



REGISTRO DE CURSOS

Practica 1 LFP



Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Laboratorio de Lenguajes Formales de Programación
Sección A-
Aux. Mario
Manual Técnico



Introducción

En el presente manual se pretende explicar de manera detallada la funcionalidad del programa de Registro de Cursos, especificando cada función de botones, paneles, entradas de texto, entre otras cosas.

En la implementación del programa Registro de Cursos se busca otorgar listados ordenados y estructurados de los cursos, detallando sus créditos, semestre, opcionalidad, entre otras cosas; así de esa manera los estudiantes podrán llevar un control más organizado y sencillo al momento de asignarse sus respectivos cursos.

Objetivos



Permitir a las personas que están asociadas al ámbito de la programación en Python comprender la estructura y funcionalidad del programa, así como la lógica utilizada para realización del mismo.

Requerimientos:

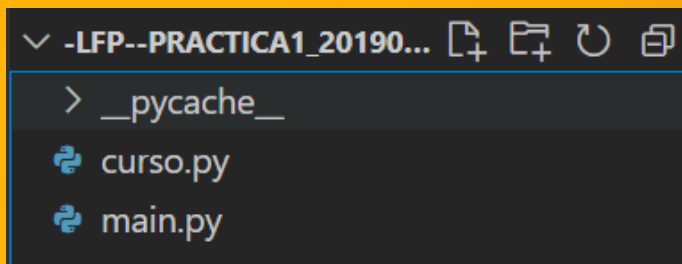
CPU:	Intel de 64 bits, Recomendado: Intel Core i3 o superior
RAM:	4GB, Recomendado: 8 GB o más
Espacio disponible en disco:	8GB
Sistema Operativo:	Windows 10
Resolución de Pantalla:	Básico: pantalla de 1280 x 800
Programas Adicionales	Python 3.10.5 - Python 3.10.6 - Python 3.10.4



main.py

main

En el archivo main.py se encuentra toda la lógica del programa como las funciones, validaciones, entre otras cosas. Así también se encuentran los elementos de la interfaz gráfica, como lo son las ventanas, botones, cuadros de texto, etc.



Curso.py

También se podrá encontrar el archivo curso.py, donde se encuentra la clase Curso, así como sus respectivos atributos: código, nombre, prerequisites, opcionalidad, semestre, créditos, estado.

```
curso.py > Curso > __init__
1 class Curso():
2     def __init__(self, codigo, nomb
3         self.codigo = codigo
4         self.nombre = nombre
5         self.prerrequisitos = prerr
6         self.opcionalidad = opciona
7         self.semestre = semestre
8         self.creditos = creditos
9         self.estado = estado
```



Interfaz Gráfica

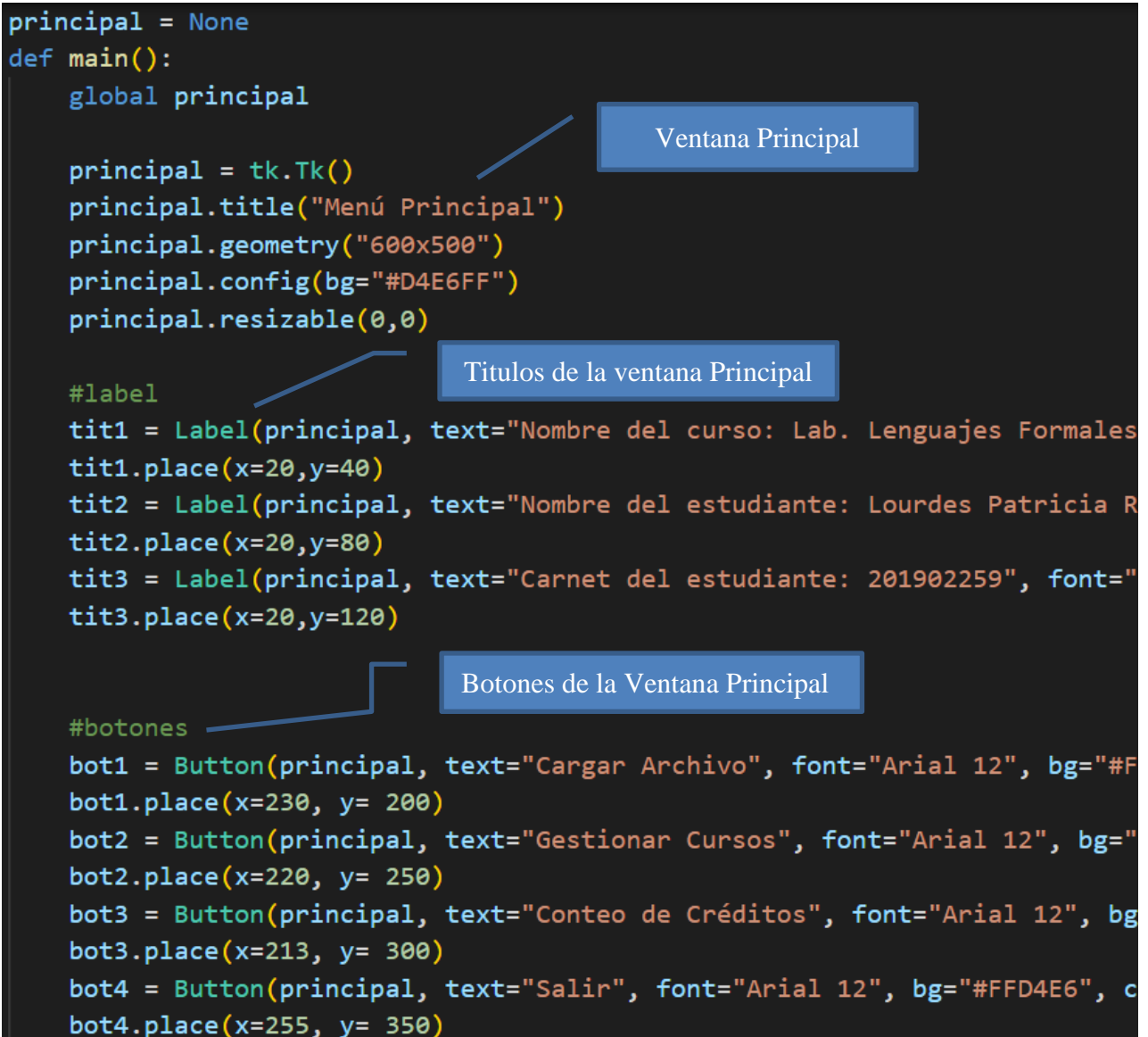
En la interfaz gráfica se pueden encontrar distintos elementos, como botones, textbox, ventanas, label, etc. Para estos se utilizaron códigos del siguiente tipo:

```
principal = None
def main():
    global principal

    principal = tk.Tk()
    principal.title("Menú Principal")
    principal.geometry("600x500")
    principal.config(bg="#D4E6FF")
    principal.resizable(0,0)

    #label
    tit1 = Label(principal, text="Nombre del curso: Lab. Lenguajes Formales")
    tit1.place(x=20,y=40)
    tit2 = Label(principal, text="Nombre del estudiante: Lourdes Patricia R")
    tit2.place(x=20,y=80)
    tit3 = Label(principal, text="Carnet del estudiante: 201902259", font="")
    tit3.place(x=20,y=120)

    #botones
    bot1 = Button(principal, text="Cargar Archivo", font="Arial 12", bg="#F")
    bot1.place(x=230, y= 200)
    bot2 = Button(principal, text="Gestionar Cursos", font="Arial 12", bg="")
    bot2.place(x=220, y= 250)
    bot3 = Button(principal, text="Conteo de Créditos", font="Arial 12", bg="")
    bot3.place(x=213, y= 300)
    bot4 = Button(principal, text="Salir", font="Arial 12", bg="#FFD4E6", c)
    bot4.place(x=255, y= 350)
```



Cargar Archivo

Se realizó una función que permite cargar el archivo y buscar la ruta.

```
def Carga():
    root = tk.Tk()
    root.withdraw()
    ruta = filedialog.askopenfilename(title="Carga de", filetypes=[
    if ruta != "":
        archivo = open(ruta, 'r', encoding='utf-8')
        contenido = archivo.read()
        leerarchivo(contenido)
    else:
        messagebox.showinfo("Error", "Seleccione un archivo")
```

Función que lee el archivo y eliminar los espacios vacíos para leer línea por línea.

```
def leerarchivo(contenido):
    global cursos
    contenido1=contenido.split('\n')
    for linea in contenido1:
        if linea != "":
            contenido2=linea.split(",")
            if verificarcurso(contenido2):
                nuevo = Curso(contenido2)
                cursos.append(nuevo)
```


Gestionar Cursos

Listar Cursos:

Se crea una lista para los cursos y se le agregan los cursos junto a sus atributos, adicional se insertan en la table que se encuentra en la Ventana de Listar Cursos.

```
lista = []
for c in cursos:
    lista.append((c.codigo,c.nombre,c.prerrequisitos,c.opcionalidad,c.semestre))
for num in lista:
    tabla.insert("", tk.END, values=num)
tabla.grid(row=0, column=0, sticky="nsnew")
scroll = ttk.Scrollbar(listarc, orient=tk.VERTICAL, command=tabla.yview)
tabla.configure(yscroll=scroll.set)
scroll.grid(row=0, column=1, sticky="ns", pady=8)
tabla.place(x=20, y=20)
scroll.place(x=0,y=20)
```

Mostrar Cursos:

En la función mostrar_cursos se recibe como parametro el codigo del curso y se verifica que su codigo exista para luego mostrar los demás atributos.

```
def mostrar_curso(codigo, nombre, prerrequisitos, semestre, opcionalidad, creditos, estado):
    global cursos
    if codigo != "":
        if verificarcurso(codigo)==True:
            for c in cursos:
                if codigo == c.codigo:
                    nombre["state"]="normal"
                    nombre.delete(0,"end")
                    nombre.insert(0, c.nombre)
                    nombre["state"]="disable"
```

Gestionar Cursos

Agregar Curso:

En la función `agregar_curso`, se reciben como parámetros los atributos del curso y se verifica que no haya ningún campo vacío para luego agregar ese curso a la lista.

```
def agregar_curso(codigo, nombre, prerequisitos, semestre, opcionalidad,
global cursos
if codigo != "" and nombre!= "" and semestre!="" and opcionalidad!="":
    if (verificarcurso(codigo)==False):
        nuevo = Curso(codigo, nombre, prerequisitos, opcionalidad, se
        cursos.append(nuevo)
        messagebox.showinfo("Agregado!", "Curso agregado con éxito")
    else:
        messagebox.showinfo("Error", "El curso ya existe en la lista")
else:
    messagebox.showinfo("Error", "Llene todos los campos requeridos")
```

Editar Curso:

En esta función se verifica que el curso exista por medio de su código y se recorre la lista en ese curso para modificar los nuevos atributos.

```
def editar_curso(codigo, nombre, prerequisito
global cursos
if codigo != "" and nombre!= "" and semest
    if (verificarcurso(codigo)==True):
        for c in cursos:
            if c.codigo==codigo:
                c.nombre=nombre
                c.prerequisistos=prerrequ
                c.semestre=semestre
                c.opcionalidad=opcionalida
                c.creditos=creditos
                c.estado=estado
            messagebox.showinfo("Actualizado!"
        else:
            messagebox.showinfo("Error", "El c
    else:
        messagebox.showinfo("Error", "Llene to
```


Gestionar Cursos

Eliminar Curso:

En esta función se recibe como parámetro el Código del curso y se verifica que el mismo exista para luego recorrer la lista y al encontrarlo se usa un remove para eliminarlo.

```
def eliminar_curso(codigo):  
    global cursos  
    if codigo != "":  
        if verificarcurso(codigo)==True:  
            for i in range(len(cursos)):  
                if codigo == cursos[i].codigo:  
                    cursos.remove(cursos[i])  
                    messagebox.showinfo("Eliminado!", "C  
        else:  
            messagebox.showinfo("Error", "El curso no se  
    else:  
        messagebox.showinfo("Error", "Llene todos los ca
```

Conteo de Créditos

Créditos aprobados, cursando y pendientes

En la función contar_creditos se crean los respectivos contadores de cursos y luego se recorre la lista de cursos, se utilizan los condicionales para especificar que cada vez que se encuentra en el estado respectivo 0,1 o -1, el contador del mismo aumente. Luego se configuran los label de acuerdo a los contadores.

Por ejemplo si al recorrer la lista se encuentra el curso de Física 1, y este tiene el estado "0", el Contador de aprobados aumentará su valor con una suma de los créditos que corresponden a dicho curso, y al finalizar el recorrido de la lista se mostrarán en el label1.

```
def contarcreditos(label1, label2, label3):  
    global cursos  
    contador_aprobados = 0  
    contador_pendientes = 0  
    contador_cursando = 0  
    for c in cursos:  
        if c.estado == "0":  
            contador_aprobados+=int(c.creditos)  
        elif c.estado == "1":  
            contador_cursando+=int(c.creditos)  
        elif c.estado == "-1":  
            contador_pendientes+=int(c.creditos)  
    label1.config(text=str(contador_aprobados))  
    label2.config(text=str(contador_cursando))  
    label3.config(text=str(contador_pendientes))
```

Conteo de Créditos

Créditos hasta semestre N:

En esta función se recibe como parámetro el semestre y se crea un Contador de créditos y un Contador de semestre, sin embargo este Contador de semestre inicia en uno puesto que no existe un semestre cero, se recorre la lista y si la opcionalidad es obligatoria, se proceden a verificar que ese curso se encuentre en el rango del semestre 1 hasta el semestre N, para posteriormente agregar al Contador de créditos, los créditos de dicho curso.

```
def contarcred_semestre(semestre, textbox):
    global cursos
    if semestre!="":
        contador_creditos = 0
        for c in cursos:
            contador_semestre = 1
            if c.opcionalidad == "1":
                for i in range(contador_semestre, int(semestre)+1):
                    if int(c.semestre)== i:
                        contador_creditos+=int(c.creditos)

        textbox["state"]="normal"
        textbox.delete(0,"end")
        textbox.insert(0, str(contador_creditos))
        textbox["state"]="disable"
    else:
        messagebox.showinfo("Error", "Seleccione el número de semestre")
```


Conteo de Créditos

Créditos del Semestre:

En esta función se tienen los 3 contadores de estado del curso, y se verifica que el semestre exista, entonces se recorre la lista y se verifica qué tipo de opcionalidad tiene y si este pertenece al semestre seleccionado para posteriormente añadir los créditos al Contador correspondiente.

```
def contar_semestre(semestre, label1, label2, label3):
    contador_aprobados = 0
    contador_pendientes = 0
    contador_cursando = 0
    global cursos
    if semestre!="":
        for c in cursos:
            if c.estado == "0" and c.semestre==semestre:
                contador_aprobados+=int(c.creditos)
            elif c.estado == "1" and c.semestre==semestre:
                contador_cursando+=int(c.creditos)
            elif c.estado == "-1" and c.semestre==semestre:
                contador_pendientes+=int(c.creditos)
        label1.config(text="Creditos aprobados: "+str(contador_aprobados))
        label2.config(text="Creditos cursando: "+str(contador_cursando))
        label3.config(text="Creditos pendientes: "+str(contador_pendientes))
    else:
        messagebox.showinfo("Error", "Seleccione el número de semestre")
```

Otros

Funciones Regresar

Se crearon funciones para cada botón de regresar, para que de esa manera se pudiera cerrar la Ventana actual, para abrir la anterior.

```
def regresar7():
    global moscurso
    moscurso.destroy()
    gestion()

def regresar4():
    global editcurso
    editcurso.destroy()
    gestion()
```

Lista de cursos

```
cursos = []

def gestionarr():
    principal.destroy()
    gestion()

def cargar():
    principal.destroy()
    seleccionar()

def salir():
    principal.destroy()

def agregarc():
    gestionar.destroy()
    agregarcurso()

def elic():
    gestionar.destroy()
    eliminar()

def conteoc():
    principal.destroy()
    cc()

def mosc():
    gestionar.destroy()
    mostrarcursos()
```



CONCLUSIONES

Se puede afirmar que la realización del programa otorgó distintos conocimientos a los estudiantes, puesto que se aplicaron conceptos relacionados con los lenguajes formales y se lograron conocer distintas características de la programación en Python así como aplicar algoritmos para el ordenamiento de datos.

