MANUAL TECNICO

PROYECTO #1 IPC 2 N

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.

Python is a high-level interpreted programming language whose philosophy emphasizes the readability of its code. It is a multi-paradigm programming language, as it partially supports object-oriented, imperative programming and, to a lesser extent, functional programming. It is an interpreted, dynamic and cross-platform language.

Programación Orientada a Objeto s(Object Oriented Programing):

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.

Object-oriented programming (OOP) is a way of structuring code that makes it especially effective at organizing and reusing code, although its abstract nature makes it not very intuitive when starting out. In general, objects can be thought of as data types with their own characteristics that can also have their own functionalities. Similarly, we can create our own objects with their own characteristics (called attributes) and methods. Objects are defined using classes (class()) and the variables defined in it are common properties of that object.

TDA:

Tipo abstracto de datos es un modelo matemático compuesto por una colección de operaciones definidas sobre un conjunto de datos para el modelo.

Abstract data type is a mathematical model composed of a collection of operations defined on a set of data for the model.

Declaración de variables métodos y clases
Declaración of variables and methods

Variables Y Métodos:

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:

Variables and methods were declared by prefixing and keeping them in lowercase and if they included more than one word to separate one word from another the new word was kept pasted but its first letter was capitalized:

(

Clases:

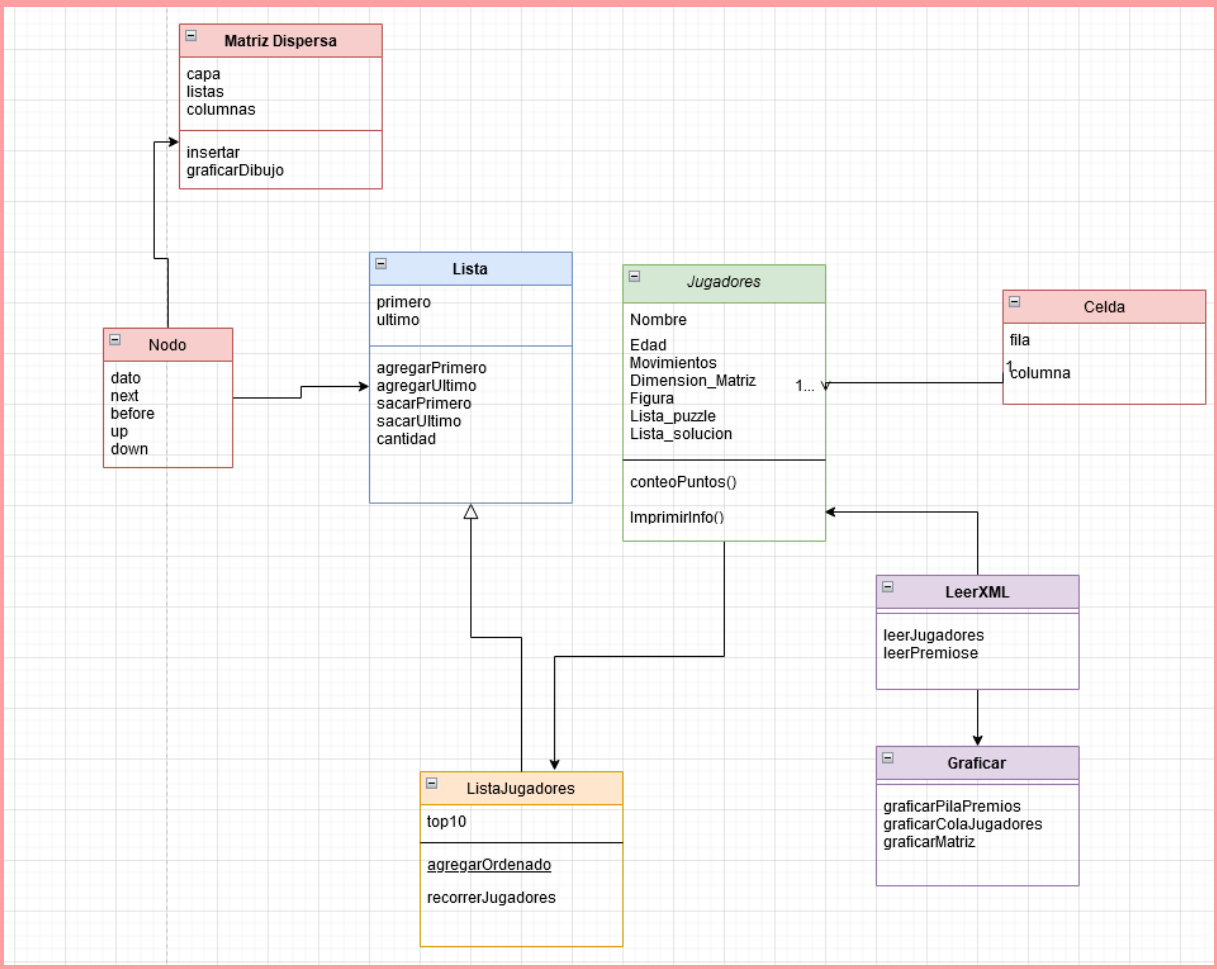
Las clases tienen que estar inicializadas con letra mayúscula seguidas de minúsculas hasta terminar la palabra y la siguiente palabra también empezarse por mayúscula.

```
class Jugador:

    def __init__(self,nombre,edad,moves, size, figura, puzzle, solucion):
        if(verificarMatriz(size) and verificarMovs(moves)):
            self.nombre=nombre
            self.edad= edad
            self.movimientos=moves
            self.size=size
            self.figura=figura
            self.puzzle= puzzle     #podria ser una lista con las celdas marcadas
            self.solucion = solucion    #tambien podria ser una lista conlas celdas
            self.puntos=conteoPuntos(size,moves,figura)

        elif(verificarMatriz(size)):
            print("El numero de movimientos no  debe ser mayor a 10000")
        else:
            print("Tamaño de matriz no valido, debe ser multiplo de 5 y maximo de 30 x 30")
```

Diagrama



Métodos

```
windowMain = tk.Tk()
windowMain.title("Menu Principal") #Asignarle titulo a ala ventana
windowMain.columnconfigure([0,4], minsize=200) #Columnas de la Ventana
windowMain.rowconfigure([0,10], minsize=100) #Filas de la Ventana y su proporción
windowMain.configure(background="#9AECDB")

label1 = tk.Label(text="Nombre del curso: Lab Lenguajes Formales y de Programacion", bg="#9AECDB")
label1.grid(row=0,column=1, pady=10)

label2 = tk.Label(text="Nombre del Estudiante: Diegno Facundo Pérez Nicolau", bg="#9AECDB")
label2.grid(row=1,column=1, pady=10)

label3 = tk.Label(text="Carne del Estudiante: 202106538", bg="#9AECDB")
label3.grid(row=2,column=1, pady=10)

button1= tk.Button(text ="Cargar Archivo", command=lambda: abrirC(), bg="#55E6C1") #Boton de cargar Archivo
button1.grid(row=3,column=1, pady=10)

button2= tk.Button(text ="Gestionar cursos", command=lambda: abrirG(), bg="#25CCF7") #Boton de Gestionar los cursos
button2.grid(row=6,column=1, pady=10)

button3= tk.Button(text ="Conteo de Créditos", command=lambda: abrirCC(), bg="#F8EFBA") #Boton de Contar los creditos segun los cursos
button3.grid(row=8,column=1, pady=10)

button4= tk.Button(text ="Salir", command=windowMain.destroy, bg="#FD7272")   #Botón de salir
button4.grid(row=9,column=1)
windowMain.mainloop()               #Llamamos a la ventana
```

```python
def LeerPremiosXML1(ruta):
    pilaRegalos= ListaSimple()

    doc = minidom.parse(ruta)
    regalos = doc.getElementsByTagName("regalo")


    for regalo in regalos:
        print(regalo.firstChild.data)
        pilaRegalos.agregarPrimero(regalo.firstChild.data)

    temp=pilaRegalos.primero
    while temp!=None:
        print("Premio: ",temp.dato)
        temp=temp.next

    crearGraficaPremios(pilaRegalos)
    return pilaRegalos
```

```python
class ListaSimple():
    def __init__(self):
        self.primero=None
        self.ultimo=None

    def estaVacia(self):
        return self.primero==None

    def agregarPrimero(self,dato): #Pila

        new=Nodo(dato)

        if self.primero==None:
            self.primero=self.ultimo=new
        else:
            temp = new
            temp.next = self.primero
            self.primero = temp

    def agregarUltimo(self,dato):  #Cola

        new=Nodo(dato)
        temp = self.primero

        if self.primero==None:
            print("Estaba vacia")
            self.primero=self.ultimo=new
        else:
            temp=self.primero
            while temp.next is not None:
                temp=temp.next

            self.ultimo=new
            new.before=temp
            temp.next=new

```

```
def celebrarPremiacion(listaJugadores:ListaJugadores, pilaPremios:ListaSimple):

    top10=listaJugadores.Top10()

    if top10.cantidad() > pilaPremios.cantidad():
        d=top10.cantidad()-pilaPremios.cantidad()
        while d>0:
            top10.sacarPrimero()
            d=top10.cantidad()-pilaPremios.cantidad()

    elif pilaPremios.cantidad() > top10.cantidad():
        d=pilaPremios.cantidad()-top10.cantidad()
        while d>0:
            pilaPremios.sacarUltimo()
            d=pilaPremios.cantidad()-top10.cantidad()

    while (pilaPremios.primero is not None) and (top10.primero is not None):
        jugadorPremiado=top10.sacarPrimero()
        premio=pilaPremios.sacarPrimero()


        print("\n ¡¡¡¡¡ ++++++++++++++++++++++")
        print("Felicidades al jugador: ", jugadorPremiado.dato.nombre, " se lleva de regalo: ", premio.dato)
        print(" ++++++++++++++++++++++ !!!!!!\n")

        crearGraficaPremios(pilaPremios)
        print("se creo grafica de pila de regalos")
        crearGraficaJugadores(listaJugadores)

        input("presione Enter para continuar")

    print("La premiacion llego a su fin xd")
```

```
def textoGraficaPremios(pilaPremios: ListaSimple):
    text=" digraph P{ \n\n"
    text+=" subgraph cluster_0 { \n\t"
    text+=" style=filled; \n\t color=palegreen2;\n\n\t"
    text+=" Pila[color=\"tomato\" shape=\"record\"    label=\"{"

    aux=pilaPremios.primero
    if aux==None:
        print("La pila esta vacia")
    while aux!=None:
        text+=str(aux.dato)
        if aux.next!=None:
            text+="|"
        aux=aux.next

    text+="}\" ] \n\n\t"
    text+="} \n}"
    return text

def crearGraficaPremios(pilaPremios):
    try:
        file= open('Graficas_Ordenes/PremiosGrafica.txt', 'w')
    except:
        os.mkdir('Graficas_Ordenes')
        file= open('Graficas_Ordenes/PremiosGrafica.txt', 'w')

    contenido=textoGraficaPremios(pilaPremios)
    file.write(contenido)
    file.close()
    os.system("dot -Tpng Graficas_Ordenes/PremiosGrafica.txt -o Graficas_Ordenes/PremiosGrafica.png")
    os.system("dot -Tpdf Graficas_Ordenes/PremiosGrafica.txt -o Graficas_Ordenes/PremiosGrafica.pdf")
```

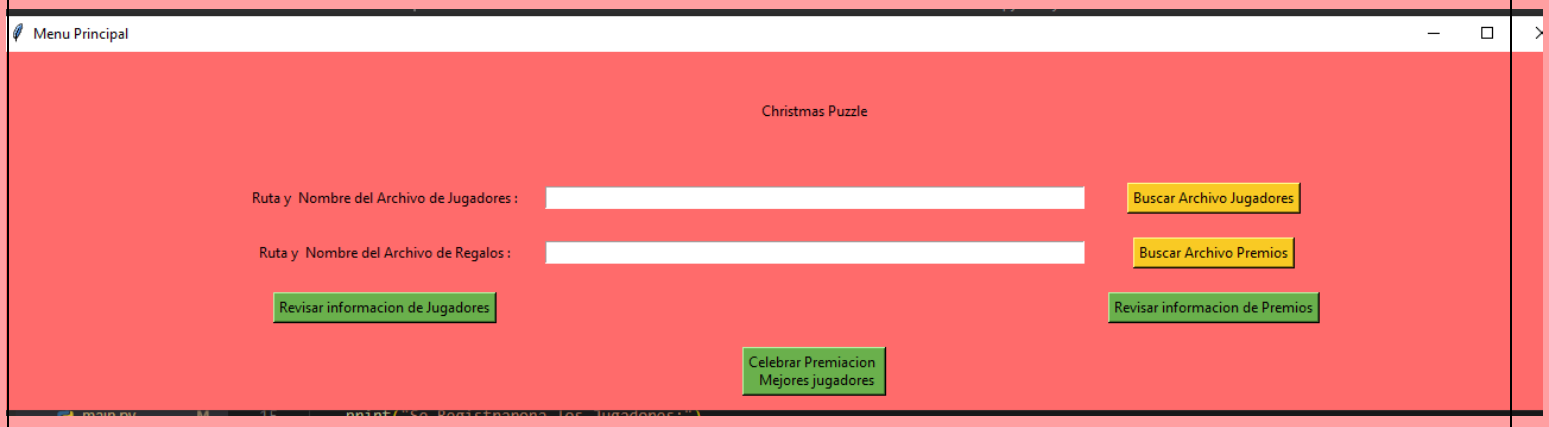
Descripción
Description

Se busca que se sea capaz de dar una solución al problema que se le plantea, mediante la lógica y conocimientos que se le han impartido durante la clase y que han sido aplicados en el laboratorio.

The aim is to be able to give a solution to the problem posed, using the logic and knowledge that have been taught during the class and that have been applied in the laboratory.

Interfaces y validaciones

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



Por otra parte, las validaciones son agregados a los métodos donde se confirma que los valores insertados son válidos para el proceso.

```
def verificarMatriz(size):
    if(-1<size<31 and size%5==0):
        return True
    else:
        return False

def verificarMovs(movimientos):
    if(-1<movimientos<10000):
        return True
    else:
        return False
```