MANUAL TÉCNICO

Práctica No.1

Por: Mario Ernesto Marroquín Pérez-202110509

CONTENIDO

INTRODUCCION

El programa ha sido desarrollado en el lenguaje Python, este software cuenta con distintas opciones para desplegar la información de un pensum de estudios, facilitando la forma visual de cada uno de ellos para saber así qué cursos puede llevar durante un semestre o escuela de vacaciones.

El programa se encuentra organizado y estructurado por funciones que retornan un valor y variables globales.

LIBRERIAS

Dentro del programa se importa una serie de librerías las cuales son utilizadas para la lectura de datos, la creación de la GUI (Interfaz Gráfica de Usuario por sus siglas en inglés), la lectura de datos duplicados en listas. Estas librerías se importan utilizando la palabra “import”.

Practica1_202110509.py > ...

```
1   import code
2   from faulthandler import disable
3   from inspect import Traceback
4   from re import X
5   import tkinter
6   import csv
7   import os
8   from tkinter import BOTTOM, HORIZONTAL, LEFT, RIGHT, VERTICAL, Y, Scrollbar, messagebox
9   from tkinter import ttk
10  from traceback import TracebackException
11  from typing import Literal
12  from xml.dom import NoModificationAllowedErr
13  from iteration_utilities import duplicates
14  from setuptools import Command
```

CREACION DE VENTANAS

Para la creación de cada ventana se sigue siempre la misma estructura o algoritmo, se indica que para la creación de la ventana se utilizará tkinter y posteriormente se agrega la respectiva configuración; el tamaño de la ventana, título de la ventana, así como indicar si se podrá o no variar el tamaño de la ventana durante la ejecución del programa. Para cada una de las ventanas se realizó de la misma manera.

```
18  #Configuraciones ventana principal
19      ventana = tkinter.Tk()
20      ventana.geometry("450x330")
21      ventana.title("Practica1")
22      ventana.resizable(False,False)
23
```

CREAR ELEMENTOS DENTRO DE LA VENTANA

Para crear elementos (botones, texto, combobox, entradas de texto), se declaran utilizando la palabra tkinter, esto para cada elemento que se encuentra dentro de una ventana y que se desea crear dentro de todo el programa, cada elemento del programa sigue la misma estructura y lógica para su creación.

```
# Crear y Configurar elementos dentro de la ventena de inicio
curso = tkinter.Label(ventana, text="Nombre del curso: Lab. Lenguajes Formales y de Programación")
nombre = tkinter.Label(ventana, text="Nombre del Estudiante: Mario Ernesto Marroquín Pérez")
carnet=tkinter.Label(ventana, text="Carnet del Estudiante: 202110509")
cargarArchivo=tkinter.Button(ventana,text="Cargar Archivos", command=seleccionar)
gestionarCurso=tkinter.Button(ventana,text="Gestionar Cursos", command=gestionar)
conteoCreditos=tkinter.Button(ventana,text="Conteo de Créditos", command=conte)
salir =tkinter.Button(ventana,text="Salir",command=ventana.destroy)
```

ORGANIZAR ELEMENTOS DENTRO DE LA VENTANA

Para organizar todos los elementos creados dentro de una ventana se configura utilizando la palabra “place”, en donde indicamos una serie de parámetros, en específico la posición en el eje “x” y la posición en el eje “y”, tomando en cuenta que esta posición se indica en pixeles, específicamente la posición se escribe de la siguiente manera: “elemento.place(x=pixel, y=pixel)”. Para cada elemento dentro de una ventana en el programa se realizó de la misma manera, todos tienen la misma lógica y algoritmo.

```
#organizar los elementos pantalla principal
curso.place(x=50, y=30)
nombre.place(x=50, y=60)
carnet.place(x=50, y=90)
cargarArchivo.place(x=170, y=150)
gestionarCurso.place(x=167, y=190)
conteoCreditos.place(x=160, y=230)
salir.place(x=195, y=270)
```

MANTENER UNA VENTANA FUNCIONANDO

Para que una ventana se mantenga funcionando se utiliza la siguiente declaración, esto para cada ventana dentro del programa.

```
#mantener la ventana funcionando
ventana.mainloop()
```

FUNCIONES DENTRO DEL PROGRAMA

Para la creación del programa se hizo uso de funciones, estas funciones realizan una ejecución en específico; ya sea para volver a la pantalla anterior, capturar la ruta del archivo ingresado por el usuario, realizar la sumatoria de los créditos, mostrar datos en la tabla entre muchas otras.

Función Volver( ):

Esta función nos permite regresar a la pantalla anterior, muy útil para navegar entre ventanas, su lógica se basa en enviarnos a la función perteneciente a la ventana anterior y destruir la ventana actual.

```
def volver():
    ventana.destroy()
    principal()
```

Función Llevar( ):

Estas funciones no están llamadas explícitamente “Llevar”, sino que esta palabra esta siendo utilizada para explicar su funcionamiento, ya que estas funciones son las encargadas de llevarnos de una ventana a otra, para que exista navegación entra ventanas, la lógica de estas funciones es similar a la función “volver”, ya que nos llevan a la nueva ventana y destruyen la ventana anterior. Este funcionamiento es el mismo para cada función que nos lleva de una ventana a otra, misma lógica y algoritmo.

```
def listarC():
    ventana.destroy()
    listar()

def agregarC():
    ventana.destroy()
    agregar()

def editarC():
    ventana.destroy()
    editar()

def eliminarC():
    ventana.destroy()
    eliminar()
```

```
def seleccionar():
    ventana.destroy()
    seleccionarArchivo()

def gestionar():
    ventana.destroy()
    gestionarCursos()

def conte():
    ventana.destroy()
    conteoCredito()
```

Función obtenerRuta( ):

La lógica de esta función consiste en obtener la ruta del archivo que se desea ingresar al sistema, una vez obtenida la ruta esta función devuelve dicha ruta.

```
def obtenerRuta():
    rutaArchivo=rutaTexto.get()
    return rutaArchivo
```

Función imprimir( ):

Esta función es la encargada de leer dicho archivo ingresado y mostrar un mensaje en pantalla si el archivo ha sido cargado al sistema o si ha ocurrido algún error durante la carga. Así mismo esta función está dentro de un try-except para evitar un error de ingreso de ruta y evitar que el programa termine el proceso si no que muestre un mensaje en pantalla que se ha ingresado mal la ruta.

```
def imprimir():
    try:
        csv_file = open(f'{obtenerRuta()}','r')
        global reader
        reader = csv.reader(csv_file,delimiter=",")

        if obtencion() !=1:
            messagebox.showinfo(message="Carga completada!", title="Felicidades")
            volver()

        if obtencion() ==1:
            messagebox.showwarning(message="EXISTEN CURSOS REPETIDOS DENTRO DEL ARCHIVO o YA SE HA CARGADO UN ARCHIVO", title="ADVERTENCIA")
            volver()

    except FileNotFoundError:
        messagebox.showerror(message="Error, ingrese una direccion adecuada", title="ERROR")
    except OSError:
        messagebox.showerror(message="Error, ingrese una direccion adecuada", title="ERROR")
```

Función obtencion( ):

Esta función es la encargada de obtener los datos leídos del archivo cargado al programa y almacenarlos en una lista, así mismo esta función verifica si hay datos repetidos dentro del archivo ingresado y devuelve un mensaje indicando dicho error, esta función también se encuentra dentro de un try-except para evitar un error de lectura del archivo y evitar que el programa termine la ejecución si no que muestre un mensaje en pantalla indicando el error.

```
cursos=[]
codigosIngre=[]
nombresIngre=[]


def obtencion():
    try:
            global cursos
            global codigosIngre
            global nombresIngre
            for columnas in reader:
                codigo = columnas [0]
                nombre = columnas [1]
                preRequisito=columnas [2]
                semestre=int(columnas [3])
                opcionalidad=int(columnas [4])
                creditos=int(columnas[5])
                estados=int(columnas [6])
                cursos.append({
                    "co":codigo,
                    "nom":nombre,
                    "pre":preRequisito,
                    "sem":semestre,
                    "op":opcionalidad,
                    "cred":creditos,
                    "es":estados,
                })
                nombresIngre.append({
                    "nomb":nombre,
                })
                codigosIngre.append({
                    "codi": codigo,
                })
            duplicados = list(duplicates(nombresIngre))
            duplicados1 = list(duplicates(codigosIngre))
            if len(duplicados)>0 or len(duplicados1)>0:
                cursos.clear()
                return 1
            else:
                return cursos
    except ValueError:
            messagebox.showerror(message="POR FAVOR NO MEZCLAR LOS APARTADOS NUMERICOS CON LETRAS, verifique el archivo, Se HAN CARGADO LOS CURSOS SIN PROBLEMAS", title="ERROR")
            return 1
```

Función agregacion( ):

Esta función es la encargada de agregar los cursos uno a uno, esta función solo se utiliza dentro de la ventana “agregar curso”. La función se encarga de verificar que los datos sean ingresados de manera correcta y no de manera errónea, así mismo verifica que se ingresen los datos de manera correcta y no mezclar números con letras, la función también verifica que no se ingrese un curso ya cargado, si la función detecta esto le indica al usuario que se dirija a la ventana de edición del curso.

Si no se presenta ningún error la función agrega el curso a la lista donde se encuentran los cursos previamente guardados, una vez ya guardado el curso se podrá visualizar así como manipular dentro del programa.

```
def agregacion():
    try:
        curso = rutaCodigo.get()
        nombre = rutaNombre.get()
        preRequisito = rutaPreRequisito.get()
        semestre = int (rutaSemestre.get())
        opcion = int(rutaOpcionalidad.get())
        creditos = int(rutaCreditos.get())
        estado = int(rutaEstado.get())

        if (semestre >= 1 and semestre <=10):
            if (opcion==1 or opcion==0):
                if(estado == -1 or estado == 0 or estado ==1):
                    nombresIngre.append({
                                "nomb":nombre,
                            })
                    codigosIngre.append({
                                "codi": curso,
                            })
                    duplicados = list(duplicates(nombresIngre))
                    duplicados1 = list(duplicates(codigosIngre))

                    if len(duplicados)>0 or len(duplicados1)>0:
                        duplicados.clear()
                        duplicados1.clear()
                        messagebox.showwarning(message="ESTE CURSO YA EXISTE, vaya al apartado de edición", title="Ya existe!")

                    else:
                        cursos.append({
                                    "co":curso,
                                    "nom":nombre,
                                    "pre":preRequisito,
                                    "sem":opcion,
                                    "op":semestre,
                                    "cred":creditos,
                                    "es":estado,
                                })
                        messagebox.showinfo(message="Curso Agregado!", title="Felicidades")

                        volver()
                else:
                    messagebox.showerror(message="INGRESE LOS DATOS CORRECTOS en el apartado de ESTADO", title="ERROR")
            else:
                messagebox.showerror(message="INGRESE LOS DATOS CORRECTOS en el apartado de OPCIONALIDAD", title="ERROR")
        else:
            messagebox.showerror(message="INGRESE LOS DATOS CORRECTOS en el apartado de SEMESTRE", title="ERROR")

    except ValueError:
        messagebox.showerror(message="POR FAVOR NO MEZCLAR LOS APARTADOS NUMERICOS CON LETRAS", title="ERROR")
```

Función editar( ):

La función primero obtiene los datos ingresados dentro de la ventana y verifica que el archivo exista para poder editarlo, si el archivo no existe salta un error indicando que ese código de curso no existe, si el curso existe la función evalúa que los datos se hayan ingresado de la forma correcta y no se estén mezclando datos y que estos tengan sentido, así mismo la función está dentro de un try-except para verificar si existe un error cuando se ingresan los datos y así saltar un error indicando el error al editar el curso.

Si no existe ningún error, la función busca dentro de la lista de cursos y edita dicho curso, se pueden editar todos los valores del curso a EXCEPCION del código del curso, será el único identificador que no podrá ser alterado.

```
def editar():
    try:
        curso = rutaCodigo.get()
        nombre = rutaNombre.get()
        preRequisito = rutaPreRequisito.get()
        semestre = int (rutaSemestre.get())
        opcion = int(rutaOpcionalidad.get())
        creditos = int(rutaCreditos.get())
        estado = int(rutaEstado.get())

        if (semestre >= 1 and semestre <=10):
            if (opcion==1 or opcion==0):
                if(estado == -1 or estado == 0 or estado ==1):
                    for nom in cursos:
                        if nom["co"] == curso:
                            nom["nom"]=nombre
                            nom["pre"]=preRequisito
                            nom["sem"]=opcion
                            nom["op"]=semestre
                            nom["cred"]=creditos
                            nom["es"]=estado
                            messagebox.showinfo(message="Se ha actualizado el curso", title="Felicidades!")
                            volver()
                            break
                    messagebox.showerror(message="Este curso NO existe", title="ERROR")
                else:
                    messagebox.showerror(message="INGRESE LOS DATOS CORRECTOS en el apartado de ESTADO", title="ERROR")
            else:
                messagebox.showerror(message="INGRESE LOS DATOS CORRECTOS en el apartado de OPCIONALIDAD", title="ERROR")
        else:
            messagebox.showerror(message="INGRESE LOS DATOS CORRECTOS en el apartado de SEMESTRE", title="ERROR")

    except ValueError:
        messagebox.showerror(message="POR FAVOR NO MEZCLAR LOS APARTADOS NUMERICOS CON LETRAS", title="ERROR")
```

Función eliminar( ):

Esta función se encarga de obtener el código del curso que se desea eliminar, una vez obtenido el código la función se encarga de verificar que el curso exista dentro de la lista de cursos, si este curso existe la función procede a eliminar dicho curso ingresado.

Si el curso ingresado no existe dentro de la lista de cursos cargados al sistema la función devuelve un error indicando que dicho curso ingresado no se encuentra en la memoria.

```
def elimir():
    elim = rutaCodigo.get()
    i=0
    for nom in cursos:
        i=i+1
        if nom["co"] == elim:
            cursos.pop(i-1)
            messagebox.showinfo(message="Se ha eliminado el curso", title="Felicidades!")
            volver()
            break
    messagebox.showerror(message="Este curso NO existe", title="ERROR")
```

Funciones sumarCreditos( ):

Estas funciones son las encargadas de realizar la suma de los créditos evaluando las condiciones dadas en cada función, cada función recorre la lista de cursos buscando los cursos que cumplan con las condiciones dadas y suma sus créditos para luego mostrar dicha suma en pantalla.

```
def creditosHasta():
    sumatoria =0
    semestre1 = int(combo.get())
    for suma in cursos:
                semestres = int(suma["op"])
                if semestres <= semestre1:
                    if suma["sem"] == 1:
                        sumatoria += suma["cred"]
    numeroTotalCreditos.config(text=sumatoria)
    return sumatoria

def mostrarCreditosHasta():
    aprovados=0
    cursando =0
    pendientes=0
    semestre1 = int(combo1.get())
    for suma in cursos:
        semestres = int (suma["op"])
        if semestres <= semestre1:
            if suma["es"] == 0:
                aprovados += suma["cred"]
            if suma["es"]== 1:
                cursando += suma["cred"]
            if suma ["es"]==-1:
                pendientes += suma["cred"]
    numeroTotalCreditos1.config(text = aprovados)
    numeroTotalCreditos2.config(text=cursando)
    numeroTotalCreditos3.config(text=pendientes)
```

```
def sumarAprobados():
    sumatoria = 0
    for suma in cursos:
        if suma["es"] == 0:
            sumatoria += suma["cred"]
    return sumatoria

def sumarCursando():
    sumatoria = 0
    for suma in cursos:
        if suma["es"] == 1:
            sumatoria += suma["cred"]
    return sumatoria

def sumarPendientes():
    sumatoria = 0
    for suma in cursos:
        if suma["es"] == -1:
            if suma["sem"] == 1:
                sumatoria += suma["cred"]
    return sumatoria
```

TABLA DENTRO DEL PROGRAMA

Primero se define que se trabajará con una tabla en tkinter, indicamos que la tabla estará en la ventana e indicamos que tendrá un número específico de columnas, indicamos el número y ancho de cada columna, así como el encabezado de cada una de ellas, indicamos su ubica dentro de la ventana y para imprimir cada curso con sus elementos la función recorre la lista de cursos e inserta en la tabla cada curso.

```
tabla = ttk.Treeview(ventana, columns=("#1","#2","#3","#4","#5","#6"))


tabla.column("#0",width=50)
tabla.column("#1",width=200)
tabla.column("#2",width=150)
tabla.column("#3",width=80)
tabla.column("#4",width=80)
tabla.column("#5",width=80)
tabla.column("#6",width=80)


tabla.heading("#0",text="Código")
tabla.heading("#1",text="Nombre")
tabla.heading("#2",text="Pre requisito")
tabla.heading("#3",text="Opcionalidad")
tabla.heading("#4",text="Semestre")
tabla.heading("#5",text="Créditos")
tabla.heading("#6",text="Estado")

tabla.pack()
regresar.pack()


curso = cursos
for alumno in curso:
        co = alumno["co"]
        nom = alumno["nom"]
        pre = alumno["pre"]
        sem = alumno["sem"]
        op =alumno["op"]
        cr=alumno["cred"]
        es=alumno["es"]
        tabla.insert("",0,text=(co),values=(nom, pre, sem, op, cr, es))
```