ITESO, Universidad Jesuita de Guadalajara

Programa: “Friendly Timetable”
Equipo: “Minimal Effort”
Miembros: Naim Towfighian Ortiz y Manuel Antonio Rodriguez

Documentación técnica

Profesor: Gonzalo de Jesus Guzman Jauregui

DOCUMENTACIÓN TÉCNICA:

Detalle y explicación de Algoritmos de nuestro programa por medio de funciones y los métodos implementados.

Función 1: color_tema1

En esta función se corre cuando se selecciona el botón del primer tema. Lo que hace la función es cambiar el color de los componentes del programa. Esto lo hace al abrir el archivo acolor.txt, el cual tiene los colores del programa, lo borra por completo y lo reescribe con la información del nuevo color que va a tener, de esta manera al volver a correr el programa va a iniciar con los colores: LightCyan4, DarkSlateGray3 y PaleTurquoise1.

```
17  def color_tema1():
18      archivo_color = open("acolor.txt", "w")
19      archivo_color.write("LightCyan4\n")
20      archivo_color.write("DarkSlateGray3\n")
21      archivo_color.write("PaleTurquoise1")
22      archivo_color.close()
```

Función 2: color_tema2

En esta función se corre cuando se selecciona el botón del segundo tema. Lo que hace la función es cambiar el color de los componentes del programa. Esto lo hace al abrir el archivo acolor.txt, el cual tiene los colores del programa, lo borra por completo y lo reescribe con la información del nuevo color que va a tener, de esta manera al volver a correr el programa va a iniciar con los colores: gray25, MistyRose2 y SteelBlue3.

```
26  def color_tema2():
27      archivo_color = open("acolor.txt", "w")
28      archivo_color.write("gray25\n")
29      archivo_color.write("MistyRose2\n")
30      archivo_color.write("SteelBlue3")
31      archivo_color.close()
```

Función 3: color_tema3

En esta función se corre cuando se selecciona el botón del tercer tema. Lo que hace la función es cambiar el color de los componentes del programa. Esto lo hace al abrir el archivo acolor.txt, el cual tiene los colores del programa, lo borra por completo y lo reescribe con la información del nuevo color que va a tener, de esta manera al volver a correr el programa va a iniciar con los colores: gray80, CadetBlue1 y DarkOrchid3.

```
35  def color_tema3():
36      archivo_color = open("acolor.txt", "w")
37      archivo_color.write("gray80\n")
38      archivo_color.write("CadetBlue1\n")
39      archivo_color.write("DarkOrchid3")
40      archivo_color.close()
```

Función 4: color_tema4

En esta función se corre cuando se selecciona el botón del cuarto tema. Lo que hace la función es cambiar el color de los componentes del programa. Esto lo hace al abrir el archivo acolor.txt, el cual tiene los colores del programa, lo borra por completo y lo reescribe con la información del nuevo color que va a tener, de esta manera al volver a correr el programa va a iniciar con los colores: VioletRed4, DeepPink3, y HotPink1.

```
44  def color_tema4():
45      archivo_color = open("acolor.txt", "w")
46      archivo_color.write("VioletRed4\n")
47      archivo_color.write("DeepPink3\n")
48      archivo_color.write("HotPink1")
49      archivo_color.close()
```

Función 5: color_tema5

En esta función se corre cuando se selecciona el botón del quinto tema. Lo que hace la función es cambiar el color de los componentes del programa. Esto lo hace al abrir el archivo acolor.txt, el cual tiene los colores del programa, lo borra por completo y lo

reescribe con la información del nuevo color que va a tener, de esta manera al volver a correr el programa va a iniciar con los colores: wheat4, coral1, y burlywood1.

```
53  def color_tema5():
54      archivo_color = open("acolor.txt", "w")
55      archivo_color.write("wheat4\n")
56      archivo_color.write("coral1\n")
57      archivo_color.write("burlywood1")
58      archivo_color.close()
```

Función 6: valores_de_color

Esta función se corre al principio del programa. Esta función hace que que los colores del archivo acolor.txt se guarden en variables para poder ser utilizados más adelante para widgets. Para ello, se abre el archivo acolor.txt, se lee línea por línea y la parte del archivo que dice el color se guarda en una variable global. Esto se hace tres veces, una vez por color lo que nos da tres colores distintos a utilizar. También esta función guarda en una variable el ícono del usuario.

```
63  def valores_de_color():
64      global color1, color2, color3, Icono
65      archivo_color=open("acolor.txt","r")
66      color1_en_archivo = str(archivo_color.readlines(1))
67      color1 = color1_en_archivo[2:len(color1_en_archivo)-4]
68
69      color2_en_archivo = str(archivo_color.readlines(2))
70      color2 = color2_en_archivo[2:len(color2_en_archivo)-4]
71
72      color3_en_archivo = str(archivo_color.readlines(3))
73      color3=color3_en_archivo[2:len(color3_en_archivo)-2]
74      archivo_color.close()
75
76      #Iconos
77      archivo_icono=open("icon.txt","r")
78      Icono_en_archivo= str(archivo_icono.readlines(1))
79      Icono= Icono_en_archivo[2:len(Icono_en_archivo)-2]
80
81      archivo_icono.close()
```

Función 7: click1

Esta función se corre cuando se selecciona el primer icono. Lo que hace es guardar en el archivo icono.txt el icono que selecciona el usuario y cerrarlo.

```
85  def click1(event):
86      archivo_icono=open("icon.txt","w")
87      archivo_icono.write("icono1.ppm")
88      archivo_icono.close()
```

Función 8: click2

Esta función se corre cuando se selecciona el segundo icono. Lo que hace es guardar en el archivo icono.txt el icono que selecciona el usuario y cerrarlo.

```
92  def click2(event):
93      archivo_icono=open("icon.txt","w")
94      archivo_icono.write("icono2.ppm")
95      archivo_icono.close()
```

Función 9: click3

Esta función se corre cuando se selecciona el tercer icono. Lo que hace es guardar en el archivo icono.txt el icono que selecciona el usuario y cerrarlo.

```
 99  def click3(event):
100      archivo_icono=open("icon.txt","w")
101      archivo_icono.write("icono3.ppm")
102      archivo_icono.close()
```

Función 10: click4

Esta función se corre cuando se selecciona el cuarto icono. Lo que hace es guardar en el archivo icono.txt el icono que selecciona el usuario y cerrarlo.

```
106     def click4(event):
107         archivo_icono=open("icon.txt","w")
108         archivo_icono.write("icono4.ppm")
109         archivo_icono.close()
```

Función 11: click5

Esta función se corre cuando se selecciona el quinto icono. Lo que hace es guardar en el archivo icono.txt el icono que selecciona el usuario y cerrarlo.

```
113     def click5(event):
114         archivo_icono=open("icon.txt","w")
115         archivo_icono.write("icono5.ppm")
116         archivo_icono.close()
```

Función 12: mostrar_error

Esta función hace un widget de tipo messagebox warning, el cual muestra un anuncio con una imagen con un signo de admiración. Esta función toma como parámetro la descripción que se le da al error y pone por default el título “Error”. La función se corre cuando hay un error en el llenado de algún parámetro y permite poner descripciones personalizadas al tipo de error encontrado.

```
128     def mostrar_error(descripcion_del_error):
129         messagebox.showwarning("Error", str(descripcion_del_error))
```

Función 13: mostrar_exito

Esta función hace un widget de tipo messagebox showinfo, el cual muestra un anuncio con una imagen con una “i” de información. Esta función toma como parámetro la descripción que se le da al mensaje y pone por default el título “Exito”. La función se corre cuando se realiza un cambio de fecha o se agrega una actividad exitosamente y permite poner descripciones personalizadas al tipo de mensaje a mostrar.

```
133     def mostrar_exito(descripcion_exito):
134         messagebox.showinfo("Exito", str(descripcion_exito))
```

Función 14: pasar_mes_de_num_a_letra

Esta función crea una variable llamada nombre_mes en la que guarda el nombre del mes en palabra. Esto lo hace tomando la variable mes_numero, la cual almacena el número de mes en el que nos encontramos o uno distinto pedido por el usuario y escribe en la nueva variable (nombre_mes) el nombre del mes con letra. Esta función se corre al inicio del programa y cuando se hace un cambio de fecha.

```
139 def pasar_mes_de_num_a_letra():
140     global nombre_mes
141     if mes_numero == 1:
142         nombre_mes = "Enero"
143     elif mes_numero == 2:
144         nombre_mes = "Febrero"
145     elif mes_numero == 3:
146         nombre_mes = "Marzo"
147     elif mes_numero == 4:
148         nombre_mes = "Abril"
149     elif mes_numero == 5:
150         nombre_mes = "Mayo"
151     elif mes_numero == 6:
152         nombre_mes = "Junio"
153     elif mes_numero == 7:
154         nombre_mes = "Julio"
155     elif mes_numero == 8:
156         nombre_mes = "Agosto"
157     elif mes_numero == 9:
158         nombre_mes = "Septiembre"
159     elif mes_numero == 10:
160         nombre_mes = "Octubre"
161     elif mes_numero == 11:
162         nombre_mes = "Noviembre"
163     elif mes_numero == 12:
164         nombre_mes = "Diciembre"
```

Función 15: borrar_linea_archivo_hoy

Esta función borra la última fila de actividades de la tabla del día (tabla de hoy). Para ello, abre el archivo Actividades_de_hoy.txt, mueve el puntero al inicio, y guarda todo su contenido en la variable contenido. Luego crea una variable con los elementos de esa lista menos uno, ya que ese valor será el elemento a eliminar, la variable se llama a_elim. Luego elimina el elemento en el índice de la que tiene el valor de la variable a_elim de la lista del contenido. Luego borra todo lo que hay en el archivo Actividades_de_hoy.txt y lo reescribe con el nuevo contenido (sin el elemento que se eliminó) y se le resta uno a la variable a_elim (en caso de que se quiera eliminar otro valor). Luego muestra las actividades en el archivo ya actualizado y tapa las actividades eliminadas con un label del color del fondo con puros espacios, esto a través de un ciclo for.

```
168  def borrar_linea_archivo_hoy(archivo):
169      global cont1, band, a_elim
170
171      band=True
172      with open(archivo, "a+") as arch:
173          arch.seek(0)
174          contenido = arch.readlines()
175          if band == True:
176              a_elim = len(contenido)-1
177              band = False
178          contenido.pop(a_elim)
179
180      with open(archivo, "w") as arch2:
181          arch2.writelines("")
182          arch2.writelines(contenido)
183      a_elim -=1
184
185      mostrar_acts()
186
187      for i in range(a_elim+1,cont1+1):
188          Label(f_tablaHoy, text="                                    ", font=10, bg=color1).grid(row=i, column=0)
189          Label(f_tablaHoy, text="                                    ", font=10, bg=color1).grid(row=i, column=1)
190          Label(f_tablaHoy, text="                                    ", font=10, bg=color1).grid(row=i, column=2)
191          Label(f_tablaHoy, text="                                    ", font=10, bg=color1).grid(row=i, column=3)
192          Label(f_tablaHoy, text="                                    ", font=10, bg=color1, height=2).grid(row=i, column=4)
```

Función 16: borrar_linea_archivo_diario

Esta función borra la última fila de actividades de la tabla diaria. Para ello, abre el archivo ActividadesDiarias.txt, mueve el puntero al inicio, y guarda todo su contenido en la variable contenido. Luego crea una variable con los elementos de esa lista menos uno, ya que ese valor será el elemento a eliminar, la variable se llama a_elim. Luego elimina el elemento en el índice de la que tiene el valor de la variable a_elim de la lista del contenido. Luego borra todo lo que hay en el archivo

ActividadesDiarias.txt y lo reescribe con el nuevo contenido (sin el elemento que se eliminó) y se le resta uno a la variable a_elim (en caso de que se quiera eliminar otro valor). Luego muestra las actividades en el archivo ya actualizado y tapa las actividades eliminadas con un label del color del fondo con puros espacios, esto a través de un ciclo for.

```
195
196 def borrar_linea_archivo_diario(archivo):
197     global cont1, band, a_elim
198
199     band=True
200     with open(archivo, "a+") as arch:
201         arch.seek(0)
202         contenido = arch.readlines()
203         if band == True:
204             a_elim = len(contenido)-1
205             band = False
206         contenido.pop(a_elim)
207
208     with open(archivo, "w") as arch2:
209         arch2.writelines("")
210         arch2.writelines(contenido)
211     a_elim -=1
212
213     mostrar_acts()
214
215     for i in range(a_elim+2,cont2+1):
216         Label(f_tablaDiaria, text="                                    ", font=10, bg=color1).grid(row=i, column=0)
217         Label(f_tablaDiaria, text="                                    ", font=10, bg=color1).grid(row=i, column=1)
218         Label(f_tablaDiaria, text="                                    ", font=10, bg=color1).grid(row=i, column=2)
219         Label(f_tablaDiaria, text="                                    ", font=10, bg=color1).grid(row=i, column=3)
220         Label(f_tablaDiaria, text="                                    ", font=10, bg=color1, height=2).grid(row=i, column=4)
221
```

Función 17: mostrar_frame

Esta función simplemente toma como parámetro un frame y lo muestra, colocándolo por encima del resto. Esta función se utiliza para mostrar los frames principales del menú (Agenda, Calendario, Configuración, Personalizar) y se corre al presionar el botón correspondiente a cada frame.

```
224 # Funcion para mostrar un frame del menú
225 def mostrar_frame(frame):
226     frame.tkraise()
```

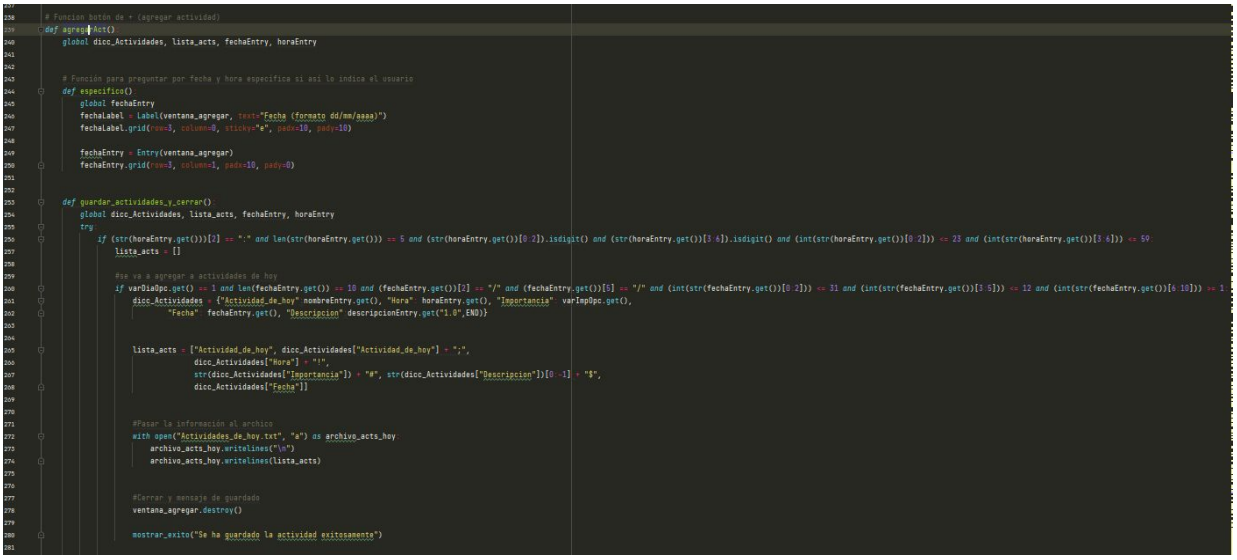
función 18: hacer_calendario

Esta función muestra el calendario del mes presente y permite seleccionar días y cambiar de mes. Esto lo hace por medio de la librería tkcalendar y el objeto calendar. Esta función se corre al inicio, en el frame del calendario.

```
# Función para mostrar el calendario
def hacer_calendario(ano, mes):
    # año y mes del calendario
    calendario = Calendar(frame_opciones_calendario, selectmode="day", year=ano_numero, month=mes_numero, day=dia_numero, width= width_screen-100)
    calendario.pack(pady=32)
```

Función 19: agregarAct

Esta función crea una nueva ventana llamaga “Agregar una actividad” la cual pide el nombre, la hora, si es específico o si es diario, la importancia, y la descripción de la actividad, también crea un botón para guardar las actividades. Esta función se corre cuando se presiona el botón “+”. Esta función tiene dos funciones dentro, una es para crear el parámetro que pide la hora (si es para un día específico) y el otro es para guardar las actividades.



```
# Funcion botón de + (agregar actividad)
def agregarAct():
    global dicc_Actividades, lista_acts, fechaEntry, horaEntry


    # Función para preguntar por fecha y hora especifica si así lo indica el usuario
    def especifico():
        global fechaEntry
        fechaLabel = Label(ventana_agregar, text="Fecha (formato dd/mm/aaaa)")
        fechaLabel.grid(row=3, column=0, sticky="e", padx=10, pady=10)

        fechaEntry = Entry(ventana_agregar)
        fechaEntry.grid(row=3, column=1, padx=10, pady=0)


    def guardar_actividades_y_cerrar():
        global dicc_Actividades, lista_acts, fechaEntry, horaEntry
        try:
            if (str(horaEntry.get()))[2] == ":" and len(str(horaEntry.get())) == 5 and (str(horaEntry.get())[0:2]).isdigit() and (str(horaEntry.get())[3:6]).isdigit() and (int(str(horaEntry.get())[0:2])) <= 23 and (int(str(horaEntry.get())[3:6])) <= 59:
                lista_acts = []

                #se va a agregar a actividades de hoy
                if varDiaOpc.get() == 1 and len(fechaEntry.get()) == 10 and (fechaEntry.get())[2] == "/" and (fechaEntry.get())[5] == "/" and (int(str(fechaEntry.get())[0:2])) <= 31 and (int(str(fechaEntry.get())[3:5])) <= 12 and (int(str(fechaEntry.get())[6:10])) >= 1:
                    dicc_Actividades = {"Actividad_de_hoy" nombreEntry.get(), "Hora"  horaEntry.get(), "Importancia"  varImpOpc.get(),
                            "Fecha"  fechaEntry.get(), "Descripcion" descripcionEntry.get("1.0",END)}


                    lista_acts = ["Actividad_de_hoy", dicc_Actividades["Actividad_de_hoy"] + ";",
                                dicc_Actividades["Hora"] + "!",
                                str(dicc_Actividades["Importancia"]) + "#", str(dicc_Actividades["Descripcion"])[0:-1] + "$",
                                dicc_Actividades["Fecha"]]


                    #Pasar la información al archico
                    with open("Actividades_de_hoy.txt", "a") as archivo_acts_hoy:
                        archivo_acts_hoy.writelines("\n")
                        archivo_acts_hoy.writelines(lista_acts)


                    #Cerrar y mensaje de guardado
                    ventana_agregar.destroy()

                    mostrar_exito("Se ha guardado la actividad exitosamente")
```

```
            #se va a agregar a actividades diarias
            elif varDiaOpc.get() == 2:
                dicc_Actividades = {"Actividad_diaria" nombreEntry.get(), "Hora"  horaEntry.get(), "Importancia"  varImpOpc.get(),
                        "Descripcion" descripcionEntry.get("1.0",END)}


                lista_acts = ["Actividad_diaria",dicc_Actividades["Actividad_diaria"] + ";", dicc_Actividades["Hora"] + "!",
                              str(dicc_Actividades["Importancia"]) + "#",dicc_Actividades["Descripcion"], 1]

                #Pasar la información al archivo
                with open("ActividadesDiarias.txt", "a") as archivo_acts_diarias:
                    archivo_acts_diarias.writelines(lista_acts)

                #Cerrar y mensaje de guardado
                ventana_agregar.destroy()
                mostrar_exito("Se ha guardado la actividad exitosamente")


        #Errores en la fecha, mostrar mensaje apropiado y cerrar
            elif len(fechaEntry.get()) != 10:
                mostrar_error("Error: la fecha no esta escrita en el formato correcto.")
                ventana_agregar.destroy()

            elif (fechaEntry.get())[2] != "/" or (fechaEntry.get())[5] != "/":
                mostrar_error("Error: la fecha debe tener un '/' entre el dia, el mes y el año.")
                ventana_agregar.destroy()

            elif (int(str(fechaEntry.get())[0:2])) >= 31 or (int(str(fechaEntry.get())[3:5])) >= 12 or (int(str(fechaEntry.get())[6:10])) <= 1:
                mostrar_error("Error: los valores de la fecha son imposibles.")
                ventana_agregar.destroy()

            else:
                mostrar_error("Error: la sintaxis o valor de la fecha es incorrecto.")
                ventana_agregar.destroy()


        #Errores en la hora, mostrar mensaje apropiado y cerrar
        elif str(horaEntry.get())[2] != ":":
                mostrar_error("Error: la hora debe tener un ':' entre las horas y los minutos.")
                ventana_agregar.destroy()

        elif len(str(horaEntry.get())) != 5:
                mostrar_error("Error: la hora no esta escrita en el formato correcto.")
                ventana_agregar.destroy()
```

```
            elif (str(horaEntry.get())[0:2]).isalpha() or (str(horaEntry.get())[3:6]).isalpha():
                    mostrar_error("Error: los valores de la hora deben ser numericos.")
                    ventana_agregar.destroy()

            elif (int(str(horaEntry.get())[0:2])) >= 23 or (int(str(horaEntry.get())[3:6])) >= 59:
                    mostrar_error("Error: los valores de la hora son imposibles.")
                    ventana_agregar.destroy()

            else:
                mostrar_error("Error: la sintaxis o valor de la hora es incorrecto.")
                ventana_agregar.destroy()

        #Errores mayores, mostrar mensaje apropiado y cerrar
        except:
            if varDiaOpc.get() == 1:
                if (str(fechaEntry.get())[0:2]).isalpha() or (str(fechaEntry.get())[3:5]).isalpha() or (str(fechaEntry.get())[6:10]).isalpha():
                    mostrar_error("Error: los valores de la fecha deben ser numéricos.")
                    ventana_agregar.destroy()

                else:
                    mostrar_error("Error: De deben llenar los campos requeridos.")
                    ventana_agregar.destroy()
            else:
                mostrar_exito("Se ha guardado la actividad exitosamente")
                ventana_agregar.destroy()

        mostrar_acts()


    dicc_Actividades = {}


    # Variables para opciones de importancia y si es dia es especifico o diario
    varDiaOpc = IntVar()
    varImpOpc = IntVar()

    # Información del root de la nueva pestaña de agragar actividad (ventana_agregar)
    ventana_agregar = Toplevel(width=1200, height=1000)
    ventana_agregar.title("Agregar una actividad")

    # Pide nombre
    nombreLabel = Label(ventana_agregar, text="Nombre de la actividad")
    nombreLabel.grid(row=0, column=0, sticky="e", padx=10, pady=10)

    # Espacio para escribir nombre
    nombreEntry = Entry(ventana_agregar)
    nombreEntry.grid(row=0, column=1, padx=10, pady=0)
```

```
    # Pide hora
    horaLabel = Label(ventana_agregar, text="Hora (formato hh:mm)")
    horaLabel.grid(row=4, column=0, sticky="e", padx=10, pady=10)

    horaEntry = Entry(ventana_agregar)
    horaEntry.grid(row=4, column=1, padx=10, pady=0)

    # Pregunta si la actividad se va a agregar a un dia especifico o va a ser diaria
    varDiaOpc = IntVar()
    varDiaOpc.set(2)

    # Opción de dia especifico, (corre la función de especifico)
    diaOpc1 = Radiobutton(ventana_agregar, text="Dia especifico", variable=varDiaOpc, value=1, command=especifico)
    diaOpc1.grid(row=2, column=0, padx=10, pady=10)

    # Opción de diario
    diaOpc2 = Radiobutton(ventana_agregar, text="Diario", variable=varDiaOpc, value=2)
    diaOpc2.grid(row=2, column=1, padx=10, pady=10)

    # Pide importancia de la actividad
    varImpOpc = StringVar()
    varImpOpc.set(1)
    importanciaLabel = Label(ventana_agregar, text="Importancia")
    importanciaLabel.grid(row=5, column=0, sticky="e")

    # Opciones de los diferentes tipos de importancia
    impOpc1 = Radiobutton(ventana_agregar, text="Verde", variable=varImpOpc, value="Verde")
    impOpc1.grid(row=5, column=1, padx=10, sticky="w")

    impOpc2 = Radiobutton(ventana_agregar, text="Amarillo", variable=varImpOpc, value="Amarillo")
    impOpc2.grid(row=6, column=1, padx=10, sticky="w")

    impOpc3 = Radiobutton(ventana_agregar, text="Rojo", variable=varImpOpc, value="Rojo")
    impOpc3.grid(row=7, column=1, padx=10, sticky="w")

    # Pide la descripcion
    descripcionLabel = Label(ventana_agregar, text="Descripción: ")
    descripcionLabel.grid(row=8, column=0, sticky="e", padx=10, pady=10)

    descripcionEntry = Text(ventana_agregar, width=16, height=5)
    descripcionEntry.grid(row=8, column=1, padx=0, pady=10)

    # Le pone un scrollbar al cuadro de texto de descripción
    scrollVert_descripcion = Scrollbar(ventana_agregar, command=descripcionEntry.yview)
    scrollVert_descripcion.grid(row=8, column=2, sticky="nsew")
    descripcionEntry.config(yscrollcommand=scrollVert_descripcion.set)
```

```
    # Botón para guardar y cerrar ventana ventana_agregar
    botonEnviar = Button(ventana_agregar, text="Guardar", command=guardar_actividades_y_cerrar)
    botonEnviar.grid(row=9, column=0, sticky="e", pady="10")
```

Función 20: específico

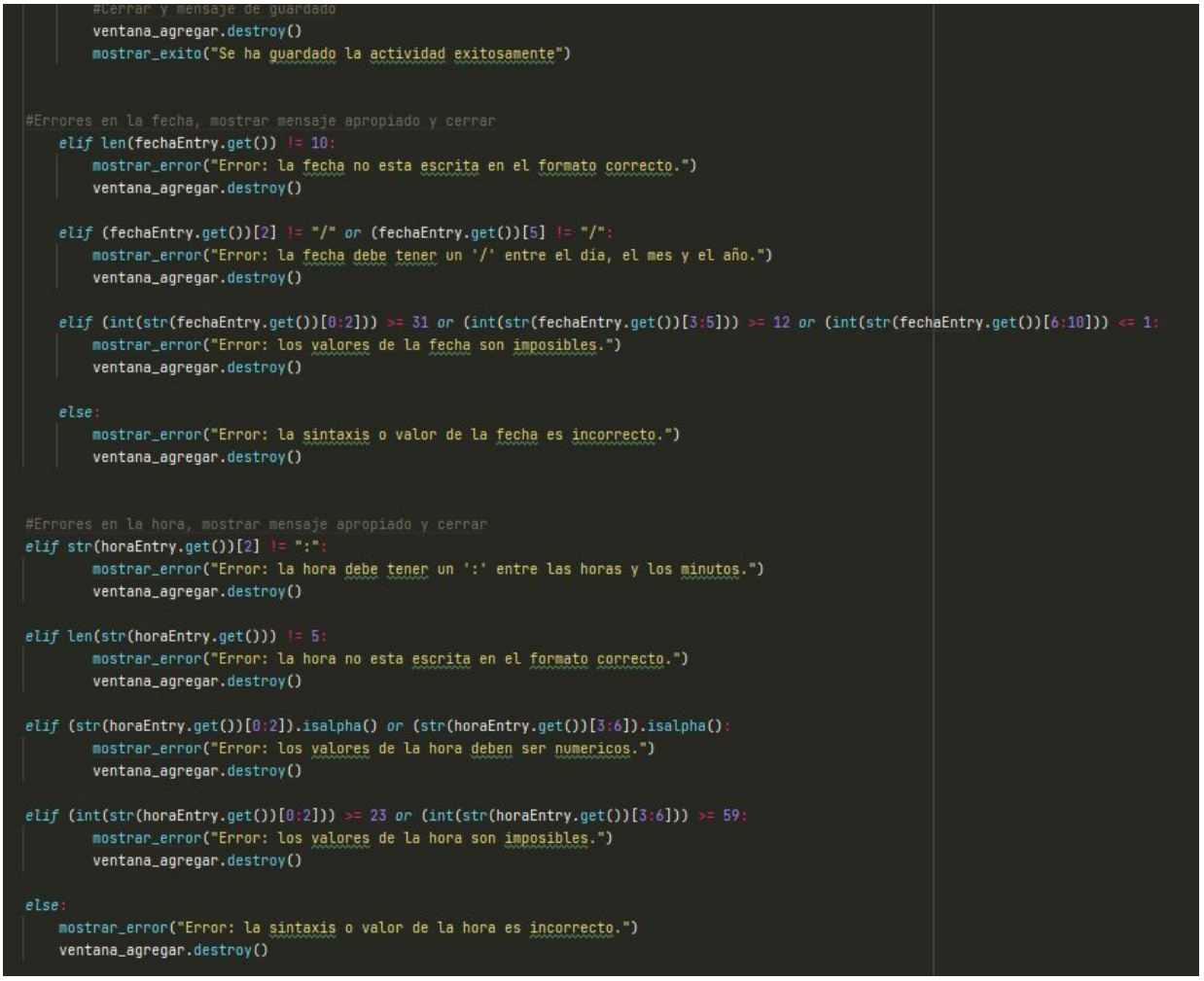
Esta función está dentro de la función agregarAct y se corre cuando se selecciona la opción de día específico. La función crea un Label y un Entry para colocar la fecha del día específico.

```
242
243         # Función para preguntar por fecha y hora específica si así lo indica el usuario
244         def especifico():
245             global fechaEntry
246             fechaLabel = Label(ventana_agregar, text="Fecha (formato dd/mm/aaaa)")
247             fechaLabel.grid(row=3, column=0, sticky="e", padx=10, pady=10)
248
249             fechaEntry = Entry(ventana_agregar)
250             fechaEntry.grid(row=3, column=1, padx=10, pady=0)
```

Función 21: guardar_actividades_y_cerrar

Esta función está dentro de la función agregarAct y se corre al presionar el botón de guardado de actividad. Lo que hace la función es revisar cada una de las entradas del usuario en busca de errores. Si encuentra un error, muestra un anuncio en pantalla con la descripción específica del error encontrado, si no se encuentra un error, guarda todas las entradas del usuario en una lista, separados por distintos símbolos y más tarde en el archivo de texto correspondiente al tipo de actividad (diaria o de hoy), luego muestra un anuncio de éxito y cierra la ventana ventana_agregar y muestra las actividades con la función mostrar_acts..

```
def guardar_actividades_y_cerrar():
    global dicc_Actividades, lista_acts, fechaEntry, horaEntry
    try:
        if (str(horaEntry.get())[2] == ":" and len(str(horaEntry.get())) == 5 and (str(horaEntry.get())[0:2]).isdigit() and (str(horaEntry.get())[3:6]).isdigit() and (int(str(horaEntry.get())[0:2])) <= 23 and (int(str(horaEntry.get())[3:6])) <= 59:
            lista_acts = []

            #se va a agregar a actividades de hoy
            if varDiaOpc.get() == 1 and len(fechaEntry.get()) == 10 and (fechaEntry.get())[2] == "/" and (fechaEntry.get())[5] == "/" and (int(str(fechaEntry.get())[0:2])) <= 31 and (int(str(fechaEntry.get())[3:5])) <= 12 and (int(str(fechaEntry.get())[6:10])) >= 1:
                dicc_Actividades = {"Actividad_de_hoy": nombreEntry.get(), "Hora": horaEntry.get(), "Importancia": varImpOpc.get(),
                        "Fecha": fechaEntry.get(), "Descripcion": descripcionEntry.get("1.0",END)}


                lista_acts = ["Actividad_de_hoy", dicc_Actividades["Actividad_de_hoy"] + ";",
                              dicc_Actividades["Hora"] + "!",
                              str(dicc_Actividades["Importancia"]) + "#", str(dicc_Actividades["Descripcion"])[0:-1] + "$",
                              dicc_Actividades["Fecha"]]


                #Pasar la información al archico
                with open("Actividades_de_hoy.txt", "a") as archivo_acts_hoy:
                    archivo_acts_hoy.writelines("\n")
                    archivo_acts_hoy.writelines(lista_acts)


                #Cerrar y mensaje de guardado
                ventana_agregar.destroy()

                mostrar_exito("Se ha guardado la actividad exitosamente")


            #se va a agregar a actividades diarias
            elif varDiaOpc.get() == 2:
                dicc_Actividades = {"Actividad_diaria": nombreEntry.get(), "Hora": horaEntry.get(), "Importancia": varImpOpc.get(),
                        "Descripcion": descripcionEntry.get("1.0",END)}


                lista_acts = ["Actividad_diaria",dicc_Actividades["Actividad_diaria"] + ";", dicc_Actividades["Hora"] + "!",
                              str(dicc_Actividades["Importancia"]) + "#",dicc_Actividades["Descripcion"], 1]

                #Pasar la información al archivo
                with open("ActividadesDiarias.txt", "a") as archivo_acts_diarias:
                    archivo_acts_diarias.writelines(lista_acts)
```



```
        #Cerrar y mensaje de guardado
        ventana_agregar.destroy()
        mostrar_exito("Se ha guardado la actividad exitosamente")


#Errores en la fecha, mostrar mensaje apropiado y cerrar
    elif len(fechaEntry.get()) != 10:
        mostrar_error("Error: la fecha no esta escrita en el formato correcto.")
        ventana_agregar.destroy()

    elif (fechaEntry.get())[2] != "/" or (fechaEntry.get())[5] != "/":
        mostrar_error("Error: la fecha debe tener un '/' entre el dia, el mes y el año.")
        ventana_agregar.destroy()

    elif (int(str(fechaEntry.get())[0:2])) >= 31 or (int(str(fechaEntry.get())[3:5])) >= 12 or (int(str(fechaEntry.get())[6:10])) <= 1:
        mostrar_error("Error: los valores de la fecha son imposibles.")
        ventana_agregar.destroy()

    else:
        mostrar_error("Error: la sintaxis o valor de la fecha es incorrecto.")
        ventana_agregar.destroy()


#Errores en la hora, mostrar mensaje apropiado y cerrar
elif str(horaEntry.get())[2] != ":":
        mostrar_error("Error: la hora debe tener un ':' entre las horas y los minutos.")
        ventana_agregar.destroy()

elif len(str(horaEntry.get())) != 5:
        mostrar_error("Error: la hora no esta escrita en el formato correcto.")
        ventana_agregar.destroy()

elif (str(horaEntry.get())[0:2]).isalpha() or (str(horaEntry.get())[3:6]).isalpha():
        mostrar_error("Error: los valores de la hora deben ser numericos.")
        ventana_agregar.destroy()

elif (int(str(horaEntry.get())[0:2])) >= 23 or (int(str(horaEntry.get())[3:6])) >= 59:
        mostrar_error("Error: los valores de la hora son imposibles.")
        ventana_agregar.destroy()

else:
    mostrar_error("Error: la sintaxis o valor de la hora es incorrecto.")
    ventana_agregar.destroy()
```

```
    #Errores mayores, mostrar mensaje apropiado y cerrar
    except:
        if varDiaOpc.get() == 1:
            if (str(fechaEntry.get())[0:2]).isalpha() or (str(fechaEntry.get())[3:5]).isalpha() or (str(fechaEntry.get())[6:10]).isalpha():
                mostrar_error("Error: los valores de la fecha deben ser numéricos.")
                ventana_agregar.destroy()

            else:
                mostrar_error("Error: De deben llenar los campos requeridos.")
                ventana_agregar.destroy()
        else:
            mostrar_exito("Se ha guardado la actividad exitosamente")
            ventana_agregar.destroy()

    mostrar_acts()
```

Función 22: hacer_frame_tabla_hoy

Esta función crea un frame llamado f_tablaHoy dentro del frame de la agenda. En este frame se crean cinco columnas. La comumna Actividad de Hoy, Hora, Importancia, Descripción y Realizada. Esta función se corre al inicio del programa.

```
def hacer_frame_tabla_hoy():
    global f_tablaHoy
    f_tablaHoy = Frame(f_agenda, bg=color1)
    f_tablaHoy.pack(side="top", fill="x")
    f_tablaHoy.config(heigh=height_screen / 2 - 80)

    for column in range(13):
        Grid.columnconfigure(f_tablaHoy, column, weight=1)

    columna_Act_hoy = Label(f_tablaHoy, bg=color3, text="      Actividad de Hoy      ", font=10).grid(row=0, column=0)
    columna_Hora_hoy = Label(f_tablaHoy, bg=color3, text="       Hora       ", font=10).grid(row=0, column=1)
    columna_Importancia_hoy = Label(f_tablaHoy, bg=color3, text="       Importancia       ", font=10).grid(row=0,
                                                                                                            column=2)
    columna_Descripcion_hoy = Label(f_tablaHoy, bg=color3, text="       Descripción       ", font=10).grid(row=0,
                                                                                                            column=3)
    columna_Realizada_hoy = Label(f_tablaHoy, bg=color3, text="       Realizada       ", font=10).grid(row=0, column=4)
```

Función 23: cambiar_fecha

Esta función crea una nueva ventana llamada ventana_cambiar, la cual pide una nueva fecha a cambiar y tiene un botón para guardar. Esta función se corre cuando se presiona el botón de cambiar fecha y tiende dentro la función guardar_fecha.

```
# Función para (cambiar la fecha)
def cambiar_fecha():
    global dia_numero, ano_numero, mes_numero, nombre_mes, f_tablaHoy


    #Cuando se le da al boton de guardado
    def guardar_fecha():
        global dia_numero, ano_numero, nombre_mes, mes_numero

        dia_ingresado = fechaEntry1.get()

        try:

            #Checar que todos los datos esten bien
            if len(dia_ingresado) == 10 and dia_ingresado[2] == "/" and dia_ingresado[5] == "/" and (int(dia_ingresado[0:2])) <= 31 and (int(dia_ingresado[3:5])) <= 12 and
                #cambiar dia
                dia_numero = 0
                str_dia_numero = ""
                for i in str(dia_ingresado)[0:2]:
                    if len(str_dia_numero) < 2:
                        str_dia_numero += i
                        dia_numero = int(str_dia_numero)

                #cambiar mes
                mes_numero = 0
                str_mes_numero = ""
                for i in str(dia_ingresado)[3:5]:
                    if len(str_mes_numero) < 2:
                        str_mes_numero += i
                        mes_numero = int(str_mes_numero)

                #cambiar año
                ano_numero = 0
                str_ano_numero = ""
                for i in str(dia_ingresado)[6:10]:
                    if len(str_ano_numero) < 4:
                        str_ano_numero += i
                        ano_numero = int(str_ano_numero)
```

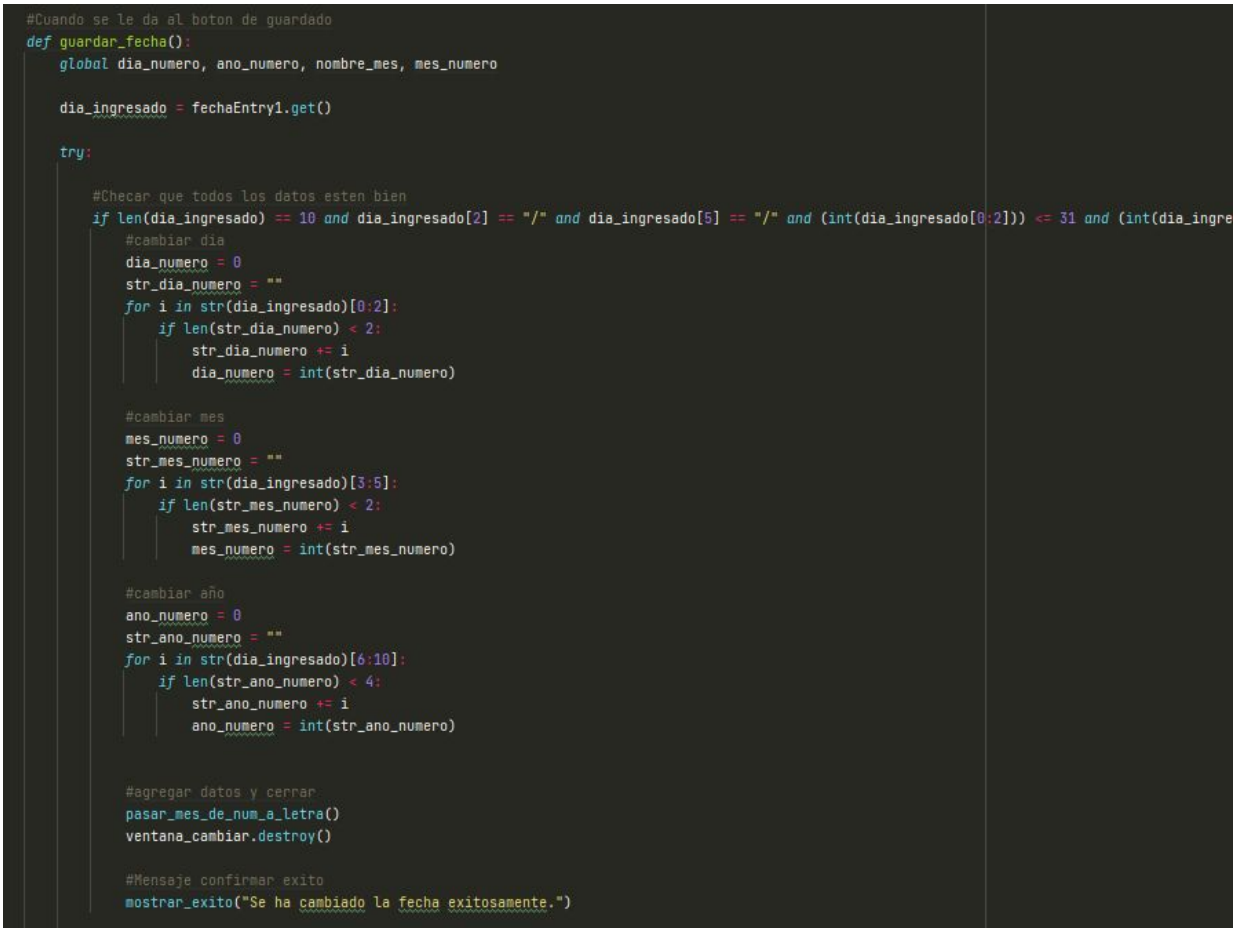
```
        #agregar datos y cerrar
        pasar_mes_de_num_a_letra()
        ventana_cambiar.destroy()

        #Mensaje confirmar exito
        mostrar_exito("Se ha cambiado la fecha exitosamente.")

    #Errores en la fecha, mostrar mensaje apropiado y cerrar
    elif not dia_ingresado:
        mostrar_error("Error: se debe ingresar una fecha.")
        ventana_cambiar.destroy()

    elif len(dia_ingresado) != 10:
        mostrar_error("Error: el formato de la fecha es incorrecto.")
        ventana_cambiar.destroy()

    elif dia_ingresado[2] != "/" or dia_ingresado[5] != "/":
        mostrar_error("Error: la fecha debe tener un '/' entre el dia, el mes y el año.")
        ventana_cambiar.destroy()

    elif (int(dia_ingresado[0:2])) >= 31 or (int(dia_ingresado[3:5])) >= 12 or (int(dia_ingresado[6:10])) <= 1:
        mostrar_error("Error: los valores de la fecha son imposibles.")
        ventana_cambiar.destroy()

    elif (str(dia_ingresado[6:10])).isalpha() or (str(dia_ingresado[0:2])).isalpha() or (str(dia_ingresado[3:5])).isalpha() or (str(dia_ingresado[6:10])).isalpha():
        mostrar_error("Error: los valores de la fecha deben ser numéricos.")
        ventana_cambiar.destroy()

    else:
        mostrar_error("Error: la sintaxis o valor de la fecha es incorrecto.")
        ventana_cambiar.destroy()
```

```
    #Errores mayores, mostrar mensaje apropiado y cerrar
    except:
        if not dia_ingresado:
            mostrar_error("Error: se debe ingresar una fecha.")
            ventana_cambiar.destroy()

        elif (str(dia_ingresado[6:10])).isalpha() or (str(dia_ingresado[0:2])).isalpha() or (str(dia_ingresado[3:5])).isalpha() or (str(dia_ingresado[6:10])).isalpha():
            mostrar_error("Error: los valores de la fecha deben ser numéricos.")
            ventana_cambiar.destroy()

        else:
            mostrar_error("Error: la sintaxis o valor de la fecha es incorrecto.")
            ventana_cambiar.destroy()


    #Mostrar nueva fecha
    textoFecha.set(str(dia_numero) + " de " + nombre_mes + " del " + str(ano_numero))


    #f_tablaHoy.destroy()
    for i in range(1, cont1 + 1):
        Label(f_tablaHoy, text="                          ", font=10, bg=color1).grid(row=i, column=0)
        Label(f_tablaHoy, text="                          ", font=10, bg=color1).grid(row=i, column=1)
        Label(f_tablaHoy, text="                          ", font=10, bg=color1).grid(row=i, column=2)
        Label(f_tablaHoy, text="                          ", font=10, bg=color1).grid(row=i, column=3)
        Label(f_tablaHoy, text="                          ", font=10, bg=color1, height=2).grid(row=i, column=4)
    #hacer_frame_tabla_hoy()
    mostrar_acts()
```

```
# Crea una nueva ventana (camb)
ventana_cambiar = Toplevel(width=1200, height=1000)
ventana_cambiar.title("Cambiar de día")

# Pide nueva fecha
fechaLabel1 = Label(ventana_cambiar, text="Fecha (formato dd/mm/aaaa)")
fechaLabel1.grid(row=0, column=0, sticky="e", padx=10, pady=10)
fechaEntry1 = Entry(ventana_cambiar)
fechaEntry1.grid(row=0, column=1, padx=10, pady=0)

# Guardar nueva fecha y cerrar ventana
Button(ventana_cambiar, text="Guardar", command=guardar_fecha).grid(row=1, column=0, padx=10, pady=0)

#mostrar_acts() (en proceso, si no se comenta, se repiten)
frame_opciones_calendario.pack()
```

Función 24: guardar_fecha

Esta función está dentro de la función cambiar fecha y corre cuando se presiona el botón de guardar fecha. La función toma el valor que está en el entry de la nueva fecha y revisa el formato de la fecha seleccionada por el usuario. Si hay algún error en la fecha, muestra un mensaje de error personalizado para el error que se haya cometido. Si no se cometen errores, la función guarda los valores de la fecha dados por el usuario en las variables de dia_numero, mes_numero y ano_numero, los convierte a letra con la función pasar_mes_de_num_a_letra, cierra la ventana, muestra un mensaje de éxito en el cambio de fecha, muestra la nueva fecha en su respectivo frame, tapa las actividades que no tienen la fecha del día con un labes con espacios y muestra las actividades del día correspondiente.



```
#Cuando se le da al boton de guardado
def guardar_fecha():
    global dia_numero, ano_numero, nombre_mes, mes_numero

    dia_ingresado = fechaEntry1.get()

    try:

        #Checar que todos los datos esten bien
        if len(dia_ingresado) == 10 and dia_ingresado[2] == "/" and dia_ingresado[5] == "/" and (int(dia_ingresado[0:2])) <= 31 and (int(dia_ingre
            #cambiar dia
            dia_numero = 0
            str_dia_numero = ""
            for i in str(dia_ingresado)[0:2]:
                if len(str_dia_numero) < 2:
                    str_dia_numero += i
                    dia_numero = int(str_dia_numero)

            #cambiar mes
            mes_numero = 0
            str_mes_numero = ""
            for i in str(dia_ingresado)[3:5]:
                if len(str_mes_numero) < 2:
                    str_mes_numero += i
                    mes_numero = int(str_mes_numero)

            #cambiar año
            ano_numero = 0
            str_ano_numero = ""
            for i in str(dia_ingresado)[6:10]:
                if len(str_ano_numero) < 4:
                    str_ano_numero += i
                    ano_numero = int(str_ano_numero)


            #agregar datos y cerrar
            pasar_mes_de_num_a_letra()
            ventana_cambiar.destroy()

            #Mensaje confirmar exito
            mostrar_exito("Se ha cambiado la fecha exitosamente.")
```

```
#Errores en la fecha, mostrar mensaje apropiado y cerrar
elif not dia_ingresado:
    mostrar_error("Error: se debe ingresar una fecha.")
    ventana_cambiar.destroy()

elif len(dia_ingresado) != 10:
    mostrar_error("Error: el formato de la fecha es incorrecto.")
    ventana_cambiar.destroy()

elif dia_ingresado[2] != "/" or dia_ingresado[5] != "/":
    mostrar_error("Error: la fecha debe tener un '/' entre el dia, el mes y el año.")
    ventana_cambiar.destroy()

elif (int(dia_ingresado[0:2])) >= 31 or (int(dia_ingresado[3:5])) >= 12 or (int(dia_ingresado[6:10])) <= 1:
    mostrar_error("Error: los valores de la fecha son imposibles.")
    ventana_cambiar.destroy()

elif (str(dia_ingresado[6:10])).isalpha() or (str(dia_ingresado[0:2])).isalpha() or (str(dia_ingresado[3:5])).isalpha() or (str(dia_ingresado[6:10])).isalpha():
    mostrar_error("Error: los valores de la fecha deben ser numéricos.")
    ventana_cambiar.destroy()

else:
    mostrar_error("Error: la sintaxis o valor de la fecha es incorrecto.")
    ventana_cambiar.destroy()
```

```
#Errores mayores, mostrar mensaje apropiado y cerrar
except:
    if not dia_ingresado:
        mostrar_error("Error: se debe ingresar una fecha.")
        ventana_cambiar.destroy()

    elif (str(dia_ingresado[6:10])).isalpha() or (str(dia_ingresado[0:2])).isalpha() or (str(dia_ingresado[3:5])).isalpha() or (str(dia_ingresado[6:10])).isalpha():
        mostrar_error("Error: los valores de la fecha deben ser numéricos.")
        ventana_cambiar.destroy()

    else:
        mostrar_error("Error: la sintaxis o valor de la fecha es incorrecto.")
        ventana_cambiar.destroy()


#Mostrar nueva fecha
textoFecha.set(str(dia_numero) + " de " + nombre_mes + " del " + str(ano_numero))


#f_tablaHoy.destroy()
for i in range(1, cont1 + 1):
    Label(f_tablaHoy, text="                          ", font=10, bg=color1).grid(row=i, column=0)
    Label(f_tablaHoy, text="                          ", font=10, bg=color1).grid(row=i, column=1)
    Label(f_tablaHoy, text="                          ", font=10, bg=color1).grid(row=i, column=2)
    Label(f_tablaHoy, text="                          ", font=10, bg=color1).grid(row=i, column=3)
    Label(f_tablaHoy, text="                          ", font=10, bg=color1, height=2).grid(row=i, column=4)
#hacer_frame_tabla_hoy()
mostrar_acts()
```

Función 25: actualizar_etiqueta

Esta función crea un label y toma como parámetros el frame o la raíz, el texto, la fila y la columna a la que se hace grid. Esta función se utiliza para agregar las actividades a las tablas diarias y de día específico según especifique el usuario.

```
568
569     def actualizar_etiqueta(root, texto, fila, column):
570         Label(root, text=texto, font=10, bg=color3).grid(row=fila, column=column)
571
```

Función 26: mostrar_acts

Esta función lee los archivos tanto de las actividades diarias como de las actividades de hoy en el archivo correspondiente en modo “r” read y busca las partes en la que está el nombre de actividad, la hora, la importancia, la descripción y la fecha a partir de los símbolos que las separan y las coloca en sus respectivas tablas dependiendo del archivo, la fecha, etc. Si los archivos no existen los crea, esto se checa por medio de un try, except. La función se corre al inicio del programa, al borrar una actividad, al agregar una actividad nueva y al cambiar de fecha.

```
#Mostrar las actividades en frame agenda (en proceso)
def mostrar_acts():
    global cont1, cont2, f_tablaHoy
    cont1 = 1
    cont2 = 1

    #Si ya existe el archivo de acts de hoy
    try:
        #Leer archivo de actividades de hoy
        with open("Actividades_de_hoy.txt", "r") as archivo_acts_hoy:
            archivo_acts_hoy.seek(0)
            lista_acts_hoy = archivo_acts_hoy.readlines()


        #Agregar actividades
            for elem in lista_acts_hoy:
                inicio_hora_hoy = str(elem).find(";")
                inicio_importancia_hoy = str(elem).find("!")
                inicio_descripcion_hoy = str(elem).find("#")
                inicio_fecha = str(elem).find("$")

                if dia_numero<10:
                    str_dia_numero = "0" + str(dia_numero)
                else:
                    str_dia_numero = str(dia_numero)

                if mes_numero<10:
                    str_mes_numero = "0" + str(mes_numero)
                else:
                    str_mes_numero = str(mes_numero)

                str_ano_numero = str(ano_numero)
```

```
            if (str(elem)[inicio_fecha+1:-1] == str(str_dia_numero) + "/" + str(str_mes_numero) + "/" + str(str_ano_numero)) or (str(elem)[inicio_fecha+1:]

                if str(elem)[0:16] == "Actividad_de_hoy":
                    #Label(f_tablaHoy, text=str(elem)[16:inicio_hora_hoy], font=10).grid(row=cont1, column=0)
                    actualizar_etiqueta(f_tablaHoy, (str(elem))[16:inicio_hora_hoy], cont1, 0)

                if str(elem)[inicio_hora_hoy] == ";":
                    #Label(f_tablaHoy, text=str(elem)[inicio_hora_hoy + 1:inicio_importancia_hoy],

                    #font=10).grid(row=cont1, column=1)
                    actualizar_etiqueta(f_tablaHoy, str(elem)[inicio_hora_hoy + 1:inicio_importancia_hoy], cont1, 1)


                if str(elem)[inicio_importancia_hoy] == "!":
                    actualizar_etiqueta(f_tablaHoy, str(elem)[inicio_importancia_hoy + 1:inicio_descripcion_hoy], cont1, 2)


                if str(elem)[inicio_descripcion_hoy] == "#":

                    actualizar_etiqueta(f_tablaHoy, str(elem)[inicio_descripcion_hoy + 1:inicio_fecha], cont1, 3)


                Button(f_tablaHoy, text="Realizado", font=10, command=lambda: borrar_linea_archivo_hoy("Actividades_de_hoy.txt"), height=1).grid(row=cont1,

                cont1 += 1


    #Si no existe el archivo hacer uno nuevo
    except:
        with open("Actividades_de_hoy.txt", "a") as archivo_acts_hoy:
            archivo_acts_hoy.write("")
```

```
try:                                                                                                    ▲ 452
    #Leer archivo de acts diarias
    with open("ActividadesDiarias.txt", "r") as archivo_acts_diarias:
        lista_acts_diarias = archivo_acts_diarias.readlines()

    #Agregar actividades

        for elem in lista_acts_diarias:

            inicio_hora_diaria = str(elem).find(";")
            inicio_importancia_diaria = str(elem).find("!")
            inicio_descripcion_diaria = str(elem).find("#")


            if str(elem)[0:16] == "Actividad_diaria":

                actualizar_etiqueta(f_tablaDiaria, str(elem)[16:inicio_hora_diaria], cont2, 0)

            if str(elem)[inicio_hora_diaria] == ";":
                Label(f_tablaDiaria,bg=color3, text=str(elem)[inicio_hora_diaria+1:inicio_importancia_diaria], font=10).grid(row=cont2, column=1)

            if str(elem)[inicio_importancia_diaria] == "!":
                Label(f_tablaDiaria,bg=color3, text=str(elem)[inicio_importancia_diaria+1:inicio_descripcion_diaria], font=10).grid(row=cont2, column=2)

            if str(elem)[inicio_descripcion_diaria] == "#":
                actualizar_etiqueta(f_tablaDiaria, str(elem)[inicio_descripcion_diaria + 1:-1], cont2, 3)
                #Label(f_tablaDiaria,bg=color3, text=str(elem)[inicio_descripcion_diaria+1:], font=10).grid(row=cont2, column=3)


            Button(f_tablaDiaria, text="Realizado", font=10,
                   command=lambda: borrar_linea_archivo_diario("ActividadesDiarias.txt")).grid(row=cont2, column=4)

            cont2 += 1
```

```
#Si no existe el archivo hacer uno nuevo
except:
    with open("ActividadesDiarias.txt", "a") as archivo_acts_diarias:
        archivo_acts_diarias.write("")
```

Función 27: inicio

Esta es la función que se llama al inicio del programa. Primero que nada, ella obtiene la fecha de hoy por medio de la librería datetime y lo guarda en las variables ano_numero, mes_numero y dia_numero, luego llama a la función pasar_mes_de_num_a_letra para convertirlos a letras. Luego crea la raiz de nuestro programa (la ventana principal), le da el título de “Friendly timetable” (el nombre del programa) y le da el tamaño de la pantalla del usuario y no permite cambiar su tamaño. También, la función crea un frame para el menú en la que se ponen los cuatro botones del menú (agenda, calendario, personalizar y configuración) los cuales lllaman a la función mostrar_frame para ponerlas por encima. También crea el frame principal en donde se ponen los frames para cada una de las opciones del menú y se ponen por encima cuando se presiona su botón correspondiente. Para iniciar, muestra el frame de la agenda.

En el frame de la agenda, se agregan dos labels superiores con el título de agenda y la fecha del día en que se están viendo las actividades, el valor predeterminado es el del día actual. También se agregan los botones para cambiar de día y agregar actividad las cuales corren sus funciones correspondientes explicadas anteriormente. Luego se corre la función para hacer el frame de la tabla hoy facer_frame_tabla_hoy y se hace la tabla diaria por medio de un frame y dentro se hacen labels con los títulos de las columnas (Actividad diaria, hora, importacia, descripción y realizada) luego se corre la función de mostrar_acts para mostrar las actividades diarias y del día.

En el frame del calendario se crea un frame para el calendario y se corre la función hacer_calendario para mostrar el calendario. En el frame de personalizar, se guardan los iconos por medio del objeto PhotoImage en variables, un label con la leyenda “Selección de Icono”, se muestran las opciones de íconos y se les crea un scrollbar debajo. Luego, se hace un frame para la selección del tema personalizado y un label que dice selección de tema. Luego se muestran todos los temas (devs2, straid, joy, deadly y west) y un scrollbar debajo.

En el frame de la configuración se hace un frame y en él se hace un label que dice Perfil, luego se muestra el icono seleccionado por el usuario y dos botones, para cambiar de nombre y de cumpleaños y debajo hay dos labels, una que muestra el nombre y otra el cumpleaños.

```python
def inicio():
    global f_agenda, mes_numero, height_screen, frame_opciones_calendario, ano_numero, dia_numero, width_screen, textoFecha, f_tablaDiaria, f_tablaHoy, cont1, cont2
    # obtener datos del día de hoy
    #locale.setlocale(locale.LC_ALL, 'es-ES')
    ano_numero = datetime.date.today().year
    mes_numero = datetime.date.today().month
    dia_numero = datetime.date.today().day

    #Ejecucion de funciones para iniciar
    pasar_mes_de_num_a_letra()


    # Se inicia el main loop (raiz)
    raiz = Tk()


    # Configuración del raiz principal
    raiz.title("Friendly Timetable")

    # w y h son el temaño de la pantalla del usuario
    width_screen, height_screen = raiz.winfo_screenwidth(), raiz.winfo_screenheight()
    raiz.geometry("%dx%d+0+0" % (width_screen, height_screen - 60))

    # No se puede cambiar el tamaño de la ventana
    raiz.resizable(False, False)
```

```python
# Frame para el menú de opciones de página
frameMenu = Frame(raiz,bg=color1)

# Tamaño y posicion del frame
frameMenu.pack(side="left", anchor="w", fill="y", expand="True")
frameMenu.config(width="300", heigh="350")



# FramePrincipal
f_princ = Frame(raiz, height=height_screen - 50, width=width_screen, bg=color1)
f_princ.pack(side="left")


# Frames por modulo que se muestran cuando apretamos los botones de frame menu
f_agenda = Frame(f_princ, height=height_screen - 70, width=width_screen - 200, bg=color1)
f_agenda.grid(row=0, column=0, sticky='news')

f_calendario = Frame(f_princ, height=height_screen - 50, width=width_screen, bg=color1)
f_calendario.grid(row=0, column=0, sticky='news')


f_personalizar = Frame(f_princ, height=height_screen - 50, width=width_screen, bg=color1)
f_personalizar.grid(row=0, column=0, sticky='news')

f_configuracion = Frame(f_princ, height=height_screen - 50, width=width_screen, bg=color1)
f_configuracion.grid(row=0, column=0, sticky='news')
```

```python
# Botones FrameMenu
# Boton que muestra el frame de agenda
boton_agenda = Button(frameMenu,bg=color2, text="Agenda", heigh="8", width="20", font="50",
                      command=lambda: mostrar_frame(f_agenda))
boton_agenda.grid(row="0", column="0", sticky="we")

# Boton que muestra el frame de calendario
boton_Calendario = Button(frameMenu,bg=color2, text="Calendario", heigh="8", width="20", font="50",
                      command=lambda: mostrar_frame(f_calendario))
boton_Calendario.grid(row="1", column="0", sticky="we")

# Boton que muestra el frame de personalizar
boton_Personalizar = Button(frameMenu,bg=color2, text="Personalizar", heigh="8", width="20", font="50",
                    command=lambda: mostrar_frame(f_personalizar))
boton_Personalizar.grid(row="2", column="0", sticky="we")

# Boton que muestra el frame de configuración
boton_Configuracion = Button(frameMenu,bg=color2, text="Configuración", heigh="8", width="20", font="50",
                     command=lambda: mostrar_frame(f_configuracion))
boton_Configuracion.grid(row="3", column="0", sticky="we")


# Iniciar mostrando el frame de agenda
mostrar_frame(f_agenda)
```

```python
# Frame Agenda

# labels superiores, muestran titulo (agenda), y fecha que se muestra
textoFecha = StringVar()
textoFecha.set(str(dia_numero) + " de " + nombre_mes + " del " + str(ano_numero))

label_titulo = Label(f_agenda,bg=color3, text="Agenda", font="10").pack(fill="x")
label_mostrar_fecha = Label(f_agenda,bg=color3, textvariable=(textoFecha), font="10").pack(fill="x")


# boton para cambiar de día y para agregar actividad, corren sus funciones respectivas
frameBotones = Frame(f_agenda,bg=color1)
frameBotones.pack(anchor="n")
frameBotones.config(heigh=2)

botonCambio = Button(frameBotones,bg=color2, text="Cambiar de día", font="10", command=cambiar_fecha)
botonCambio.pack(side="left")

ventana_agregar_Boton = Button(frameBotones,bg=color2, text="+", font="500", heigh="3", width="5", command=agregarAct)
ventana_agregar_Boton.pack(side="top")

hacer_frame_tabla_hoy()
```

```
    # frame table diaria
    f_tablaDiaria = Frame(f_agenda, bg=color1)
    f_tablaDiaria.pack(anchor="n", fill="x", expand="True")
    f_tablaDiaria.config(heigh=height_screen / 2 - 60)
    f_tablaDiaria.grid_propagate(False)


    #Que las columnas tengan ancho
    for column in range(13):
        Grid.columnconfigure(f_tablaDiaria, column, weight=1)

    # Columnas de la tabla diaria
    columna_Act_diaria = Label(f_tablaDiaria,bg=color3, text="       Actividad Diaria       ", font=10).grid(row=0, column=0)
    columna_Hora_diaria = Label(f_tablaDiaria,bg=color3, text="       Hora       ", font=10).grid(row=0, column=1)
    columna_Importancia_diaria = Label(f_tablaDiaria,bg=color3, text="       Importancia       ", font=10).grid(row=0, column=2)
    columna_Descripcion_diaria = Label(f_tablaDiaria,bg=color3, text="       Descripción       ", font=10).grid(row=0, column=3)
    columna_Realizada_diaria = Label(f_tablaDiaria,bg=color3, text="       Realizada       ", font=10).grid(row=0, column=4)


    #Ya cuando se creó el frame, se muestran las actividades

    mostrar_acts()
```

```
# Frame Calendario

#Frames de la pantalla calendario
frame_opciones_calendario = Frame(f_calendario, bd=5)

f_calendario_mes = Frame(frame_opciones_calendario, height=height_screen*0.1, width=width_screen, bg=color3)
f_calendario_mes.grid(row=0, column=0, sticky='news')

#Iniciar mostrando mes
mostrar_frame(f_calendario_mes)

# Hacer calendario
frame_opciones_calendario = Frame(f_calendario, bg=color3, height=100, width=100)
frame_opciones_calendario.pack(side=TOP)


#funcion para crear el calendario
hacer_calendario(ano_numero, mes_numero)


#Boton Mostrar
lista_actividades=Label(frame_opciones_calendario, bg=color3, text="",width=4)
lista_actividades.pack(side=BOTTOM)
boton_seleccionar_calendario_mes= Button(frame_opciones_calendario,bg=color2, text="Mostrar actividades", width = 30)
boton_seleccionar_calendario_mes.pack(side=TOP)
```

```
#frame personalizar
#Iconos
icono1=PhotoImage(file="icono1.ppm")
icono2=PhotoImage(file="icono2.ppm")
icono3=PhotoImage(file="icono3.ppm")
icono4=PhotoImage(file="icono4.ppm")
icono5=PhotoImage(file="icono5.ppm")

f_seleccionar_Icono_personalizado = Frame(f_personalizar, bg=color1, heigh=100, width= width_screen*.6)
f_seleccionar_Icono_personalizado.pack(side=TOP, expand= FALSE)

label_seleccionar_icono = Label(f_seleccionar_Icono_personalizado, bg=color3, font= ('Consolas', 14), text= "Selección de Icono")
label_seleccionar_icono.pack(fill=X, expand=FALSE)

seleccion_Icono_Scroll= Scrollbar(f_seleccionar_Icono_personalizado, orient=HORIZONTAL)
lista_iconos= Listbox(f_seleccionar_Icono_personalizado, bg=color3, xscrollcommand=seleccion_Icono_Scroll.set)

seleccion_Icono_Scroll.config(command=lista_iconos.xview)
seleccion_Icono_Scroll.pack(side=BOTTOM, fill=X)
lista_iconos.pack(padx= 15)

boton_icono1= Canvas(lista_iconos, width= 184, bg=color2)
boton_icono1.pack(side=LEFT, expand=FALSE)
boton_icono1.create_image(1,1, anchor=NW, image=icono1)
boton_icono1.bind("<Button-1>", click1)

boton_icono2= Canvas(lista_iconos, width= 184, bg=color2)
boton_icono2.pack(side=LEFT, expand=FALSE)
boton_icono2.create_image(1,1, anchor=NW, image=icono2)
boton_icono2.bind("<Button-1>", click2)
```

```
boton_icono3= Canvas(lista_iconos, width= 184, bg=color2)
boton_icono3.pack(side=LEFT, expand=FALSE)
boton_icono3.create_image(1,1, anchor=NW, image=icono3)
boton_icono3.bind("<Button-1>", click3)

boton_icono4= Canvas(lista_iconos, width= 184, bg=color2)
boton_icono4.pack(side=LEFT, expand=FALSE)
boton_icono4.create_image(1,1, anchor=NW, image=icono4)
boton_icono4.bind("<Button-1>", click4)

boton_icono5= Canvas(lista_iconos, width= 184,bg=color2)
boton_icono5.pack(side=LEFT, expand=FALSE)
boton_icono5.create_image(1,1, anchor=NW, image=icono5)
boton_icono5.bind("<Button-1>", click5)

#Temas(Cada valor se cambia al volver a cargar la aplicación: Los colores aún siguen siendo feos)
f_vacio=Frame(f_personalizar, bg=color1, heigh=20,  width= width_screen*.6)
f_vacio.pack()

f_seleccionar_tema_personalizado = Frame(f_personalizar, bg=color3, heigh=100,  width= width_screen*.6)
f_seleccionar_tema_personalizado.pack(expand=FALSE)

label_seleccionar_tema = Label(f_seleccionar_tema_personalizado, bg=color3, font= ('Consolas', 14), text= "Selección de Tema")
label_seleccionar_tema.pack(fill=X, expand=FALSE)

seleccionar_tema_scrollbar=Scrollbar(f_seleccionar_tema_personalizado, orient=HORIZONTAL)
listatemas=Listbox(f_seleccionar_tema_personalizado, xscrollcommand=seleccionar_tema_scrollbar.set)

seleccionar_tema_scrollbar.config(command=listatemas.xview)
seleccionar_tema_scrollbar.pack(side=BOTTOM, fill=X)
listatemas.pack(padx=15)
```

```
Tema1= Button(listatemas, text= "Devs2",height=2, width= 32, command=lambda: color_tema1(), bg="DarkSlateGray3")
Tema1.pack(side=LEFT, expand=FALSE)

Tema2= Button(listatemas, text= "Staid",height=2, width= 32, command=lambda: color_tema2(), bg="MistyRose2")
Tema2.pack(side=LEFT, expand=FALSE)

Tema3= Button(listatemas, text= "Joy",height=2, width= 32, command=lambda: color_tema3(), bg="CadetBlue1")
Tema3.pack(side=LEFT, expand=FALSE)

Tema4= Button(listatemas, text= "Deadly",height=2, width= 32, command=lambda: color_tema4(), bg="DeepPink3")
Tema4.pack(side=LEFT, expand=FALSE)

Tema5= Button(listatemas, text= "West",height=2, width= 32, command=lambda: color_tema5(), bg="coral1")
Tema5.pack(side=LEFT, expand=FALSE)

# Frame Configuración
foto_perfil=PhotoImage(file=str(Icono))
nombre=""
cumpleanos=""

f_acomodo_config=Frame(f_configuracion,bg=color3)
f_acomodo_config.pack(side=TOP)

#Este frame es para arreglar un problema de distribución de causa desconocida

f_arreglo=Frame(f_configuracion,height=1000,width=2000,bg=color1)
f_arreglo.pack(side=RIGHT)

Nombre=""
Cumple=""
```

```
f_perfil=Frame(f_acomodo_config, bg=color3, heigh=200,  width=184)
f_perfil.grid(row=0, column=0)

Etiqueta_perfil=Label(f_perfil,bg=color3,text="Perfil", font=('Consolas',14))
Etiqueta_perfil.pack(side=TOP, fill=X)

Icono_Perfil=Canvas(f_perfil,height=184, width=184,bg=color3)
Icono_Perfil.pack()
Icono_Perfil.create_image(1,1, anchor=NW, image=foto_perfil)

l_nombre=Label(f_perfil, bg=color3, text="Nombre: " + str(Nombre))
l_nombre.pack()

l_cumpleanos=Label(f_perfil, bg=color3, text="Cumpleaños: " + str(Cumple))
l_cumpleanos.pack()


f_config=Frame(f_acomodo_config, bg=color3,height=258, width= width_screen*.6)
f_config.grid(row=0, column=1)

Cambiar_nombre=Button(f_config,bg=color2,height=2,text="Cambiar nombre")
Cambiar_nombre.grid(row=0, column=0)

Cambiar_cumple=Button(f_config,bg=color2,height=2,text="Cambiar cumpleaños")
Cambiar_cumple.grid(row=0, column=1)

# Se cierra el mainloop
raiz.mainloop()
```

Correr funciones iniciales

Se utiliza un try, except para crear los archivos en caso de que falten.

```
963
964     try:
965         valores_de_color()
966     except:
967         color_tema1()
968         click1("event")
969         valores_de_color()
970
971
972     perfil()
973     inicio()
974
975     #Fin :)
```

Funciones que fueron más retantes

Creo que aunque parece simple, la función que más nos costó trabajo fue la de borrar_linea_archivo_hoy y borrar_linea_archivo_diario, ya que tuvimos que pensar mucho para encontrar una forma de borrar todas las actividades y volverlas a mostrar cuando se cambia de fecha y cuando se marca como realizado. También la función de guardar_actividades_y_cerrar fue compleja, ya que se manejaron muchas cadenas de texto y se tenía que ir viendo cómo separar los distintos atributos de las actividades (fecha, hora, importancia etc.) para luego mostrarlos de manera correcta.