MANUAL TECNICO

PRÁCTICA #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.

Listas:

Para declarar una lista se usan los corchetes [], en cambio, para declarar una tupla se usan los paréntesis (). En ambas los elementos se separan por comas.

Tanto las listas como las tuplas pueden contener elementos de diferentes tipos. No obstante, las listas suelen usarse para elementos del mismo tipo en cantidad variable.

Para acceder a los elementos de una lista o tupla se utiliza un índice entero (empezando por "0", no por "1"). Se pueden utilizar índices negativos para acceder elementos a partir del final.

Las listas se caracterizan por ser mutables, es decir, se puede cambiar su contenido en tiempo de ejecución.

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

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:

```
crdtsAprobados=str(creditosAprobados())
crdtsCursando=str(creditosCursando())
crdtsPendientes=str(creditosPendientes())
semestre = "N"
```

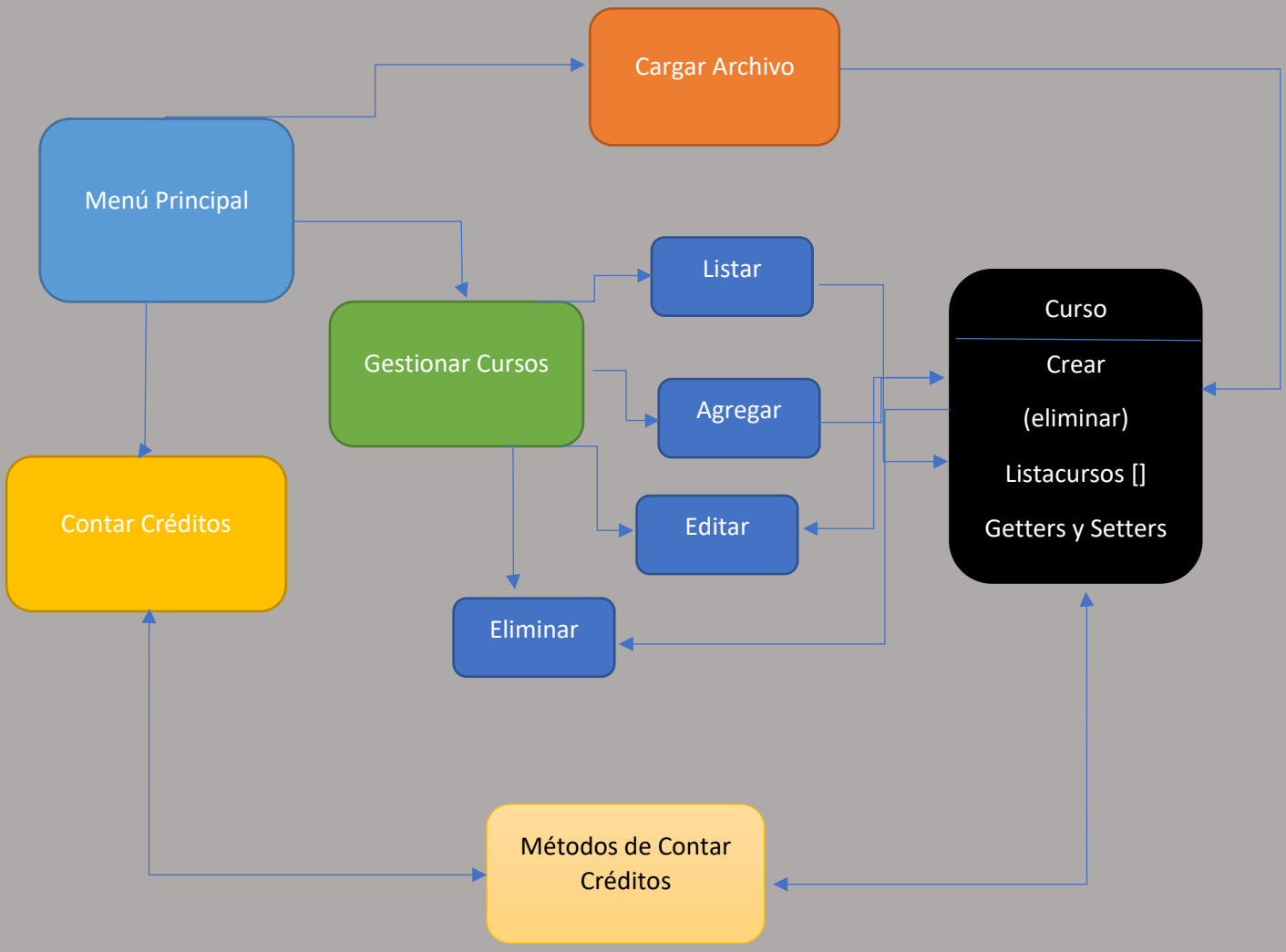
(ejemplo dónde también se resume una palabra en las variables)

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 Curso:

    def __init__(self, codigo, nombre, prerrequisitos, obligatorio, semestre, creditos, estado):
        try:
            self.codigo = readNum(codigo)
            self.nombre = nombre
            self.prerrequisitos = prerrequisitos
            self.obligatorio = readNum(obligatorio)
            self.semestre = readNum(semestre)
            self.creditos = readNum(creditos)
            self.estado = readNum(estado)
        except:
            print("Fallo al crear el curso intente de nuevo llenando correctamente los campos")
```

Diagrama


Cargar Archivo
Menú Principal
Listar
Gestionar Cursos
Agregar
Curso
Crear
(eliminar)
Listacursos []
Getters y Setters
Contar Créditos
Editar
Eliminar
Métodos de Contar Créditos

Métodos

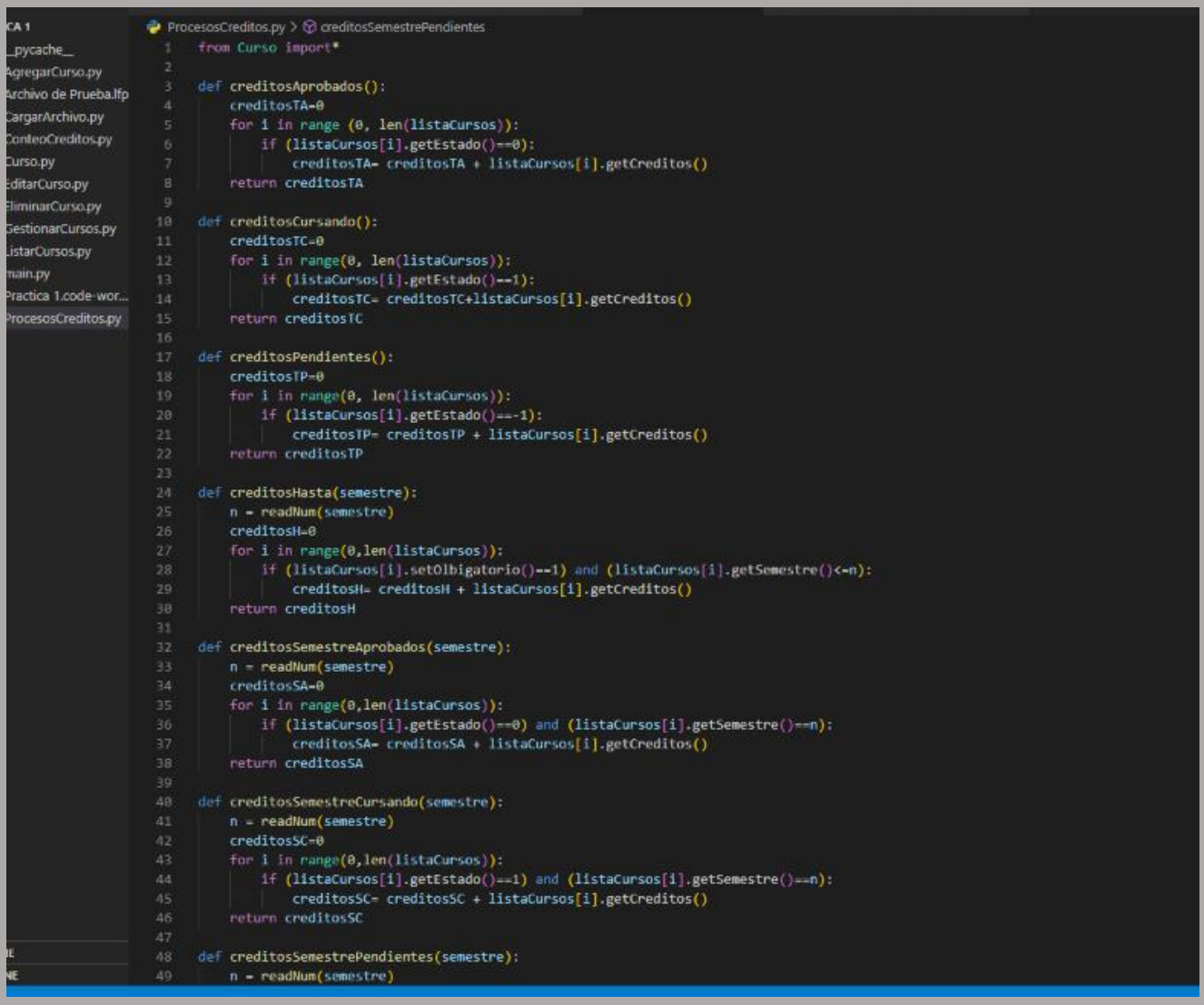
1. Cargar archivos: En este método que sirve para cualquier formato donde se dividan los cursos por los cambios de línea y los datos de los cursos por comas, haremos una carga masiva en la que para confirmar el proceso se ira imprimiendo los datos de cada curso

```
def cargaMasiva():
    print('Subiendo datos') #Declaracion de que se estan subiendo
    imprimir()
    nombreArchivo = 'Archivo de Prueba.lfp'
    nombreArchivo = textBox1.get()  #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
    print(datos)      #imprime el texto en el archivo
    cargaPorTexto(datos)
    archivo.close()     #cierra el archivo

def cargaPorTexto(datos):
    intento = False
    cuantos=0
    variosCursos = datos.split("\n")
    print('Numero de Cursos ingresados '+ str(len(variosCursos)))

    for i in range(0,len(variosCursos)):
        cuantos=cuantos+1

        try:
            infoCurso=variosCursos[i].split(",")
            codigo= str(infoCurso[0])
            nombre= str(infoCurso[1])
            prerrequisitos= str(infoCurso[2])
            obligatorio= str(infoCurso[3])
            semestre= str(infoCurso[4])
            creditos= str(infoCurso[5])
            estado= str(infoCurso[6])
            print("-----------------------------------------------------------------------------------------------------------------<3")
            print("# "+str(cuantos)+". Codigo: "+codigo+", Nombre: "+nombre+", Prerrequisistos: "+prerrequisitos+", Obligatorio: "+obligatorio+", Semestre: "+semestre+", Creditos: "+credito
            intento= True
        except:
            print('Hubo un error al guardar')
        if (intento):
            try:
                guardarCurso(Curso(codigo,nombre,prerrequisitos,obligatorio,semestre,creditos,estado))
                print('     Se guardó el curso '+nombre)
            except:
                print('Hubo un error al guardar')
```

2. Abrir ventanas con su información y botones: Las ventanas tienen muchas especificaciones como su tamaño y los elementos que las conforman a su vez estos elementos como botones tienen características.

```
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
```

3. Conteo de Créditos: todos los cursos tienen un atributo que son los créditos, estos créditos tienen opciones de ser contados en función de los datos de sus cursos, por ejemplo, cuantos créditos se tendrán en N semestre o cuantos créditos están en cursos aprobados, etc.



4. Crear, eliminar y editar cursos: para crear cursos se toma la información ya sea de la carga masiva o de la ventana con las cajas de texto, para editarlos se ingresa el código del curso y de las demás cajas de texto sus datos a cambiar, y para eliminarlos se toma y confirma el código del curso a eliminar.

```python
class Curso:

    def __init__(self, codigo, nombre, prerrequisitos, obligatorio, semestre, creditos, estado):
        try:
            self.codigo = readNum(codigo)
            self.nombre = nombre
            self.prerrequisitos = prerrequisitos
            self.obligatorio = readNum(obligatorio)
            self.semestre = readNum(semestre)
            self.creditos = readNum(creditos)
            self.estado = readNum(estado)
        except:
            print("Fallo al crear el curso intente de nuevo llenando correctamente los campos")
```

```python
def editarCurso():
    imprimirTodosCursos()
    if (Entry1.get()=="" or Entry2.get()=="" or comboBox1.get()=="" or comboBox2.get()=="" or Entry4.get()=="" or comboBox3.get()==""):
        print("Llene todos los datos adecuadamente")
    else:
        cursoEdit = buscarCurso(Entry1.get()) #Se busca al curso que se editara por su codigo
        cursoEdit.setNombre(Entry2.get())
        cursoEdit.setPrerrequisitos(Entry3.get())
        cursoEdit.setSemestre(comboBox1.get())
        cursoEdit.setOlbigatorio(comboBox2.get())
        cursoEdit.setCreditos(Entry4.get())
        cursoEdit.setEstado(comboBox3.get())

    imprimirDatosCurso(cursoEdit)
```

```python
def eliminarCurso(curso):
    for i in range(0,len(listaCursos)):
        try:
            if curso==listaCursos[i]:
                listaCursos.pop(i)
        except:
            print("ese curso ya no está")
```

Descripción

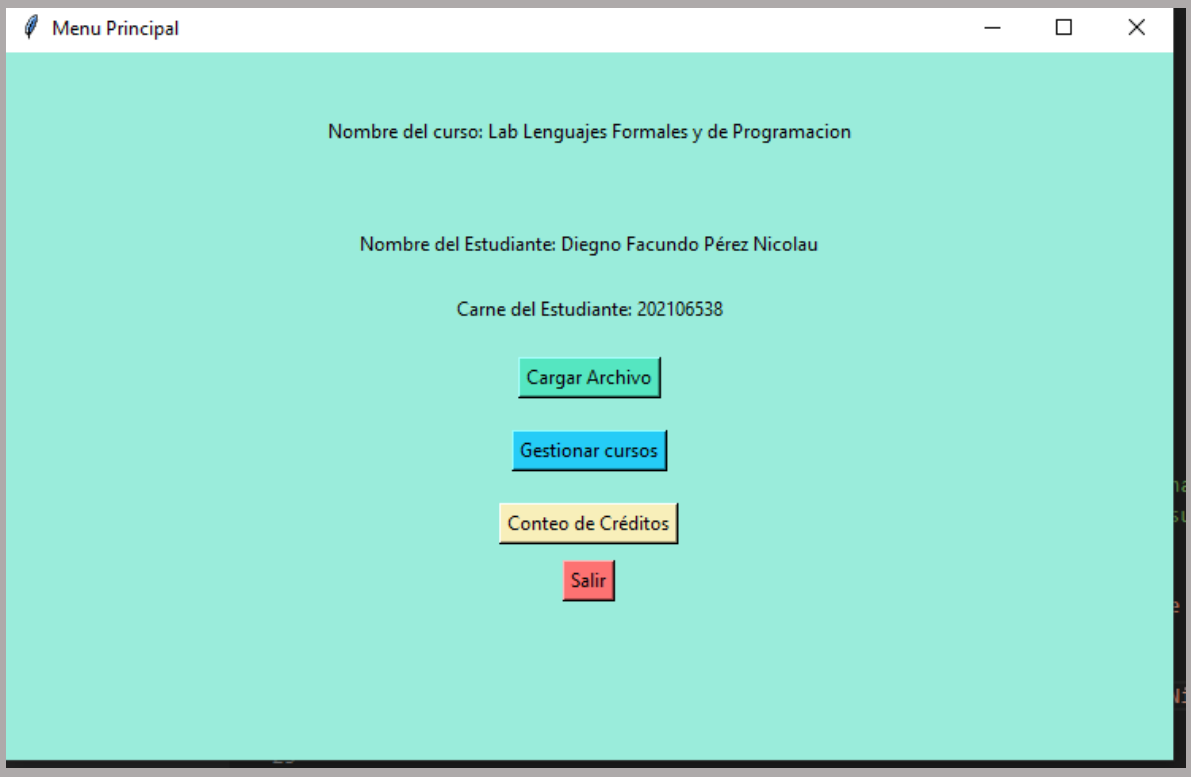
La presente practica tuvo como finalidad evaluar el manejo de sistemas con este ejemplo de ver controlar os cursos que se tiene y con estos sus datos y las posibilidades de organización con estas herramientas:

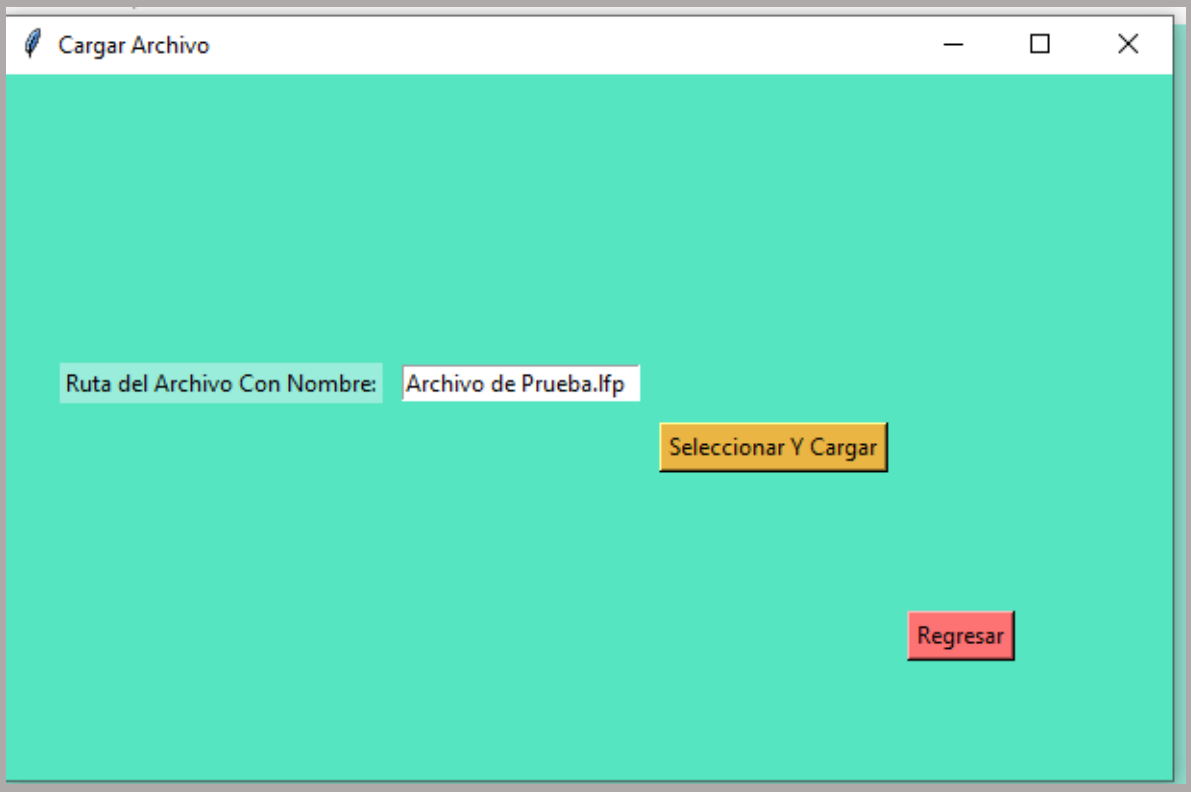
En esta practicas se utilizo la interfaz de visual studio code por la versatilidad y múltiples opciones que tiene para ayudar al trabajo del programador.

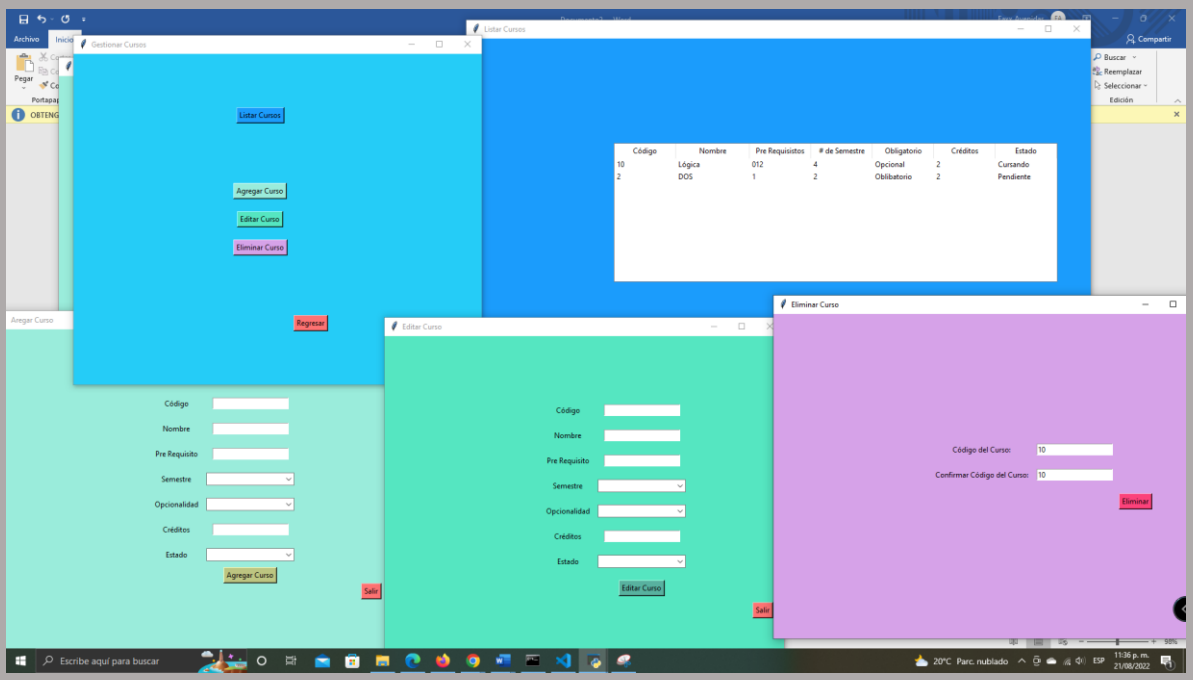
Para la interfaz grafica se uso tkinter porque es la más básica y usada, ideal para empezar y por qué será usada en el futuro.

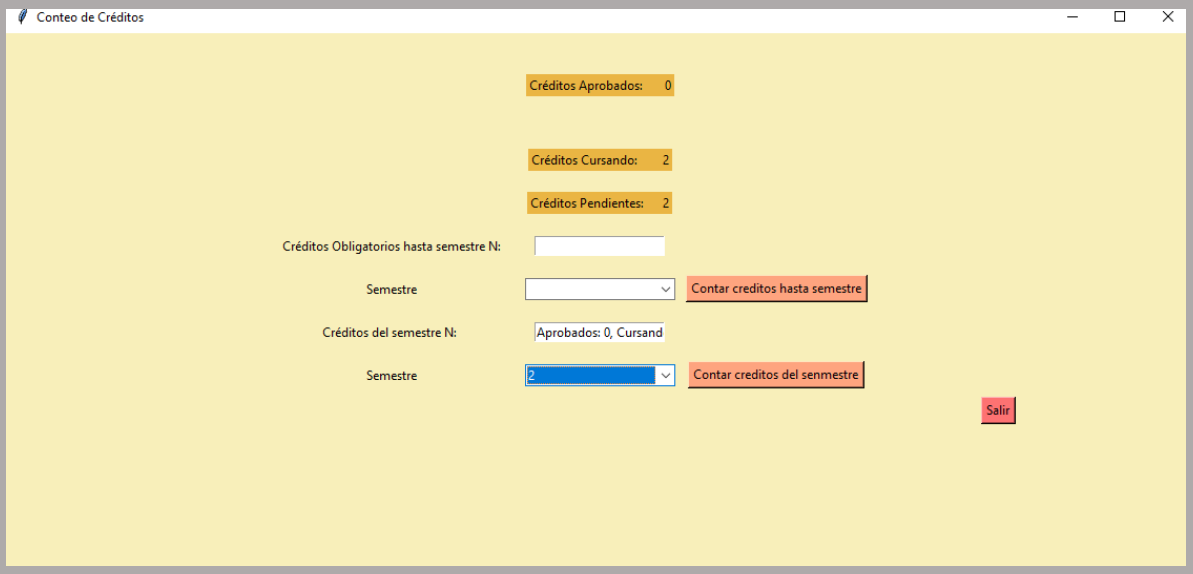
Interfaces y validaciones

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 readNum(num):
    if(type(num)==int):
        return num
    else:
        try:
            return int(num)
        except:
            print("¡¡Entrada fallida, se ingreso un valor que no es un numero!!")
```

```python
def  getTextoObligatorio(self):
    n=self.getObligatorio()
    if (n==1):
        return "Oblibatorio"
    elif (n==0):
        return "Opcional"
    else:
        return "ERROR NUMERO DE OBLIGATORIO NO VALIDO"

def  getTextoEstado(self):
    estado=self.getEstado()
    if (estado==0):
        return "Aprobado"
    elif (estado==1):
        return "Cursando"
    elif (estado==-1):
        return "Pendiente"
    else:
        return "ERROR NUMERO DE ESTADO NO VALIDO"
```