archivo con el cual se sustrae la información para hacer operaciones y el texto de este, con estos datos se crea un archivo HTML que muestra el texto e información del archivo.

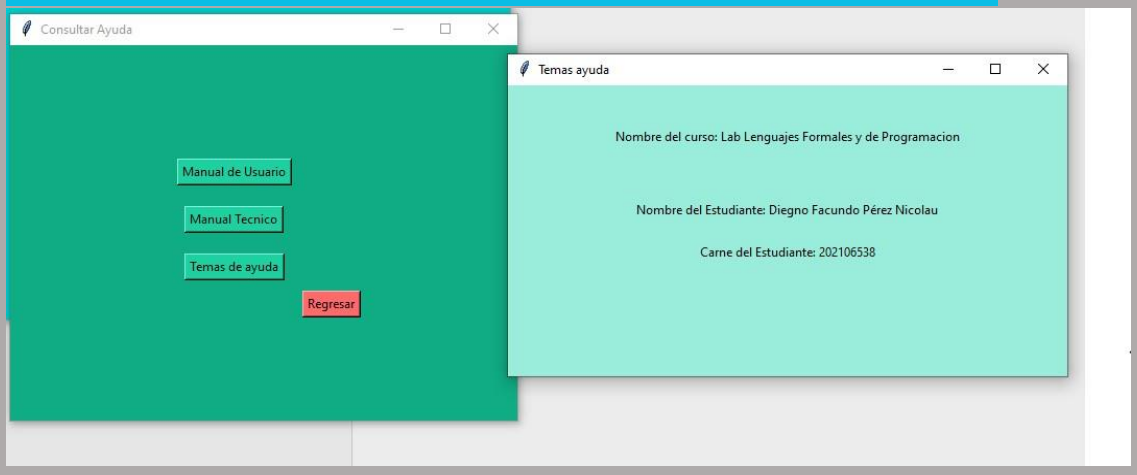
```
def crearHTMLResultados(ruta):
    archivo = open (ruta,'r')
    datos = archivo.read()
    textoA = conseguirTexto(datos)
    listasOperaciones = conseguirOperaciones(datos)
    archivo.close()

    archivoHTml = open ("PaginasHTML\RESULTADOS_202106538.html", "w")

    archivoHTml.write(inicioHTML)
    archivoHTml.write(cabezaHTML)
    archivoHTml.write(cabeceraHTML)

    archivoHTml.write("\n<article>")
    archivoHTml.write(textoA)
    archivoHTml.write("</article>\n")

    archivoHTml.write("\n<article>")

    for operacion in listasOperaciones:
        archivoHTml.write("\n<br>")
        archivoHTml.write("---------------Operacion----------------")
        for componente in operacion:
            archivoHTml.write("<br>")
            archivoHTml.write(str(componente))

crearHTMLResultados("Archivo.txt")
```

Interfaces y validaciones

Las interfaces es lo que vera el usuario al acceder al sistema.







Revisión de que el texto tuviera caracteres aceptados:

```
def revisarErrores(datos):
        nFila=0
        nError=0

        lineas =datos.split("\n")
        for linea in lineas:
                nFila=nFila+1
                nColumna =0

                for token in linea:
                        nColumna=nColumna+1

                        if(token.isdigit()|token.isalpha() or token==" " or token == ">" or token=="<" or token == "/" or token==","  or token == "[" or token=="]" or token == "=" or toke
                                nError=nError
                        else:
                                nError=nError+1
                                print("Error # ", nError, ". ",token, "Fila: ",nFila, "  Columna: ",nColumna)
```