Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2021.

PROYECTO 2

201344708 – Jose Manuel Lara Elias

Resumen

La programación orientada a objetos difiere de la programación estructural tradicional, en la que los datos y procedimientos de unos datos de entrada para obtener otros de salida. La programación Estructurada anima a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan. En el presente proyecto se escriben funciones que procesan datos. Empleando POO, en cambio, primero definiendo objetos para luego enviarles mensajes solicitándole que realicen sus métodos por si mismos

Palabras clave
Matriz, nodo, algoritmo, tuplas, interfaz.

Abstract

Object-oriented programming differs from traditional structured programming, in which data and procedures are separate and unrelated, since the only thing that is sought is the processing of some input data to obtain other output. Structured programming encourages me to think mostly in terms of procedures or functions, and secondly in the data structures that those procedures handle. Functions that process data are written in this project. using OOP, instead, first defining objects and then sending them messages asking them to perform their methods themselves.

Keywords
Matrix, node, algorithm, tuples, interface.

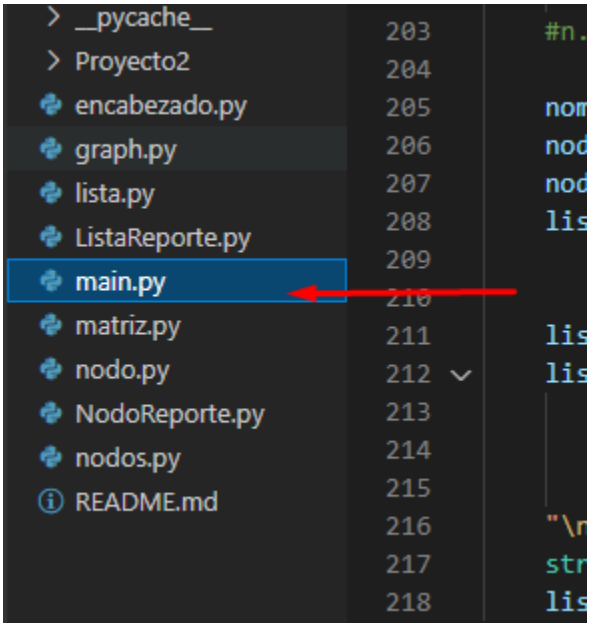
Introducción

Esta investigación tiene como objetivo realizar una descripción de las matrices dispersas, además evidencia su importancia, el funcionamiento de dichas matrices y sus diferentes formas de representación, mediante la implementación de esta teoría en un lenguaje de programación Visual Estudio. Como se visualiza en el artículo de matrices dispersas cada día tiende a ser mas utilizadas en las aplicaciones de computación científica, biomédicas y se caracterizan por que la mayoría de los elementos son cero.

Las matrices dispersas son ampliamente usadas en la computación científica, especialmente en la optimización a gran escala, análisis estructural y de circuitos, dinámica y fluidos computacionales y en general en solución numérica de ecuaciones diferenciales parciales; otras áreas de interés en donde se pueden aplicar la representación dispersas son la teoría de grafos, teoría de redes, la combinatoria, los métodos numéricos, entre otros.

Desarrollo del tema

Para desarrollar este proyecto necesité implementar una lista de matrices y en esta lista de matrices ingresé a mi matriz dispersa y así fui manejando la memoria de este proyecto.
Asimismo, tuve que crear una clase encabezado que se utiliza como su nombre lo indica para los encabezados de la matriz dispersa, también se crearon otras listas para los reportes como se puede apreciar en la imagen**.**



En las siguientes imágenes podremos observar como se desarrolló el interfaz.
Como primer paso creamos un objeto de el tipo Tk y lo llamamos ventana. Este objeto es el que utilizaremos para poder manejar la Liberia tkinter que tuvimos que importar con el nombre de tk como lo muestra la siguiente imagen.

```
ain.py > ...
from os import name
from tkinter import *
from tkinter.filedialog import askopenfilename
import xml.etree.ElementTree as ET
from matriz import matriz
from lista import ListaEnlazada
from tkinter import ttk
import tkinter as tk
from tkinter.messagebox import showinfo
```

Me vi obligado que manejar un ancho de ventana y alto para poder centrar la ventana para que ser viera de una mejor manera. Para obtener los valores en pixeles de el monitor utilizamos los dos siguientes comandos.

- ventana.winfo_screenwidth()
- ventana.winfo_screenheight()

Para poder enviar entonces parametors a la ventana utilizamos Geometry y reasizable con parámetros 0,0 para impedir que se pueda mover. Como podemos ver en la siguiente información.

```
ventana.geometry(posicion)
ventana.resizable(0,0)
```

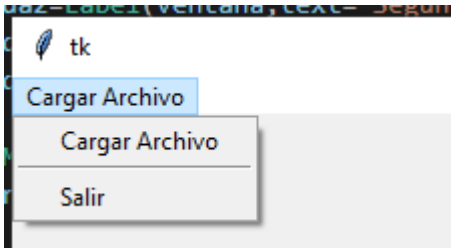
Para agregar un menú de barra de tareas implemente el siguiente código.

```
menubar = Menu(ventana)
ventana.config(menu=menubar)
filemenu = Menu(menubar, tearoff=0)

filemenu.add_command(label="Cargar Archivo", command=abrir)
filemenu.add_separator()
filemenu.add_command(label="Salir", command=ventana.quit)

menubar.add_cascade(label="Cargar Archivo", menu=filemenu, command=abrir)
```

Ya en el interfaz lo podemos visualizar de esta manera.



Para poder fijar los botones, labels y cajas de texto se utilizo los comandos place y config. Como ejemplo

```
lblCordenada2=Label(ventana,text='Segunda Cordenada')
lblCordenada2.place(x=600,y=500)
lblCordenada2.config(padx=10,pady=4)
```

- **Element Tree**

La biblioteca ElementTree incluye herramientas para analizar XML usando APIs basadas en eventos y documentos, buscando documentos analizados con expresiones XPath, y creando nuevos o modificando documentos existentes.
Para implementarlo lo primero que hice fue hacer el import de la siguiente manera.

```
import xml.etree.ElementTree as ET
```

Ya dentro de un metodo cree una variable y le agregue el valor de **:**

```
askopenfilename()
```

Para realizar el parseo se utilizan las letras ET.parse() mandándole el valor a parsear y el comando. Por ultimo utilice root para poder manejar el árbol completo es así:

```
filename = askopenfilename()
tree = ET.parse(filename)
root = tree.getroot()
```

En este método en el primer foro que se puede visualizar creamos un elemento llamado n tipo matriz utilizando ese foro podemos irse corriendo todos los valores que nos devolvió nuestro árbol llamado Ruth entonces cada vez que el Ford encuentre un nuevo elemento va a crear un valor llamado n de tipo matriz de la clase que creamos tipo matriz que más adelante explicaremos de qué se encarga.

```
for elemento in root:
 n = matriz()
   # print(elemento.tag) #tag
 for subelemento in elemento:
        #print('> ' + subelemento.text) #valores ->
        if subelemento.tag == 'nombre':
            nombre = subelemento.text
        if subelemento.tag == 'filas':
            filas = subelemento.text
        if subelemento.tag == 'columnas':
```

En la siguiente imagen vamos a poder visualizar como adentro del primer Ford que explicamos anteriormente se encuentra otro Ford con 4 ifs anidados eso para ir reconociendo los tags e ir separándolos ya sea nombre, fila, columnas, imagen. A la hora de encontrar una imagen inicializamos el contador columnas en cero y el contador de filas en cero.

```
for elemento in root:
 n = matriz()
   # print(elemento.tag) #tag
 for subelemento in elemento:
        #print('> ' + subelemento.text) #valores -> text
        if subelemento.tag == 'nombre':
            nombre = subelemento.text
        if subelemento.tag == 'filas':
            filas = subelemento.text
        if subelemento.tag == 'columnas':
            columnas = subelemento.text
        if subelemento.tag == 'imagen':
            imagen = subelemento.text
            contadorColumnas = 0
            contadorFilas = 0
```

En la siguiente imagen vamos a poder visualizar que vamos a tener el tercer por anidado en este foro tenemos 3 if adentro los cuales van a ir diferenciando el tipo de carácter que tenemos en la imagen y poder ir separándolos y guardándolos en el objeto tipo n tipo matriz cree que creamos anteriormente. cada vez que encontramos un carácter se suma una columna para poder ir moviéndonos alrededor de la imagen y cada vez que encontramos un

Se vuelve el contador de columnas 1 y le sumamos 1 al contador de filas respectivamente.

```
for k in imagen:
    if(k == '-'):
        n.insertar(contadorFilas,contadorColumnas,'-')
        contadorColumnas = contadorColumnas +1

    if(k == '*'):
        n.insertar(contadorFilas,contadorColumnas,'*')
        contadorColumnas = contadorColumnas +1

    if(k == '\n'):
        contadorFilas = contadorFilas +1
        contadorColumnas = 1
```

Después de que todos los fors terminé creó una variable llamada nodo y en esta adentro inserte un objeto tipo lista y a este objeto le envío los valores nombre, fila, columna y n. con esto en la línea 59 yo obtengo el valor DN que es una lista la lista de mi matriz dispersa, entonces en esta línea agregó mi matriz dispersa a la lista de listas de matrices que tengo por último en la línea 60 en el en la lista combo agregó los nuevos nombres de la matriz a mi lista combo box .

```
58      nodo = lista.insertarFinal(nombre,filas,columnas,n)
59      nodo.imagen = n
60      listaCombo.append(nombre)
61
```

En la siguiente imagen podemos ver el método que utilizó para rotar una imagen horizontalmente. Primero creo con objeto llamado n tipo matriz después de eso vengo yo tengo mi no respectivo utilizando el nombre que da al usuario utilizando el combobox que aparece en el interfaz luego creó una variable que se llama filas y en este ingreso el valor de cuantas filas tiene mi matriz dispersa, está la utilizó para poder hacer la rotación.

Luego con un Ford voy recorriendo toda mi matriz dispersa y voy guardando los datos ya de manera

horizontal siempre en los if anidados que aparecen sumando con columnas y sumando sus filas para que no se pierda el orden. Por último lo que hago es que insertó la nueva matriz ya rotada en la lista de matrices con su nombre respectivo Y luego ingresó la información que se ejecutó en el reporte HTML .

```
def pulsar():
    n = matriz()
    contadorColumnas = 1
    print("rotando Imagen")
    nodo = lista.getNodo(comboExample.get())
    filas = lista.getFilas(comboExample.get())
    print("el numero de filas es: " +str(filas))

    NuevaImagen = nodo.imagen.rotacion()
    print(NuevaImagen)
    contadorRotacion = int(filas)

    for k in NuevaImagen:
        if(k == '*'):
            n.insertar(contadorRotacion,contadorColumnas,'*')
            contadorColumnas = contadorColumnas +1
        if(k == '-'):
            n.insertar(contadorRotacion,contadorColumnas,'-')

            contadorColumnas = contadorColumnas +1
        if(k == '\n'):
         contadorRotacion = contadorRotacion - 1

         contadorColumnas = 1
    #n.recorrerFilas()

    nombre = comboExample.get()+"RotadaHorizontal"
    nodo = lista.insertarFinal(nombre,contadorColumnas,filas,n)
    nodo.imagen = n
    listaCombo.append(comboExample.get()+'RotadaHorizontal')

    lista.crearImagen()
    listaReporte.insertar("Se roto la imagen de manera Horizontal \n"+"Nombre de la matriz: "+comboExample.get()+"\nA la hora: "+
    str(now.date())+ " "+ str(now.time()))
    listaReporte.mostrar()
```

Generar el HTML

Para generar mi HTML cree una lista de strings para cada vez que se ejecute algo que desee guardar puedo guardarlo en esa lista, para esto creó un método que se llama mostrar el cual va recorriendo toda mi toda mi lista de strings y cada vez que encuentra un string le agrega una tabla y Así mismo voy concatenando las celdas luego de eso la variable que usé para concatenar le agregó la información para complementar la tabla en HTML correctamente luego de eso se usa el comando f.write para escribir el archivo y le enviamos la variable donde tenemos toda la información qué queremos enviar a nuestro HTML

```
def mostrar(self):
    concatenar = ""
    tmp = self.inicio
    while tmp is not None:
        print('Mensaje: '+ str(tmp.StringReporte))
        concatenar =  concatenar + "<tr>"+"<td>"+str(tmp.StringReporte)+"</td>"+"</tr>"
        tmp = tmp.siguiente

    hola = concatenar = """<html>
    <head>
    <title>Reporte Proyecto 2</title> <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="estilosCU01052D.css">
    </head>
    <body>
    <table border="1">
    <caption>Reporte Proyecto 2</caption> """ + concatenar + """</table>
    </body>
    </html>"""

    f = open('Reporte.html','w')

    mensaje = hola

    f.write(mensaje)
    f.close()
```

.

Para poder abrir un documento automáticamente HTML utilice lo siguiente:

primero tenemos que importar

```
import webbrowser
```

Luego con la siguiente línea de código abrimos el archivo enviándole el nombre del archivo HTML.

```
webbrowser.open_new_tab('Reporte.html')
```

Como se puede apreciar en la siguiente imagen este algoritmo lo utilizó para poder llenar mi lista reporte obtengo el nombre de la matriz mediante el combobox después de eso busco el nodo que tenga el nombre de la matriz y con eso llenó nuevamente mi combobox para actualizar el nombre de mis listas en el combobox.

Después de eso en un string insertó un nuevo no tipo lista reporte y le envío el reporte correspondiente.

```
lista.crearImagen()
listaReporte.insertar("Se limpio el area de "+"fila 1: "+ CordenadaFila1.get("1.0","end")+","+
                                              "columna 1: "+ CordenadaColumna1.get("1.0","end")+","+
                                              "fila 2: "+ CordenadaFila2.get("1.0","end")+","+
                                              "columna 2: "+ CordenadaColumna2.get("1.0","end")+","+
"\n"+"Nombre de la matriz: "+comboExample.get()+"\n A la hora: "+
str(now.date())+ " "+ str(now.time()))
listaReporte.mostrar()
```

Creación De un Nodo

Para la creación de 1 pondré como ejemplo la creación de mí no que utilicé en la matriz. este no en su constructor tiene como parámetros nombre coma fila, columna, imagen y en sus características tiene las mismas, pero con el detalle de que la característica imagen este tipo matriz.

```
nodo.py > Nodo > __init__
1  from matriz import matriz
2  class Nodo:
3      def __init__(self, nombre,filas, columnas, imagen):
4          self.nombre = nombre
5          self.filas = filas
6          self.columnas = columnas
7          self.imagen = matriz()
8          self.siguiente = None
```

Este es el método que se utiliza para poder tener una lista de listas, en este ejemplo es una lista de listas de matrices dispersas.
En la siguiente imagen podemos observar nuestra clase tipo matriz que mandamos a llamar en lo nodo mencionado anteriormente.

Como podemos observar el método de la clase recibe como parámetros una lista de encabezados que esa es la que nos indica las filas y las columnas

```
class matriz:
    def __init__(self) :
        self.eFilas = listaEncabezado()
        self.eColumnas = listaEncabezado()
```

En la siguiente imagen se va a poder observar el método insertar que recibe como parámetros fila, columna y valor y crea un nuevo objeto tipo no recibiendo estos parámetros mencionados anteriormente fila.

Después de haber creado el nodo correspondiente el algoritmo empieza a hacer las condiciones correspondientes para saber dónde ingresar el nuevo nodo.

```
def insertar(self, fila, columna, valor):
    nuevo = Nodo(fila, columna, valor)
      #inrestar por filas
    eFila = self.eFilas.getEncabezado(fila)
    if eFila == None:
        eFila = nodoEncabezado(fila)
        eFila.accesoNodo = nuevo
        self.eFilas.setEncabezado(eFila)
    else:
        if nuevo.columna < eFila.accesoNodo.columna:
            nuevo.derecha = eFila.accesoNodo
            eFila.accesoNodo.izquierda = nuevo
            eFila.accesoNodo = nuevo
        else:
            actual = eFila.accesoNodo
            while actual.derecha != None:
                if nuevo.columna < actual.derecha.columna:
                    nuevo.derecha = actual.derecha
                    actual.derecha.izquierda = nuevo
                    nuevo.izquierda = actual
                    actual.derecha = nuevo
                    break
                actual = actual.derecha

            if actual.derecha == None:
                actual.derecha = nuevo
                nuevo.izquierda = actual
```

Conclusiones

Algunas particularidades de POO en Python son las siguientes:

- Todo es un objeto, incluyendo los tipos y clases.
- Permite herencia múltiple.
- No existen métodos ni atributos privados.
- Los atributos pueden ser modificados directamente.
- Permite «monkey patching».
- Permite «duck typing».
- Permite la sobrecarga de operadores.
- Permite la creación de nuevos tipos de datos.

Cada objeto tiene una identidad, un tipo y un valor. Una identidad de objecto nunca cambia una vez es creada; usted puede pensar eso como la dirección de objeto en memoria. El operador in compara la identidad de dos objetos; la función id() devuelve un número entero representando la identidad
Cada objeto tiene una identidad, un tipo y un valor. Una identidad de objecto nunca cambia una vez es creada; usted puede pensar eso como la dirección de objeto en memoria. El operador in compara la identidad de dos objetos; la función id() devuelve un número entero representando la identidad

El tipo de un objeto determina las operaciones que admite el objeto (por ejemplo, «¿tiene una longitud?») Y también define los valores posibles para los objetos de ese tipo. La función type () devuelve el tipo de un objeto (que es un objeto en sí mismo). El *valor *de algunos objetos puede cambiar. Se dice que los objetos cuyo valor puede cambiar son *mutables*; los objetos cuyo valor no se puede cambiar una vez que se crean se llaman *inmutable*. (El valor de un objeto contenedor inmutable que contiene una referencia a un objeto mutable puede cambiar cuando se cambia el valor de este último; sin embargo, el contenedor todavía se considera inmutable, porque la colección de objetos que contiene no se puede cambiar.
Por lo tanto, la inmutabilidad no es estrictamente lo mismo que tener un valor incambiable, es más sutil.) La mutabilidad de un objeto está determinada por su tipo; por ejemplo, los números, las cadenas y las tuplas son inmutables, mientras que los diccionarios y las listas son mutables.

Aunque las ideas anteriores no pierden su validez al incorporar datos espaciales, la inclusión de estos no es en absoluto obvia, y presenta una complejidad adicional que requiere de nuevos planteamientos para poder seguir trabajando con la base de datos de una forma similar a como sucede cuando se trabaja con los tipos de datos habituales

Referencias bibliográficas

colaboradores de Wikipedia. (2020, 24 diciembre). Tipo de dato abstracto. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Tipo_de_dato_

Hu, J. (2020, 25 junio). Tutorial de Tkinter - Combobox. Delft Stack. https://www.delftstack.com/es/tutorial/tkinter-tutorial/tkinter-combobox/

Hernandez, C. (2021, 6 abril). Estilizando tablas - Aprende sobre desarrollo web | MDN. Computee. https://developer.mozilla.org/es/docs/Learn/CSS/Building_blocks/Styling_tables