

Instituto Tecnológico de Costa Rica

I Proyecto

Mario Cerdas Esquivel

Ingeniería en Computadores

Grupo 01: Taller a la programación

Jeff Schmidt Peralta

Lunes 03 de octubre

Tabla de contenidos

Introducción	Página 3
Descripción del problema	Páginas de la 4 hasta la 5
Análisis de resultados	Páginas de la 6 hasta la 37
Dificultades encontradas	Página de la 38
Bitácora	Páginas de la 39 hasta la 43
Estadística de tiempos	Página 44
Conclusión	Página 45
Fuentes consultadas	Página 46

Introducción

El trabajo consta de documentar todo lo que se hizo en el proyecto, como describir lo que se hizo en código, analizar lo hecho, comentar dificultades, llevar un registro de los días trabajados, además de una estadística. En esta parte del trabajo se explicará un poco más a fondo lo que se hizo en el código además de adjuntar su respectiva evidencia con cada escenario posible, con cada posibilidad del funcionamiento que tenga el código como tal, tomando en cuenta lo que pueda hacer un futuro usuario que no sea el creador y que no esté muy familiarizado con la progra o con el uso de una máquina expendedora.

Como tal esto es más una evidencia tal vez un poco más legible y ordenada de explicar lo hecho en código.

Descripción del problema

En esta sección se mencionarán la descripción de los problemas y en la siguiente parte su respectivo código y funcionamiento. En esta parte simplemente se hace mención a los problemas resueltos de una manera breve.

- Como poner una imagen: Con el uso de la biblioteca 'os' y con el uso de tkinter.
- Creación de la ventana del about, con sus respectivas características. Se necesitaba agregar información pedida.
- Creación de la maquina expendedora, con sus imágenes representativas de cada producto y con una forma que el usuario ingrese cual producto quiere y cuanta cantidad quiere.
- Función que valide los datos ingresados puestos en las entradas de la máquina. Y si son correctos que calcule el subtotal
- Función que suma o resta lo a pagar
- Función que calcula el posible vuelto y valida pago. Y si debe de dar cambio lo dará de manera inteligente.
- Funciones que leen los archivos csv
- Función que sobrescribe en las tablas csv después de la compra, esta función no se nota en la parte de la interfaz, pero si se nota en la sobre escritura de las tablas.
- Función de contraseña, verificara si la contraseña es correcta y si no lo es cerrara el programa.

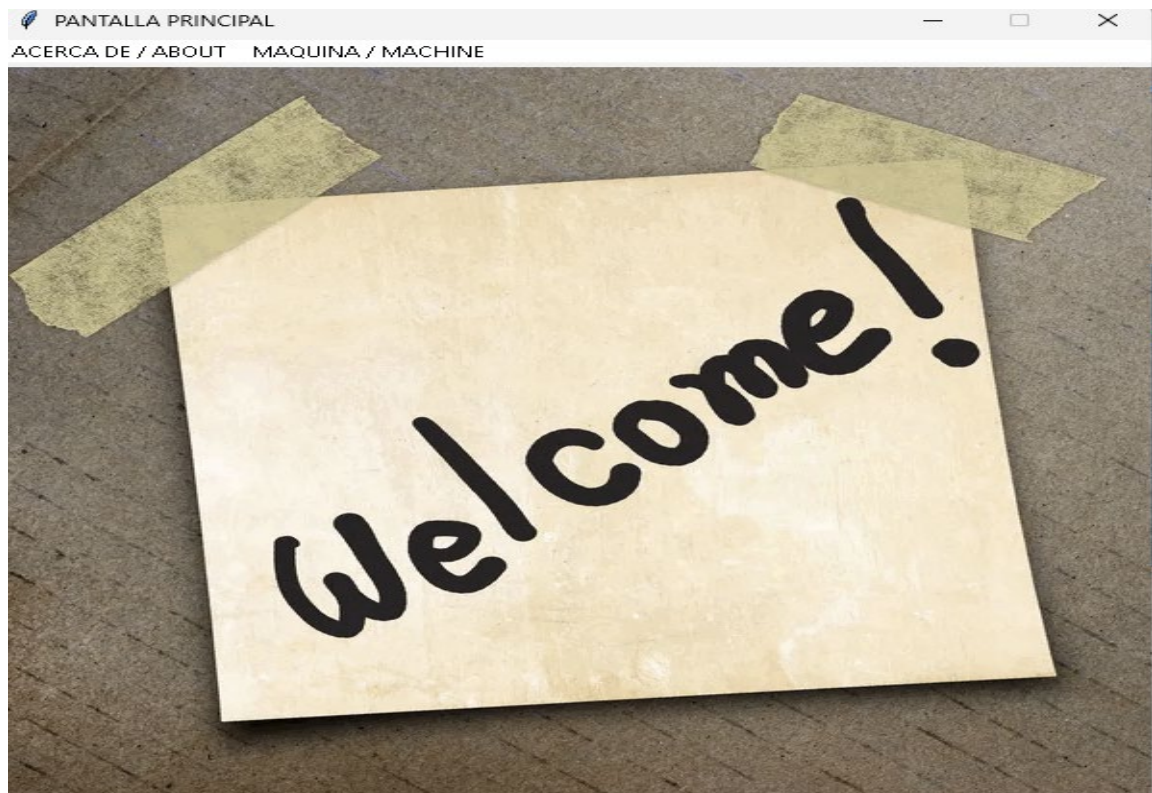
- Ventana de administración, aquí estarán la opción de cerrar el programa, de limpiar la tabla de movimientos y dos resúmenes de venta.
- Ajustes finales antes de ejecutar el código

Análisis de resultados

En esta sección se mostrarán capturas de pantalla de la parte del código y de su resultado en la parte de la interfaz gráfica. Otros detalles se explicarán en el video.

- Colocación de la imagen

```
"""
Fondo: esta funcion resivira el nombre de una imagen para usarla
E: una iamgen
R: esta depende de del formato ya sea .png o .gif
S: retorna la imagen
"""
def Fondo(img):
    ruta = os.path.join("Adicionales",img)
    imagen = PhotoImage(file=ruta)
    return imagen
```



```

Img = Fondo("Welcome.gif") |
LblFondo = Label(ventana, image = Img).place(x = 0,y = 0)

```

- Código de la pantalla del about y su respectivo funcionamiento

```

def about():
    """
    Características de la ventana del about(color de fondo,tamaño de la ventana, si se puede maximizar o no y
    su titulo)
    """
    about = Toplevel(bg = "RoyalBlue4")
    about.geometry("1200x500+70+70")
    about.resizable(width = NO, height = NO)
    about.title("ACERCA DE / ABOUT")

    """
    Informacion pedida, puesta e un label, se le da una letra distinta a la predeterminada y otras características
    """

    #Informacion del estudiante...
    info=Label( about, text = "-> Pais:\n"
                "\tCosta Rica\n"
                "-> Universidad:\n"
                "\tInstituto Tecnológico de Costa Rica\n\t\tTEC)\n"
                "-> Carrera:\n"
                "\tComputer Engineering \n"
                "-> Curso:\n"
                "\tTaller a la programacion,\n\t\tIISemestre 2022, Grupo 01\n "
                "-> Profesor:\n"
                "\tJeff Smchmidt Peralta\n"
                "-> Estudiante:\n"
                "\tMario Cerdas Esquivel\n "
                "-> Bibliotecas:\n"
                "\tTkinter, csv, pillow(PIL), os, time\n"

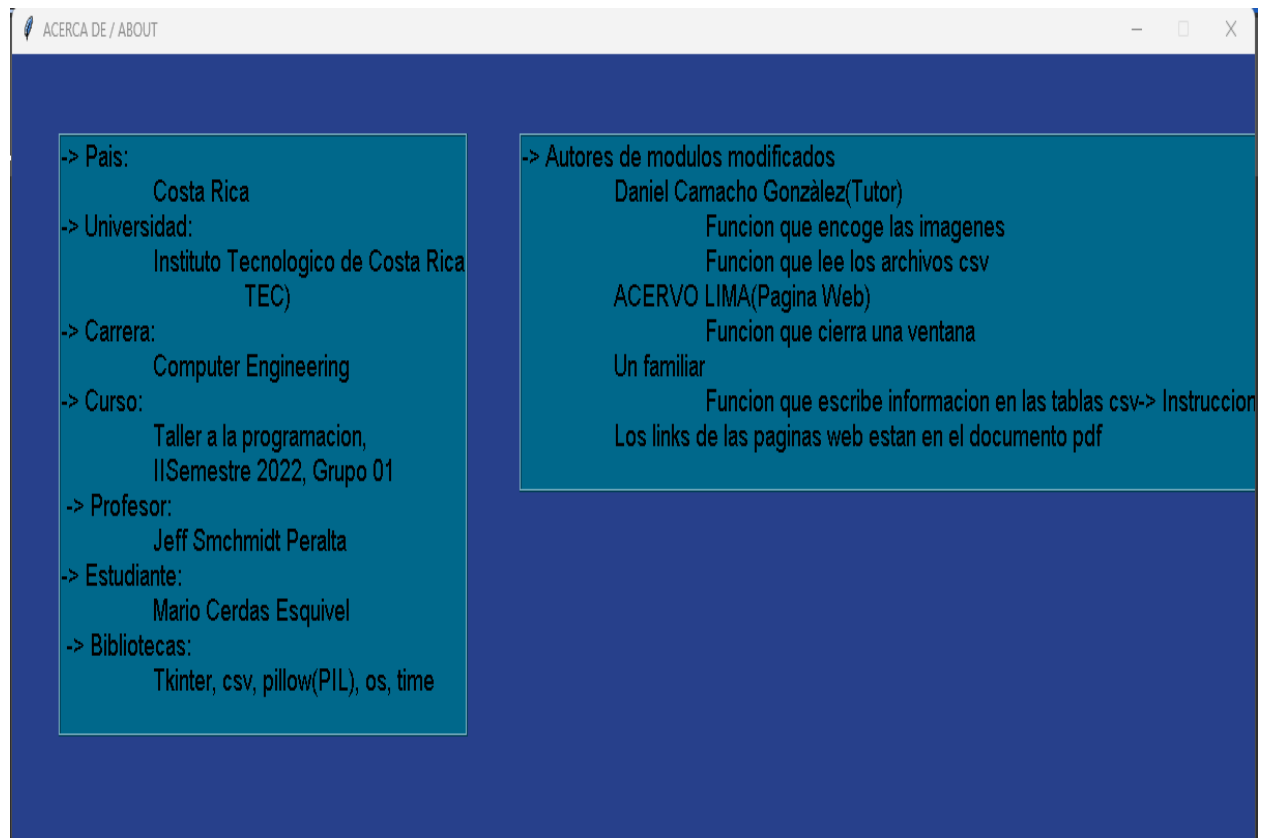
                ,bg = "DeepSkyBlue4",fg = "black",justify = "left",relief = "ridge",font = ("Arial",14))
    info.place(x = 45,y = 50)

```

```

#Informacion de autores de modulos y de Instrucciones importantes
info2 = Label (about, text = "-> Autores de modulos modificados\n"
                    "\tDaniel Camacho González(Tutor)\n"
                    "\t\tFuncion que encoge las imagenes\n"
                    "\t\tFuncion que lee los archivos csv\n"
                    "\tACERVO LIMA(Pagina Web)\n"
                    "\t\tFuncion que cierra una ventana\n"
                    "\tUn familiar\n"
                    "\t\tFuncion que escribe informacion en las tablas csv"
                    "-> Instrucciones importantes\n"
                    "\tLos links de las paginas web estan en el documento pdf\n"
                    ,bg = "DeepSkyBlue4",fg = "black",justify = "left",relief = "ridge", font =("Arial",14))
info2.place(x = 490, y =50)

```



- Código y todas las características y adiciones de la maquina

```
def Maquina():
    """
    Caracteristicas de la ventana donde se ubicara la maquina expendedora(color de fondo tamaño de la ventana,
    si se puedemaximizar o no y su titulo)
    """
    ventana.withdraw()
    maquina = Toplevel(bg = "black")
    maquina.geometry("900x680+250+5")
    maquina.resizable(width = NO, height = NO)
    maquina.title("MAQUINA / MACHINE")

    """
    Imagenes de los productos que posee la maquina expendedora, una imagen que simula la puerta donde se sacan
    los productos y un label con el nombre del fabricante.Tambien su respectivo label con el precio y el codigo
    """

    #Son 16 Productos y una iamen que simula una puerta

    "Bebidas"

    #Producto
    Coca = Imagenes("Coke.png", (64,64))
    IMG = Label(maquina,image = Coca)
    IMG.pack()
    IMG.place(x = 50, y = 120)

    #Label del precio
    CocaLbl = Label(maquina, text = "1: $800"
                    ,bg = "white",fg = "black", font =("Arial",12))
    CocaLbl.place(x = 51, y = 190)

    #Producto
    FantaN = Imagenes("FantaNaranja.png", (64,64))
    IMG1 = Label(maquina,image = FantaN)
    IMG1.pack()
    IMG1.place(x = 150, y = 120)

    #Label del precio
    FantaNLbl = Label(maquina, text = "9: $800"
                      ,bg = "white",fg = "black", font =("Arial",12))
    FantaNLbl.place(x = 151, y = 190)
```

```

#Producto
FantaU = Imagenes("FantaUva.png", (64,64))
IMG2 = Label(maquina, image = FantaU)
IMG2.pack()
IMG2.place(x = 250, y = 120)

#Label del precio
CocaLbl = Label(maquina, text = "8: €800"
               ,bg = "white",fg = "black", font =("Arial",12))
CocaLbl.place(x = 251, y = 190)

#Producto
Gatorade = Imagenes("gatorade.png", (64,64))
IMG3 = Label(maquina, image = Gatorade)
IMG3.pack()
IMG3.place(x = 350, y = 120)

#Label del precio
gatoradeLbl = Label(maquina, text = "6: €1000"
                   ,bg = "white",fg = "black", font =("Arial",12))
gatoradeLbl.place(x = 351, y = 190)

#Producto
Pepsi = Imagenes("Pepsi.png", (64,64))
IMG4 = Label(maquina, image = Pepsi)
IMG4.pack()
IMG4.place(x = 50, y = 230)

#Label del precio
PepsiLbl = Label(maquina, text = "2: €700"
                 ,bg = "white",fg = "black", font =("Arial",12))
PepsiLbl.place(x = 51, y = 300)

#Producto
TropicalF = Imagenes("tropicalfrutas.png", (64,64))
IMG5 = Label(maquina, image = TropicalF)
IMG5.pack()
IMG5.place(x = 150, y = 230)

#Label del precio
TropicalFLbl = Label(maquina, text = "4: €800"
                    ,bg = "white",fg = "black", font =("Arial",12))
TropicalFLbl.place(x = 151, y = 300)

```

```

#Producto
Power = Imagenes("Power.png", (64,64))
IMG6 = Label(maquina, image = Power)
IMG6.pack()
IMG6.place(x = 250, y = 230)

#Label del precio
PowerLbl = Label(maquina, text = "5: €1000"
                 ,bg = "white",fg = "black", font = ("Arial",12))
PowerLbl.place(x = 251, y = 300)

#Producto
Sprite = Imagenes("Sprite.png", (64,64))
IMG7 = Label(maquina, image = Sprite)
IMG7.pack()
IMG7.place(x = 350, y = 230)

#Label del precio
SpriteLbl = Label(maquina, text = "10: €800"
                  ,bg = "white",fg = "black", font = ("Arial",12))
SpriteLbl.place(x = 351, y = 300)

#Producto
Agua = Imagenes("Agua.png", (64,64))
IMG8 = Label(maquina, image = Agua)
IMG8.pack()
IMG8.place(x = 50, y = 340)

#Label del precio
AguaLbl = Label(maquina, text = "3: €700"
                ,bg = "white",fg = "black", font = ("Arial",12))
AguaLbl.place(x = 51, y = 410)

#Producto
Naranja = Imagenes("naranja.png", (64,64))
IMG9 = Label(maquina, image = Naranja)
IMG9.pack()
IMG9.place(x = 150, y = 340)

```

```

#Label del producto
NaranjaLbl = Label(maquina, text = "7: €700"
                  ,bg = "white",fg = "black", font =("Arial",12))
NaranjaLbl.place(x = 151, y = 410)

"Snacks"

#Producto
Hershey = Imagenes("Hershey.png", (64,64))
IMG10 = Label(maquina,image = Hershey)
IMG10.pack()
IMG10.place(x = 250, y = 340)

#Label del precio
HersheyLbl = Label(maquina, text = "16: €1000"
                  ,bg = "white",fg = "black", font =("Arial",12))
HersheyLbl.place(x = 251, y = 410)

#Producto
Pringles0 = Imagenes("PringlesOriginal.png", (64,64))
IMG11 = Label(maquina,image = Pringles0)
IMG11.pack()
IMG11.place(x = 350, y = 340)

#Label del precio
Pringles0Lbl = Label(maquina, text = "13: €1000"
                    ,bg = "white",fg = "black", font =("Arial",12))
Pringles0Lbl.place(x = 351, y = 410)

#Producto
Snickers = Imagenes("Snickers.png", (64,64))
IMG12 = Label(maquina,image = Snickers)
IMG12.pack()
IMG12.place(x = 50, y = 450)

#Label del precio
SnickersLbl = Label(maquina, text = "15: €1000"
                   ,bg = "white",fg = "black", font =("Arial",12))
SnickersLbl.place(x = 51, y = 520)

```

```

#Producto
TakisF = Imagenes("TakisFuego.png", (64,64))
IMG13 = Label(maquina, image = TakisF)
IMG13.pack()
IMG13.place(x = 150, y = 450)

TakisFLbl = Label(maquina, text = "12: €800"
                  ,bg = "white",fg = "black", font = ("Arial",12))
TakisFLbl.place(x = 151, y = 520)

#Producto
TakisX = Imagenes("Takisxplosion.png", (64,64))
IMG14 = Label(maquina, image = TakisX)
IMG14.pack()
IMG14.place(x = 250, y = 450)

TakisXLbl = Label(maquina, text = "11: €800"
                  ,bg = "white",fg = "black", font = ("Arial",12))
TakisXLbl.place(x = 251, y = 520)

#Producto
PringlesC = Imagenes("PringlesSourCrema.png", (64,64))
IMG15 = Label(maquina, image = PringlesC)
IMG15.pack()
IMG15.place(x = 350, y = 450)

#Label del precio
PringlesCLbl = Label(maquina, text = "14: €1000"
                     ,bg = "white",fg = "black", font = ("Arial",12))
PringlesCLbl.place(x = 351, y = 520)

#Puerta
puerta = Imagenes("puerta.gif", (200,100))
IMG16 = Label(maquina, image = puerta)
IMG16.pack()
IMG16.place(x = 130, y = 570)

#Label del nombre
maquina_i = Label(maquina, text = "CE COMPANY"
                  ,bg = "light sea green",justify = "left",relief = "ridge",font = ("GOUDY STOUT",20))
maquina_i.place(x = 80,y = 50)

```

```

"""
En esta seccion se agreganron los Label y los entry para los productos y cantidades adeamas los botones de
aceptar y el de administrar
"""

#Label Y entry de la cantidad
Cantidad = Label(maquina, text = "Introduzca la cantidad \n Enter the amount",
                 bg="DodgerBlue4",fg="black",font= ("Arial Black",20))
Cantidad.place(x=500, y = 50)
Entrada_info = Entry(maquina, width = 10, bg = "DeepSkyBlue4",font = ("Arial Ce",20))
Entrada_info.place(x = 590, y = 150)

#Label y entry del producto
Productos = Label(maquina,text = "Elija el producto \n Chooose the product",
                 bg="DodgerBlue4",fg="black",font= ("Arial Black",20))
Productos.place(x=500, y=200)
Entrada_info1 = Entry(maquina, width = 10, bg = "DeepSkyBlue4",font = ("Arial Ce",20))
Entrada_info1.place(x = 570, y = 300)

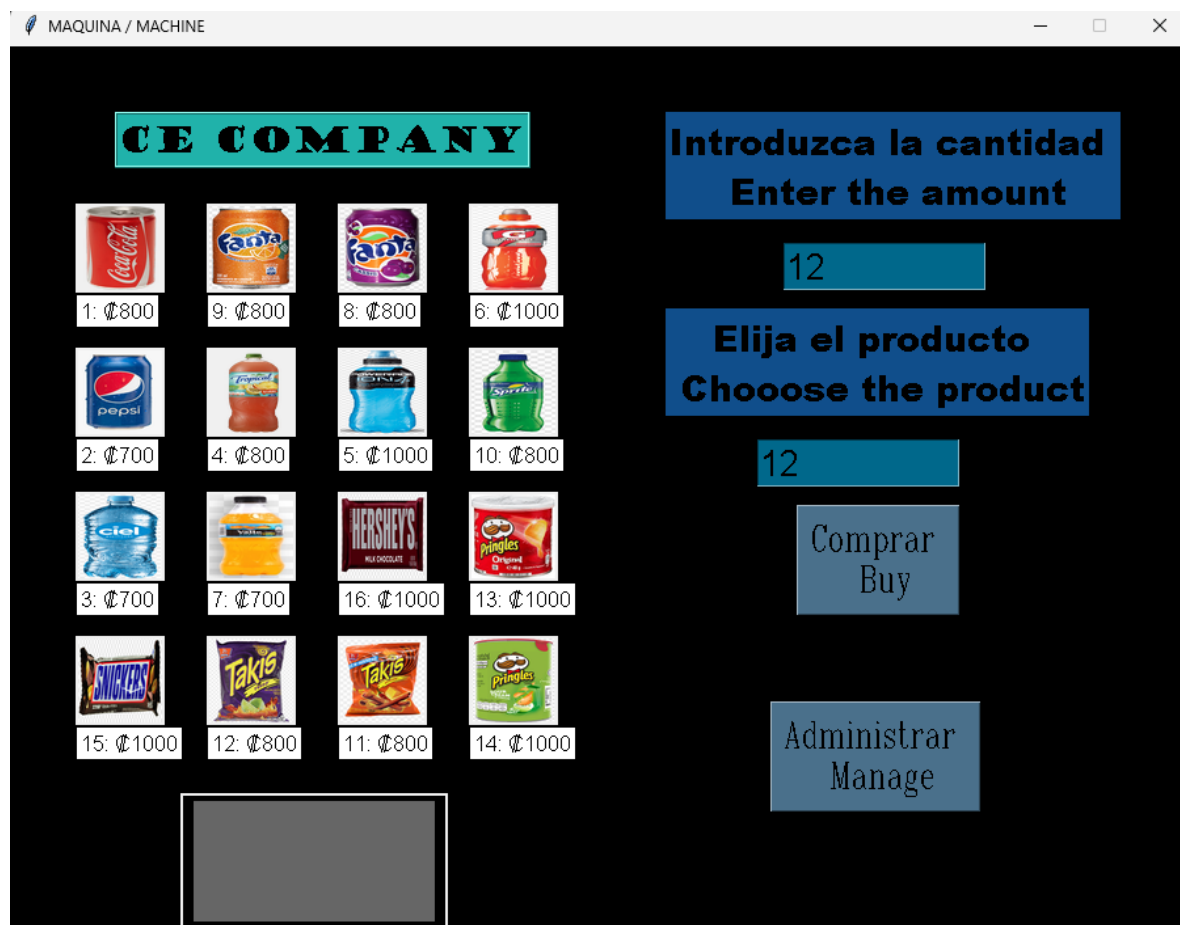
#Boton de Comprar
Comprar = Button(maquina,text = "Comprar \n Buy",bg="SkyBlue4",fg = "black",font = ("Roman",20),
                command = lambda: ingresar_dato(int(Entrada_info.get()),int(Entrada_info1.get()))))
Comprar.place(x = 600, y = 350)

#Boton de administrar
admin = Button(maquina, text = "Administrar \n Manage",bg="SkyBlue4",fg = "black",font = ("Roman",20),
               ,command = lambda: Password())
admin.place(x = 580, y = 500)

maquina.mainloop()

```





```

#####
total: total a pagar
E: cantidad de articulos
R: una cantidad adecuada(permitida)
S: el total
#####
def total(cantidad,codigo,subtotal,cantidad2):
    if cantidad == 0:
        Pago(subtotal,codigo,cantidad2)
        return "Listo, cantidad a pagar lista"
    elif cantidad == 1:
        subtotal = int(productos[codigo-1][5])
        return total(cantidad - 1, codigo,subtotal,cantidad2)
    else:
        subtotal = int(productos[codigo-1][5]) * cantidad
        return total(cantidad - cantidad, codigo,subtotal,cantidad2)

```




- Función de validación de datos

```

"""
Funcion que resive un argumento entero
E: un numero
R: entero
S: una funcion auxiliar
"""
def ingresar_dato(n,codigo):
    if n <= 15 and codigo <= 16 :
        return producto(n, codigo, 0)
    else:
        messagebox.showinfo( "Error", "El numero de productos debe ser menor a\n"
                                "16,el codigo debe ser entre 1 a 16"
                                "\n\nThe number of products must be less"
                                "than 1, the code must be between 1 to 16")
        return "Error, el dato no se pudo convertir"

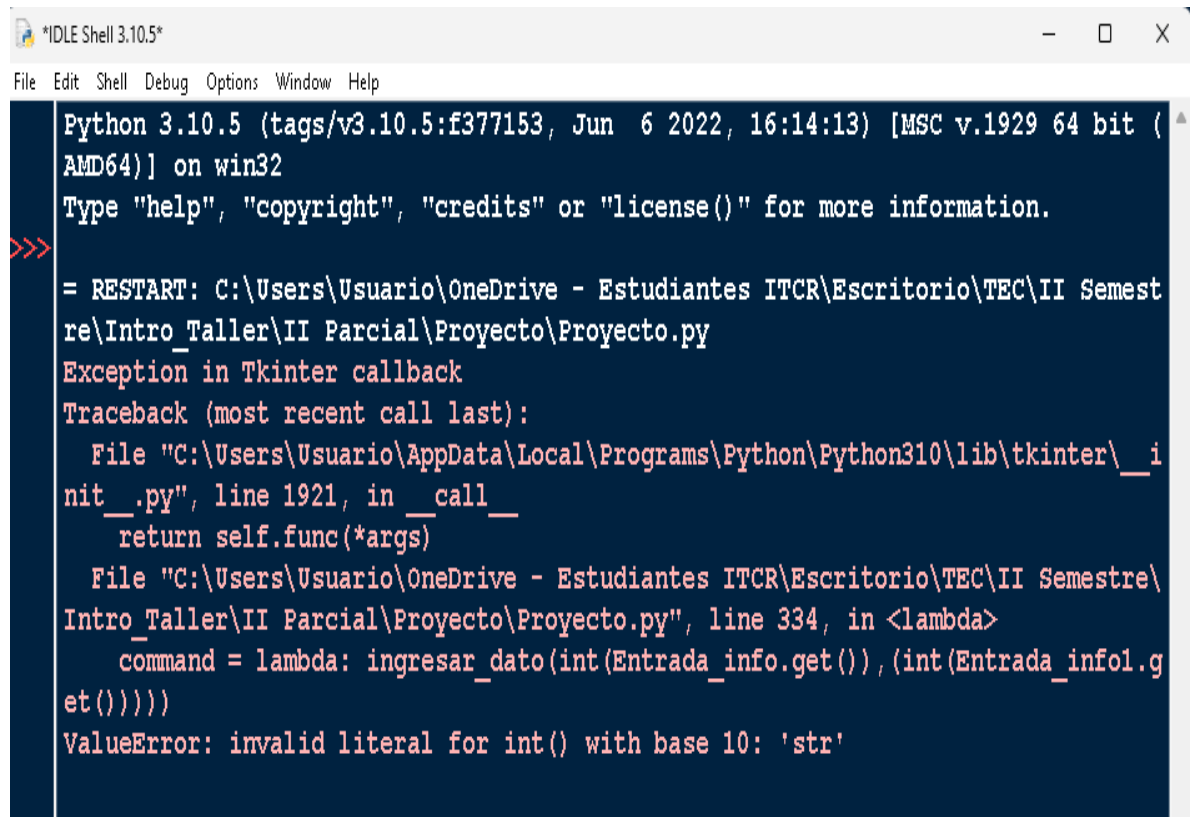
"""
producto: resive n, codigo, indice
E: numeros enteros
R: previamente verificada en la funcion anterior
S: Si hay productos suficientes
"""
def producto(n,codigo,indice):
    if indice == len(productos):
        messagebox.showinfo("Error", "El producto no esta en la maquina\n\n"
                                "The product is not in the machine")
        return False,False,-1
    elif str(codigo) == productos[indice][0] and int(productos[indice][2]) < n:
        messagebox.showinfo("Producto agotado//Sold out", "Ya no quedan productos a la venta\n\n"
                                "There are no more products for sale")
        return True,False,indice
    elif str(codigo) == productos[indice][0] and int(productos[indice][2]) >= n:
        total(n,codigo,0,n)
        return True,True,indice

    return producto(n, codigo, indice + 1)

```

Si se recibe un **str** no se podrá avanzar





The screenshot shows a Python IDLE Shell window titled "*IDLE Shell 3.10.5*". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The shell displays the following text:

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Usuario\OneDrive - Estudiantes ITCR\Escritorio\TEC\II Semestre\Intro_Taller\II Parcial\Proyecto\Proyecto.py
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\Usuario\AppData\Local\Programs\Python\Python310\lib\tkinter\__init__.py", line 1921, in __call__
    return self.func(*args)
  File "C:\Users\Usuario\OneDrive - Estudiantes ITCR\Escritorio\TEC\II Semestre\Intro_Taller\II Parcial\Proyecto\Proyecto.py", line 334, in <lambda>
    command = lambda: ingresar_dato(int(Entrada_info.get()), (int(Entrada_info.get()))))
ValueError: invalid literal for int() with base 10: 'str'
```

En cambio, si se recibe un **int**, pasara a la siguiente pantalla



Cabe recalcar que al pasar a la siguiente pantalla ya habrá calculado el
subtotal a pagar



Total, a pagar



- Comportamiento ventana pago

```

"""
En esta seccion se crea la ventana para pagar por los productos
E: los valores del subtotal,codigo y cantidad ademas de los de el valor de los billetes
R: que sean enteros
S: retorna la funcion del vuelto
"""
def Pago(subtotal,codigo,cantidad):
    """
    actualizar: resive como parametros los spinbox y hace la operacion correspondiente
    E: los spibox y el subtotal
    R: enteros
    S: el total a dar
    """
    def actualizar(cantidad,codigo,m100,m500,b1000,b2000):

        #Label
        Falta = Label(pago, text = "Falta // Lack",bg="DodgerBlue4",
                      fg="black",font= ("Arial Black",20))
        Falta.place(x = 50, y = 350)

        #operacion

        total1 = m100 * 100 + m500 * 500 + b1000 * 1000 + b2000 * 2000
        etiquetal = Label(pago, font=("Arial ",20), text="Total : ¢ " + str((total1) - subtotal)
                          ,height=1,width=20)
        etiquetal.place(x=250,y=350)

        Pagar = Button(pago, text = "Pay\\Pagar", bg = "SkyBlue4", fg = "black", font = ("Roman",20)
                      ,command =lambda: vuelto(int(total2),codigo,cantidad,Dinero100,
                                              Dinero500,Dinero1000,Dinero2000,total2,total3,total4))

        Pagar.place(x = 600, y = 350)

    """
    Variables del vuelto y el total pagado y el subtotal
    """
    total2 = 0 #vuelto
    total3 = 0 #total
    total4 = 0 #subtotal
    total2 += total1 - subtotal
    total3 += total1

```

```

Dinero100 = m100
Dinero500 = m500
Dinero1000 = b1000
Dinero2000 = b2000

pago = Toplevel()
pago.geometry("1200x500+100+150")
pago.resizable(width = False, height = False)
pago.title("PAGAR / PAY")

fondo1 = Fondo("fondo-dolares.png")
fondo2 = Canvas(pago,width = 1200, height = 500)
fondo2.pack()
fondo2.create_image(0,0,image = fondo1,anchor= NW)
fondo2.place(x = 0, y = 0)

"Moneda"

#Moneda 100
M100_colones = Imagenes("100 colones.png", (150,150))
IMG17 = Label(pago,image = M100_colones)
IMG17.pack()
IMG17.place(x = 50, y = 50)

#spinbox
m100 = IntVar()
moneda100 = Spinbox(pago, from =0,to =10,state="readonly", textvariable = m100, font =("Roman",20),
                    command = lambda:actualizar(cantidad,codigo,m100.get(),m500.get(),
                                                b1000.get(),b2000.get())
                    ,justify = CENTER, width=13)
moneda100.place(x = 50, y = 200)

#Moneda 500
M500_colones = Imagenes("500 colones.png", (150,150))
IMG18 = Label(pago,image = M500_colones)
IMG18.pack()
IMG18.place(x = 300, y = 50)

#spinbox
m500 = IntVar()
moneda500 = Spinbox(pago, from =0,to =10,state="readonly", textvariable = m500, font =("Roman",20),
                    command = lambda:actualizar(cantidad,codigo,m100.get(),m500.get(),
                                                b1000.get(),b2000.get())
                    , justify = CENTER, width=13)

```



```

moneda500.place(x = 300, y= 200)

"Billete"

#Billete
B1000_colones = Imagenes("1000-Colones.png", (150,150))
IMG19 = Label(pago,image = B1000_colones)
IMG19.pack()
IMG19.place(x = 550, y = 50)

#spinbox
b1000 = IntVar()
billete1000 = Spinbox(pago, from_=0,to =10,state="readonly", textvariable = b1000, font =("Roman",20),
                      command = lambda:actualizar(cantidad,codigo,m100.get(),m500.get(),
                                                  b1000.get(),b2000.get()),
                      justify = CENTER, width=13)
billete1000.place(x = 550, y = 200)

#Billete
B2000_colones = Imagenes("2000 Colones.png", (150,150))
IMG20 = Label(pago,image = B2000_colones)
IMG20.pack()
IMG20.place(x = 750, y = 50)

#spinbox
b2000 = IntVar()
billete2000 = Spinbox(pago, from_=0,to =10,state="readonly", textvariable = b2000, font =("Roman",20),
                      command = lambda:actualizar(cantidad,codigo,m100.get(),m500.get(),
                                                  b1000.get(),b2000.get()),
                      justify = CENTER, width=13)
billete2000.place(x = 750, y = 200)

pago.mainloop()

```

Automáticamente a mover las fechas para el pago el monto se actualiza



En el caso de que el usuario Introduzca mas de la cuenta el monto en positivo que se muestra se le vuelto que retornara.



- Función del vuelto y validación de pago

```

"""
actualizar: valida primero que se alla pagado por completo, y despues si el vuelto es cero retornara una actualizacion
de la lista de dinero, pero si es mayor retornara la funcion que dara el vuelto.
2: numeros enteros
"""
def vuelto(vuelto,codigo,cantidad,Dinero100,Dinero500,Dinero1000,Dinero2000,devolver,pago,debe):
    if vuelto < 0:
        messagebox.showinfo("Error", "Aun nos terminado de pagar\n\n"
                             "You haven't finished paying yet")
        return "Error, no se ha terminado de pagar"
    elif vuelto == 0:
        messagebox.showinfo("Gracias//Thanks","Thaks for buying\n\n"
                             "Gracias por comprar")
        Sumar(Dinero100,Dinero500,Dinero1000,Dinero2000)
        actualizar_movimientos(codigo,cantidad,devolver,pago,debe)
        cambiarcantidad(codigo,cantidad,0)
        return Cerrar()
    else:
        Sumar(Dinero100,Dinero500,Dinero1000,Dinero2000)
        vuelto_aux(vuelto,codigo,cantidad,0,0,0,0,vuelto,devolver,pago,debe)
        return "Listo"

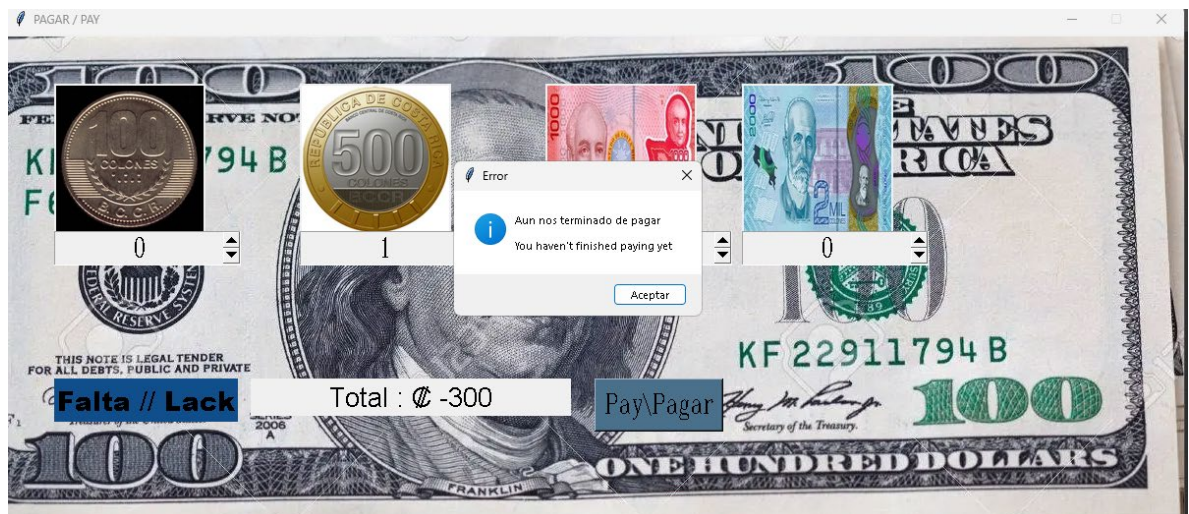
```

```

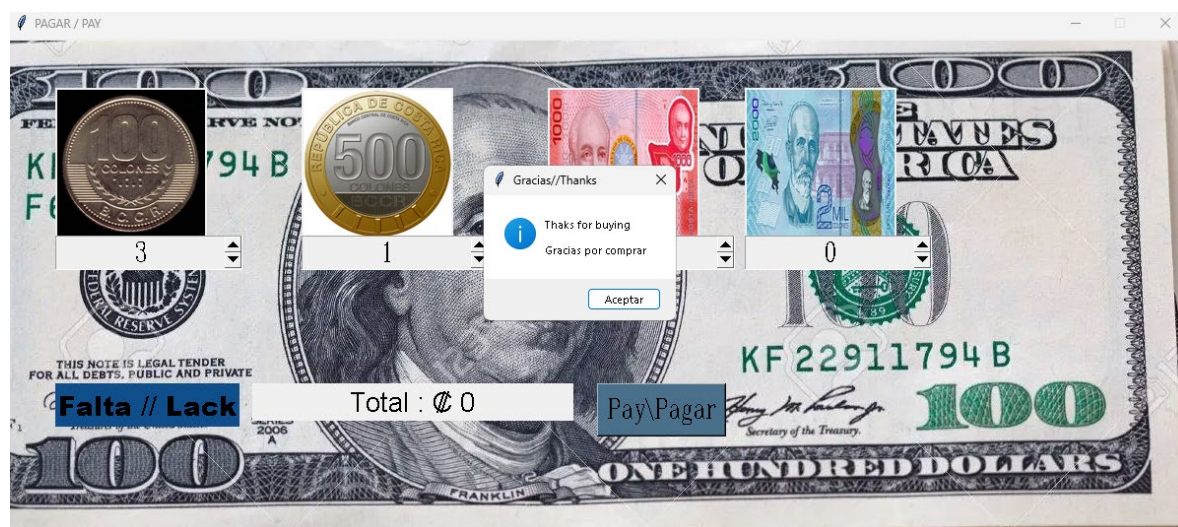
"""
Funcuion que verifica cuanto debe de otorgar de vuelto.
E: El numero de vuelto que debe de dar
R: Enteros
S: El vuelto de una manera en la que se le da prioridad a las denominaciones mas altas
"""
def vuelto_aux(vuelto,codigo,n, mil, dosmil, quinientos, cien, total,devolver,pago,debe):
    if vuelto == 0:
        messagebox.showinfo("Vuelto//Turned", "Su vuetto es de//You return is from:\t\n"
            + "¢2000:\t" + str(dosmil)+"\n¢1000:\t" + str(mil) +
            "\n¢500:\t" + str(quinientos) + "\n¢100:\t" + str(cien) +
            "\n\nTotal:\t" + "¢" + str(total))
        vendido(codigo,n,mil,dosmil,quinientos,cien,devolver,pago,debe)
        return True
    elif vuelto != 0 and vuelto >= 2000:
        return vuelto_aux(vuelto - 2000, codigo,n, mil, dosmil + 1, quinientos, cien, total,devolver,pago,debe)
    elif vuelto != 0 and vuelto >= 1000:
        return vuelto_aux(vuelto - 1000, codigo,n, mil + 1, dosmil, quinientos, cien, total,devolver,pago,debe)
    elif vuelto != 0 and vuelto >= 500:
        return vuelto_aux(vuelto - 500, codigo,n, mil, dosmil, quinientos + 1, cien, total,devolver,pago,debe)
    elif vuelto != 0 and vuelto >= 100:
        return vuelto_aux(vuelto - 100, codigo,n, mil, dosmil, quinientos, cien + 1, total,devolver,pago,debe)
    else:
        messagebox.showinfo("Error", "Algo salio mal\n\n"
            "Something went wrong")
        return "Error, algo sucedio con el vuelto"

```

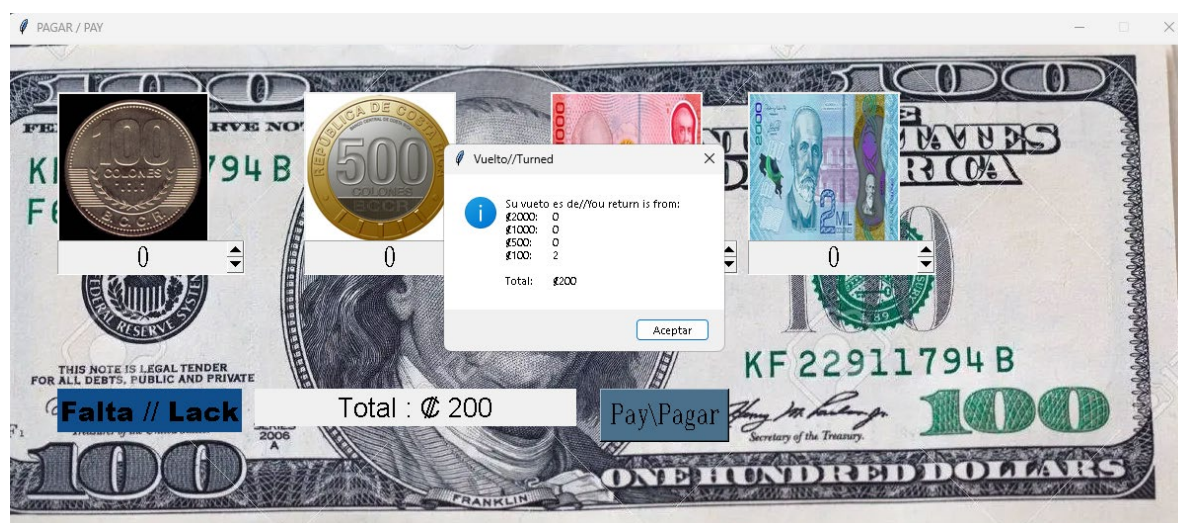
En caso de pagar con un monto inferior



En caso de pagar exacto



En caso de pagar de mas



- Leer archivo

```
"""
Funciones encargada de leer el archivo de productos, movimientos y dinero
Estas funciones simplemente leen los archivos.
"""

"""
Productos
"""
def leerP():
    with open("Productos.csv", 'r') as csvfile:
        csvreader = csv.reader(csvfile)
        for line in csvreader:
            productos.append(line)

"""
Movimientos
"""
def leerM():
    with open("Movimientos.csv", 'r') as csvfile:
        csvreader = csv.reader(csvfile)
        for line in csvreader:
            movimientos.append(line)

"""
Dinero
"""
def leerD():
    with open("Dinero.csv", 'r') as csvfile:
        csvreader = csv.reader(csvfile)
        for line in csvreader:
            dinero.append(line)
```

- Función después de comprar

```

"""
vendido: funcion que se retornara cuando el usuario pagara y se le diera vuelto si este fuese necesario, reescribira
en los archivos de texto de movimientos, dinero y productos.
E: los datos del vuelto, la lista de productos y lo pagado
R: los datos se van a convertir a un valor apropiado para su manipulacion
S: El arreglo a cada una de las listas
"""
def vendido(codigo,n,mil,dosmil,quinientos,cien,devolver,pago,deben):
    if mil == 0 and dosmil == 0 and quinientos == 0 and cien == 0:
        cambiarcantidad(codigo,n,0)
        guardarDinero()
        actualizar_movimientos(codigo,n,devolver,pago,deben)
        Cerrar()
        return True
    elif dosmil > 0:
        dinero[3][2] = str(int(dinero[3][2]) - dosmil)
        return vendido(codigo,n, mil, dosmil - dosmil, quinientos, cien,devolver,pago,deben)
    elif mil > 0:
        dinero[2][2] = str(int(dinero[2][2]) - mil)
        return vendido(codigo,n, mil - mil, dosmil, quinientos, cien,devolver,pago,deben)
    elif quinientos > 0:
        dinero[1][2] = str(int(dinero[1][2]) - quinientos)
        return vendido(codigo,n, mil, dosmil, quinientos - quinientos,cien,devolver,pago,deben)
    elif cien > 0:
        dinero[0][2] = str(int(dinero[0][2]) - cien)
        return vendido(codigo,n, mil, dosmil, quinientos, cien - cien,devolver,pago,deben)

```

```

"""
Sumar: resive los datos con los cuales se pagaron el productos y los suma
E: los datos con los cuales se pagaron
R: enteros
S: guardarDinero
"""
def Sumar(Dinero100,Dinero500,Dinero1000,Dinero2000):
    if Dinero100 == 0 and Dinero2000 == 0 and Dinero500 == 0 and Dinero100 == 0:
        guardarDinero()
        return True
    elif Dinero2000 > 0:
        dinero[3][2] = str(int(dinero[3][2]) + Dinero2000)
        return Sumar(Dinero100,Dinero500,Dinero1000,Dinero2000 - Dinero2000)
    elif Dinero1000 > 0:
        dinero[2][2] = str(int(dinero[2][2]) + Dinero1000)
        return Sumar(Dinero100 - Dinero100,Dinero500,Dinero1000 - Dinero1000 ,Dinero2000)
    elif Dinero500 > 0:
        dinero[1][2] = str(int(dinero[1][2]) + Dinero500)
        return Sumar(Dinero100,Dinero500 - Dinero500,Dinero1000,Dinero2000)
    elif Dinero100 > 0:
        dinero[0][2] = str(int(dinero[0][2]) + Dinero100)
        return Sumar(Dinero100 - Dinero100,Dinero500,Dinero1000,Dinero2000)

```

```

"""
cambiarcantidad: actualiza la cantidad de productos
E: enteros
R: que sea la lista correcta
S: la lista pructos actualizada
"""
def cambiarcantidad(codigo,cantidad,indice):
    if indice == len(productos):
        messagebox.showinfo("Error","Hubo un problema\n\n"
                             "There was a problem")

        Cerrar()
        return False
    elif str(codigo) == productos[indice][0]:
        productos[indice][2] = str(int(productos[indice][2]) - cantidad)
        productos[indice][6] = str(int(productos[indice][6]) + cantidad)
        guardarProductos()
        return True

    return cambiarcantidad(codigo,cantidad, indice + 1)

```

```

"""
Estas funciones se encargan de guardar los cambios que se le hicieron a las listas
"""

"Dinero"

def guardarDinero():
    csvfile = open("Dinero.csv",'w',newline='')
    with csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(dinero)

"Productos"

def guardarProductos():
    csvfile = open("Productos.csv",'w',newline='')
    with csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(productos)

```

```

"""
actualizar_movimientos: resive los valores nesesaros para agragarlos a la tabla de movimientos
E: enteros
R:
S: la tabla actualizada
"""
def actualizar_movimientos(codigo,n,devolver,pago,deben):
    largo = len(movimientos)
    trasaccion = (len(movimientos) +1)
    movimientos.append(['VE',trasaccion,time.asctime(),codigo,n,deben,pago,devolver])
    guardarmovimientos()
#####

"""
guardar_movimientos: hace un append a la tabla de movimientos
"""
def guardarmovimientos():
    csvfile = open("Movimientos.csv", 'w', newline="")
    with csvfile:
        writer = csv.writer(csvfile)
        writer.writerows(movimientos)
#####

```

- Password

```

"""
Password: ventana que crea la entrada y el boton para la contraseña
E: una contraseña
R: no hay
S: la funcion correct
"""
def Password():
    bloqueo = Toplevel(bg = "SteelBlue4")
    bloqueo.geometry("400x300+750+380")
    bloqueo.resizable(width = False, height = False)
    bloqueo.title("LOOK")

    """
    Label con el texto de "Ingresar la imagen", su entrada y un boton que retorna la funcion correct()
    """

    #Label

    Ingrese = Label(bloqueo, text = "Ingrese la contraseña\nEnter the password",bg="DodgerBlue4",
                    fg="black",font= ("Arial Black",20))
    Ingrese.place(x = 25, y = 50)

    #Entry

    Contraseña = Entry(bloqueo, width = 10, bg = "DeepSkyBlue4",font = ("Arial Ce",20))
    Contraseña.place(x = 125, y = 150)

    #Boton

    Aceptar = Button(bloqueo, text = "Aceptar\nAcept", bg = "SkyBlue4", fg = "black", font = ("Roman",20)
                    , command = lambda: correct(Contraseña.get()))
    Aceptar.place(x = 150, y = 200)

    bloqueo.mainloop()

```



```
"""  
correct: validad la contraseña  
E: la contraseña  
R: ninguna  
S: la ventana administrador o cierra la ventana  
"""  
def correct(contraseña):  
    if "Computer" == contraseña:  
        Administrador()  
    else:  
        Cerrar()
```

Si la contraseña es incorrecta el programa se cierra.



Si es correcta se abre el administrador



- Ventana administradora

```

"""
Funcion que crea una ventana con una especie de menu utilizando diferentes botones
"""
def Administrador():
    """
    Caracteristicas de la ventana, tamaño, titulo, fondo y si se puede maximizar o no.
    """
    admin = Toplevel(bg = "deep sky blue")
    admin.geometry("500x500+500+50")
    admin.resizable(width = NO, height = NO)
    admin.title("ADMINISTRADOR")

    fondo = Fondo("administrador.gif")
    fondo1 = Canvas(admin,width = 500, height = 500)
    fondo1.pack()
    fondo1.create_image(10,10,image = fondo,anchor= NW)
    fondo1.place(x = 0, y = 0)

    """
    Esta seccion tendra unos botones que ejecutaran algunas funciones
    Boton_Cerrar: Cerrara el programa
    """

    #Apagar
    Boton_Cerrar = Button(admin,text = "Close\n Cerrar",bg="SkyBlue4",fg = "red",font = ("Roman",20)
        ,command = Cerrar)
    Boton_Cerrar.pack(pady = 20)

    #Boton del reset
    Boton_reset = Button(admin,text = "Resetear\nReset", bg = "SkyBlue4", fg = "black", font = ("Roman",20)
        ,command = reset)
    Boton_reset.place(x = 50, y = 250)

    #Boton para el resumen de ventas
    Boton_resumen = Button(admin,text = "Resumen\nSumary", bg = "SkyBlue4", fg = "black", font = ("Roman",20)
        ,command = reporte_resumen)
    Boton_resumen.place(x = 350, y = 250)

    #Boton del resumen detallado
    Boton_resumenD = Button(admin,text = "Resumen Detallado\nDetailed Summary", bg = "SkyBlue4",
        command = reporte_detallado, fg = "black", font = ("Roman",20))
    Boton_resumenD.place(x = 130, y = 150)

```

```

"""
reporte_resumen: da un reporte resumido de ventas, con el codigo, producto,existencias, vendidas, monto
E: las tablas globales de movimientos y productos
R: las tablas correctas
S: el resumen en una ventana nueva
"""
def reporte_resumen():
    """
    Caracteristicas de la ventana
    """
    resumen = Toplevel(bg="cyan4")
    resumen.geometry("1200x500+100+150")
    resumen.resizable(width = False, height = False)
    resumen.title("RESUMEN / SUMMARY")

    #Labels

    Resumen = Label(resumen, text = "Codigo//Code\tDescripccion//Description\tCantidad//Amount\t"
        "Vendidas//Sold\tMonto//Amount",bg="DodgerBlue4",
        fg="black",font= ("Arial",18))
    Resumen.place(x = 35, y = 50)

    label = Label(resumen, text="Out of Service//Fuera de Servicio",bg="DodgerBlue4",
        fg="black",font= ("Arial",18))
    label.place(x=50,y=100)

    resumen.mainloop()

#####

```

```

"""
reporte_detallado: da un reporte detallado de ventas, con el codigo, producto, #Transaccion, fecha, cantidad,
monto
E: las tablas globales de movimientos y productos
R: las tablas correctas
S: el detalle de ventas en una ventana nueva
"""
def reporte_detallado():
    """
    Caracteristicas de la ventana
    """
    detalle = Toplevel(bg="cyan4")
    detalle.geometry("1200x500+100+150")
    detalle.resizable(width = False, height = False)
    detalle.title("Detail / DETAIL")

    Detalle = Label(detalle, text = "Codigo//Code\tDescripccion//Description\tCantidad//Amount\t"
                                   "#Trnsaccion//Trnsaction\tFecha//Date\tMonto//Amount",bg="DodgerBlue4",
                                   fg="black",font= ("Arial",12))
    Detalle.place(x = 35, y = 50)

    #ILabels

    label = Label(detalle, text="Out of Service//Fuera de Servicio",bg="DodgerBlue4",
                  fg="black",font= ("Arial",18))
    label.place(x=50,y=100)

    detalle.mainloop()

```

```

"""
Cerrar: esta funcion cierra por completo el programa
"""
def Cerrar():
    ventana.destroy()

```

```

"""
reset: limpia la tabla de movimientos
E: la global de movimientos
R: que sea la tabla correcta
S: la tabla limpia
"""
def reset():
    LimpiarMovimientos()

```

```

"""
Esta funcion es la que guarda el reset()
"""
def LimpiarMovimientos():
    csvfile = open("Movimientos.csv", 'w', newline='')
    with csvfile:
        writer = csv.writer(csvfile)
        writer.writerows([])

```

- Ajuste de la ventana principal

```

"""
Se crea la ventana principal y se le dan características de tamaño y el de poder o no minimizar o maximizar pantalla
y se le pone un nombre, además se pone un label de fondo
"""
ventana = Tk()
ventana.title("PANTALLA PRINCIPAL")
ventana.geometry("600x600+350+50")
ventana.resizable(width = NO, height = NO)
Img = Fondo("Welcome.gif")
LblFondo = Label(ventana, image = Img).place(x = 0, y = 0)

"""
Se crea el menu y sus secciones las cuales son el about y la Maquina, esta seccion crea un comando para que cuando
se le de click habra la funcion indicada
"""
menu = Menu(ventana)
menu.add_command(label = "ACERCA DE / ABOUT", command = about)
menu.add_command(label = "MAQUINA / MACHINE", command = Maquina)

"""
Se inicializa las funciones para leer los archivos de texto creados
"""
leerP()
leerM()
leerD()

"""
Esta parte hace que funcione la ventanas y la seccion del menu, sin esta parte el código no correria (al menos
la parte grafica)
"""
ventana.config(menu = menu)
ventana.mainloop()

```

Dificultades encontradas

- Una de las dificultades con las que me tope al inicio era el no saber si las tablas debían de tener algún formato en específico para facilitar la lectura en Python y a su vez al usarlas con las diferentes bibliotecas. Lo que hice fue consultar con los tutores y uno de ellos me brindo ayuda para crearla en un archivo csv.
- También puede ser que no supiese por dónde empezar a hacer la máquina expendedora. Lo que hice para resolverlo fue empezar para la parte grafica de la interfaz.
- Una dificultad también fue que no podía cambiar el tamaño de la imagen si afectarla como tal, ósea hacerla más pequeña pero que se siga apreciando la imagen completa. Investigue y con ayuda de un tutor aplique una función que cambiara el tamaño de las imágenes
- Una dificultad también era una especie de impotencia porque llevaba días pegado en ciertas partes del proyecto y no podía avanzar. Descansaba un rato o me ponía hacer otros trabajos y pedía ayuda, para poder avanzar
- No sabía como hacer la función que recibieran los datos de la maquina para efectuar la compra. Se resuelve buscando ayuda de los tutores.
- Guardar los datos ajustados en el archivo de csv. Solicite ayuda a un familiar que sabia programar. Al final la función estaba bien sin embargo nunca guardaba los datos en la global porque en el código no se había implementado

Bitácora

Sábado 17 de septiembre desde las 7pm hasta las 8 y 20pm y desde las 9 y 40 pm hasta las 10pm.

Se estudia lo que se tiene que hacer para el proyecto, también se crean los archivos de texto, pero sin agregar nada aún. También se crea la ventana principal, un menú con el about y con el ingreso a la pantalla donde iría la máquina. Además, se crea la carpeta donde irían imágenes u otros documentos necesarios para el funcionamiento del código.

Domingo 18 de septiembre desde la 1pm hasta las 5 y 30pm y de 7 y 30 hasta las 9 y 30pm.

Se configura la pantalla del about, se investiga sobre el manejo de archivos .txt o .csv para la preparación de la máquina. También se agregan algunos detalles para una mejor interfaz en las primeras pantallas, se intenta sin éxito la creación de la primera tabla de información.

Martes 20 de septiembre desde las 5 y 30 pm hasta las 6 y 30pm y desde las 8pm hasta las 11 y 30 pm.

Se estudia la lectura y agregación de datos a un archivo de texto desde Python y aplico ya un formato para usar en las tablas que se deben de hacer con la información necesaria para el funcionamiento de la máquina en un archivo en Excel, pero guardado con un .csv para poder leerlo con Python.

Se empieza con la elaboración del documento añadiendo una portada, tabla de contenidos, la introducción, algunas dificultades encontradas hasta el momento y parte de la bitácora con lo trabajado hasta ahora. Se añaden datos a la estadística de tiempos.

Miércoles 21 de septiembre de 6pm a 7pm y de 9pm a 1am.

Se crean varias funciones para la validación de datos de las tablas, además se buscó imágenes para la interfaz de la maquina y se planea el cómo luciría dicha máquina, se busca información sobre el acomodo de imágenes en Python tkinter. Se intenta hacer la parte gráfica y vistosa de la máquina, pero no se logra. Se deja unas cuantas funciones a la espera de una prueba.

Jueves 22 de septiembre de 6 y 45pm hasta las 8pm y de 9 y 40pm hasta las 12 y 20am

Se busca funciones para minimizar el tamaño de las imágenes para su agregación a la máquina expendedora, se encuentra una forma sin embargo por el momento nos retorna error. Se empieza con la parte grafica de la máquina, como los Label y los entry para ingresar la cantidad y el producto además de sus respectivos botones como el de comprar y el del administrador donde este creara un menú para sus diferentes opciones, se documenta mejor la parte del código siendo más detallado y se escribe en la documentación externa.

Sábado 24 de septiembre de 10am hasta la 1 y 15pm, desde las 2 y 15 hasta las 5pm y desde las 9 y 30pm hasta las 10 y 30 pm

Se crea una función que agarre una imagen y le cambie su tamaño por una indicado, además de otra función que cierre el programa, sobre la función anterior se averigua el cómo hacerla en una función por aparte por si se desea llamar en alguna otra función.

Se hace la parte grafica de la máquina expendedora: se colocan las imágenes con su código y precio, se agrega una imagen para que simule más una máquina expendedora la cual simula la puerta donde se sacan los productos, se le pone un nombre a la máquina(en la parte superior con un nombre de fabricante),se le da colores más apropiados a la máquina para que parezca una, se crea un botón en la parte de administrar que cierra el programa, se arregla la documentación interna arreglando ciertos errores y a su vez agregándole más a las que era necesarias. Se agrega nueva información en la parte del about, además que se trabaja nuevamente el documento.

Martes 27 de septiembre de 11am hasta las 3pm, de 4pm hasta las 7 y 30 y de 9pm hasta las 11pm

Se afinan algunos detalles en las ventanas previamente echas, se arreglan algunas partes de la interfaz que no estaban aun en inglés, se investiga como hacer la parte de la solicitud de contraseña y también se implementa, se deja casi al 100% la parte de la interfaz hecha y se inicia con las funciones que la maquina reaccione a lo que el usuario haga. Se crea la

función que recibirá la cantidad de productos y el Código para así averiguar si el producto y la cantidad seleccionada son las correctas y si hay suficiente.

Se empieza con la realización del método de pago, creando una ventana donde se efectuará el pago (primero la interfaz para después hacer la lógica).

Viernes 30 de septiembre del 2022, de 3pm hasta las 9pm.

Se arreglan algunos errores que tenía la función que validaba el código y la cantidad. Se crean las funciones restantes para leer los otros archivos csv.

Se crea la función que calculara el pago y se le dan detalles a la ventana donde se pagara, agregando los medios de pago (monedas, billetes).

Sábado 01 de octubre de 1pm hasta las 4pm, de 4 y 30pm hasta las 7 y 30pm, de 9pm hasta las 12am.

Se crea la función que calcula el vuelto de manera correcta, se crea la función que ir cambiando los billetes y monedas cuando se efectué la compra, se pone la ventana de pago un poco más interactiva y fácil de usar con un contador automático y se crea la función de reset. Además, se trabaja en la documentación y se le hacen pruebas al código.

Domingo 2 de octubre de 6 y 30pm hasta la 1y 30am.

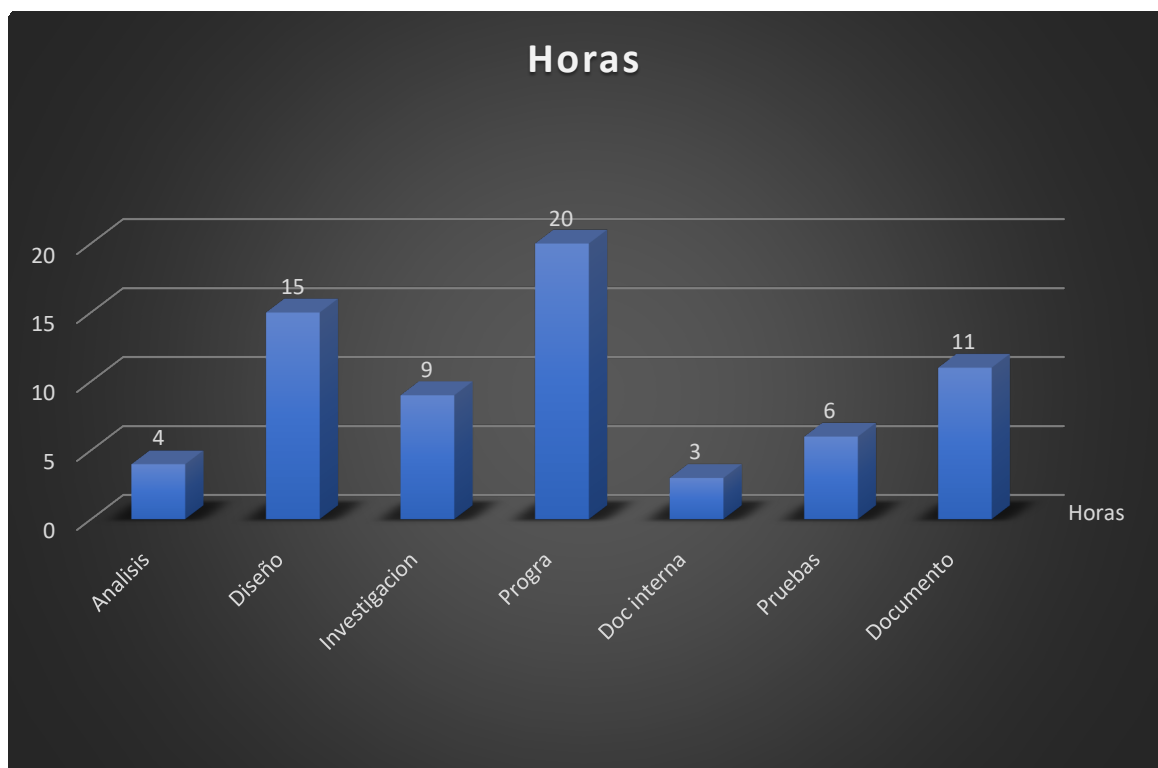
Se termina con la parte del pago del producto y sus validaciones, además se hace que después de cada compra se actualicen las tablas de productos, movimientos y dinero.

Lunes 3 de octubre de 10 y 30am hasta las 12 y 30pm y, de 1 y 10 hasta las 4 y 10pm, de 5pm hasta las 8 y 30pm y de 9 y 30 hasta las 11 y 01pm.

Se termina con la documentación interna, con la documentación externa y se hace el video. Además, se arreglaban algunas partes del código, se hace y se edita el video y se agrega en un drive para tener acceso al el.

Estadística de tiempos

Análisis de requerimientos	4 horas
Diseño de la aplicación	15 horas
Investigación de funciones	9 horas
Programación	20 horas
Documentación interna	3 horas
Pruebas	6 horas
Elaboración del Documento	11 horas
<u>Total</u>	68 horas



Video

<https://drive.google.com/drive/folders/12yN1OletIDlwX8h7U-4W2l0wEquBaVSD?usp=sharing>

Literatura o fuentes consultadas

- Tecnología Binaria, 20 jul 2021, “Cómo REDIMENSIONAR una IMAGEN con TKINTER y PILLOW | Curso de Python Intermedio #20”, [video]YouTube, <https://youtu.be/5ku3YVIY0x0>
- Arcervo Lima, <https://es.acervolima.com/como-cerrar-una-ventana-de-tkinter-con-un-boton/>
- Terrones Digital, 06 julio 2021, Python – Tkinter – Interfaz de usuario – Login, [video]YouTube, <https://youtu.be/heeghMrCMgY>