

JK LAKSHMIPAT UNIVERSITY

DIGITAL CIRCUIT AND SYSTEMS  
(EE1120)

PROJECT - 01

Vending Machine using Python

Programming language.

Date : 17th April 2024

Group Members :

* Aryaman Bohra - 2023BTECH016
* Bobby Sharma - 2023BTECH023

AIM – Design the Vending Machine using Tkinter library in python programming language.

REQUIREMENTS – Visual Studio Code, Basic knowledge of the Tkinter library of python Programming language.

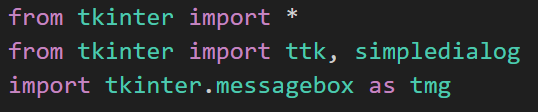
THEORY –

Finite State Machines (FSMs) are mathematical models used to represent and control the behavior of systems with a finite number of states. They are widely used in various fields, including computer science, engineering, and telecommunications, to model the behavior of systems that can be in one of a limited number of states at any given time.

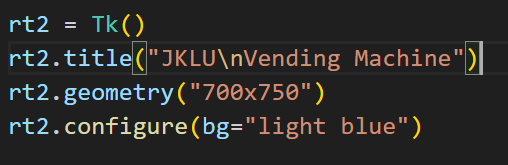
Designing a vending machine using Finite State Machine (FSM) theory involves modeling the system with various states, transitions, and actions. Each state represents a distinct condition the machine can be in, such as idle, product selection, payment processing, and product dispensing. Transitions occur based on specific events or inputs, like user selections or payments. Associated actions with each transition include displaying options, deducting payments, and dispensing products. Error handling mechanisms are also crucial for managing unexpected scenarios. Implementing FSM theory typically involves programming constructs to represent states and transitions, ensuring smooth interaction between users and the vending machine interface.

CODE EXPLANATION –

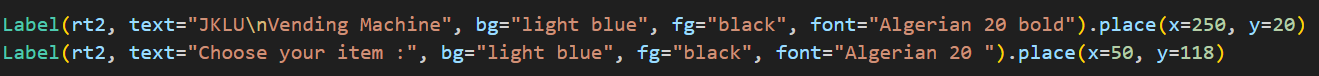
* Import Statements: The script begins by importing necessary modules from Tkinter and other libraries.



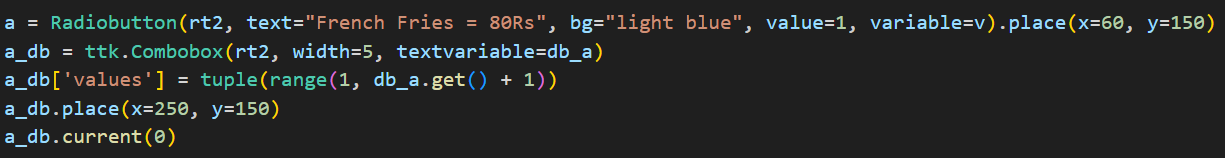
* Tkinter Setup: It initializes the main Tkinter window and sets its title, geometry, and background color.



* Labels: Several labels are added to the window for displaying text.



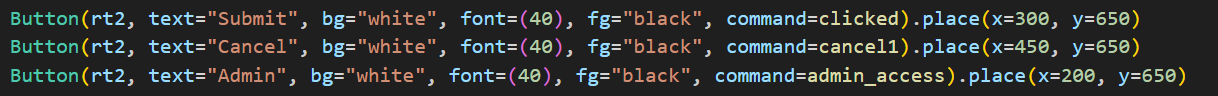
* Radio Buttons and Combo boxes: Radio buttons are used for selecting items, and combo boxes are used for selecting the quantity of each item.



* Functions: Two main functions are defined:

1. clicked(): It's called when the "Submit" button is clicked. It calculates the total bill and change based on the selected item and entered cash.
2. admin\_access(): It's called when the "Admin" button is clicked. It prompts for a password and displays the remaining quantity of each item if the correct password is entered.

* Buttons: Three buttons are added for submitting, canceling, and accessing admin functionalities.



* Icons: Attempts to set the window icon if available.
* Main Loop: Start the main event loop.



This script creates a simple GUI for a vending machine where users can select items, enter cash, and get change. It also has an admin access feature to view the remaining quantity of items.

CODE –

from tkinter import \*

from tkinter import ttk, simpledialog

import tkinter.messagebox as tmg

rt2 = Tk()

rt2.title("JKLU\nVending Machine")

rt2.geometry("700x750")

rt2.configure(bg="light blue")

Label(rt2, text="JKLU\nVending Machine", bg="light blue", fg="black", font="Algerian 20 bold").place(x=250, y=20)

Label(rt2, text="Choose your item :", bg="light blue", fg="black", font="Algerian 20 ").place(x=50, y=118)

# IntVar for checkboxes

a\_cb = IntVar(value=0)

b\_cb = IntVar(value=0)

c\_cb = IntVar(value=0)

d\_cb = IntVar(value=0)

e\_cb = IntVar(value=0)

f\_cb = IntVar(value=0)

g\_cb = IntVar(value=0)

h\_cb = IntVar(value=0)

i\_cb = IntVar(value=0)

j\_cb = IntVar(value=0)

k\_cb = IntVar(value=0)

l\_cb = IntVar(value=0)

m\_cb = IntVar(value=0)

n\_cb = IntVar(value=0)

# IntVar for comboboxes

db\_a = IntVar(value=10)

db\_b = IntVar(value=10)

db\_c = IntVar(value=10)

db\_d = IntVar(value=10)

db\_e = IntVar(value=10)

db\_f = IntVar(value=10)

db\_g = IntVar(value=10)

db\_h = IntVar(value=10)

db\_i = IntVar(value=10)

db\_j = IntVar(value=10)

db\_k = IntVar(value=10)

db\_l = IntVar(value=10)

db\_m = IntVar(value=10)

db\_n = IntVar(value=10)

a = Checkbutton(rt2, text="French Fries = 80Rs", bg="light blue", variable=a\_cb).place(x=60, y=150)

a\_db = ttk.Combobox(rt2, width=5, textvariable=db\_a)

a\_db['values'] = tuple(range(1, db\_a.get() + 1))

a\_db.place(x=250, y=150)

a\_db.current(0)

b = Checkbutton(rt2, text="Softy = 30Rs",bg = "light blue", variable=b\_cb).place(x=60, y=200)

b\_db = ttk.Combobox(rt2,width=5,textvariable=db\_b)

b\_db['values']=(1,2,3,4,5,6,7,8,9,10)

b\_db.place(x =250 ,y =200)

b\_db.current(0)

c = Checkbutton(rt2, text="Cup Cake = 20Rs",bg = "light blue", variable=c\_cb).place(x=60, y=250)

c\_db = ttk.Combobox(rt2,width=5,textvariable=db\_c)

c\_db['values']= (1,2,3,4,5,6,7,8,9,10)

c\_db.place(x =250 ,y =250)

c\_db.current(0)

d = Checkbutton(rt2, text="Dora Cake = 30Rs",bg = "light blue", variable=d\_cb).place(x=60, y=300)

d\_db = ttk.Combobox(rt2,width=5,textvariable=db\_d)

d\_db['values']= (1,2,3,4,5,6,7,8,9,10)

d\_db.place(x =250 ,y =300)

d\_db.current(0)

e = Checkbutton(rt2, text="Cold Coffee = 30Rs",bg = "light blue", variable=e\_cb).place(x=60, y=350)

e\_db = ttk.Combobox(rt2,width=5,textvariable=db\_e)

e\_db['values']= (1,2,3,4,5,6,7,8,9,10)

e\_db.place(x =250 ,y =350)

e\_db.current(0)

f = Checkbutton(rt2, text="Hot Coffee = 40Rs",bg = "light blue", variable=f\_cb).place(x=60, y=400)

f\_db = ttk.Combobox(rt2,width=5,textvariable=db\_f)

f\_db['values']= (1,2,3,4,5,6,7,8,9,10)

f\_db.place(x =250 ,y =400)

f\_db.current(0)

g = Checkbutton(rt2, text="Green Tea =30Rs",bg = "light blue", variable=g\_cb).place(x=60, y=450)

g\_db = ttk.Combobox(rt2,width=5,textvariable=db\_g)

g\_db['values']= (1,2,3,4,5,6,7,8,9,10)

g\_db.place(x =250 ,y =450)

g\_db.current(0)

h = Checkbutton(rt2, text="Tea = 20Rs",bg = "light blue", variable=h\_cb).place(x=350, y=150)

h\_db = ttk.Combobox(rt2,width=5,textvariable=db\_h)

h\_db['values']= (1,2,3,4,5,6,7,8,9,10)

h\_db.place(x =500 ,y =150)

h\_db.current(0)

i = Checkbutton(rt2, text="Ice Tea = 30Rs",bg = "light blue", variable=i\_cb).place(x=350, y=200)

i\_db = ttk.Combobox(rt2,width=5,textvariable=db\_i)

i\_db['values']= (1,2,3,4,5,6,7,8,9,10)

i\_db.place(x =500 ,y =200)

i\_db.current(0)

j = Checkbutton(rt2, text="Coca Cola = 40Rs",bg = "light blue", variable=j\_cb).place(x=350, y=250)

j\_db = ttk.Combobox(rt2,width=5,textvariable=db\_j)

j\_db['values']= (1,2,3,4,5,6,7,8,9,10)

j\_db.place(x =500 ,y =250)

j\_db.current(0)

k = Checkbutton(rt2, text="Pepsi = 40Rs",bg = "light blue", variable=k\_cb).place(x=350, y=300)

k\_db = ttk.Combobox(rt2,width=5,textvariable=db\_k)

k\_db['values']= (1,2,3,4,5,6,7,8,9,10)

k\_db.place(x =500 ,y =300)

k\_db.current(0)

l = Checkbutton(rt2, text="Mirinda = 40Rs",bg = "light blue", variable=l\_cb).place(x=350, y=350)

l\_db = ttk.Combobox(rt2,width=5,textvariable=db\_l)

l\_db['values']= (1,2,3,4,5,6,7,8,9,10)

l\_db.place(x =500 ,y =350)

l\_db.current(0)

m = Checkbutton(rt2, text="Lays = 20Rs", bg = "light blue", variable=m\_cb).place(x=350, y=400)

m\_db = ttk.Combobox(rt2,width=5,textvariable=db\_m)

m\_db['values']= (1,2,3,4,5,6,7,8,9,10)

m\_db.place(x =500 ,y =400)

m\_db.current(0)

n = Checkbutton(rt2, text="Kurkure = 20Rs",bg = "light blue", variable=n\_cb).place(x=350, y=450)

n\_db = ttk.Combobox(rt2,width=5,textvariable=db\_n)

n\_db['values']= (1,2,3,4,5,6,7,8,9,10)

n\_db.place(x =500 ,y =450)

n\_db.current(0)

Label(rt2, text="Enter Your Cash :", bg="light blue", fg="black", font=(20)).place(x=10, y=500)

Label(rt2, text="Total Bill :", bg="light blue", fg="black", font=(20)).place(x=10, y=550)

Label(rt2, text="Your Changes :", bg="light blue", fg="black", font=(20)).place(x=10, y=600)

s = IntVar()

o = Entry(rt2, font=(40), textvariable=s).place(x=200, y=500)

t = IntVar()

t1 = Entry(rt2, font=(40), textvariable=t).place(x=200, y=550)

s1 = IntVar()

p = Entry(rt2, font=(40), textvariable=s1).place(x=200, y=600)

def cancel1():

rt2.destroy()

def clicked():

# Get the amount entered by the user

i = s.get()

# Initialize the total bill

total\_bill = 0

# Calculate the total bill based on selected items and their quantities

if a\_cb.get() == 1 and db\_a.get() > 0:

quantity = db\_a.get()

total\_bill += quantity \* 80

if b\_cb.get() == 1 and db\_b.get() > 0:

quantity = db\_b.get()

total\_bill += quantity \* 30

if c\_cb.get() == 1 and db\_c.get() > 0:

quantity = db\_c.get()

total\_bill += quantity \* 20

if d\_cb.get() == 1 and db\_d.get() > 0:

quantity = db\_d.get()

total\_bill += quantity \* 30

if e\_cb.get() == 1 and db\_e.get() > 0:

quantity = db\_e.get()

total\_bill += quantity \* 30

if f\_cb.get() == 1 and db\_f.get() > 0:

quantity = db\_f.get()

total\_bill += quantity \* 40

if g\_cb.get() == 1 and db\_g.get() > 0:

quantity = db\_g.get()

total\_bill += quantity \* 30

if h\_cb.get() == 1 and db\_h.get() > 0:

quantity = db\_h.get()

total\_bill += quantity \* 20

if i\_cb.get() == 1 and db\_i.get() > 0:

quantity = db\_i.get()

total\_bill += quantity \* 30

if j\_cb.get() == 1 and db\_j.get() > 0:

quantity = db\_j.get()

total\_bill += quantity \* 40

if k\_cb.get() == 1 and db\_k.get() > 0:

quantity = db\_k.get()

total\_bill += quantity \* 40

if l\_cb.get() == 1 and db\_l.get() > 0:

quantity = db\_l.get()

total\_bill += quantity \* 40

if m\_cb.get() == 1 and db\_m.get() > 0:

quantity = db\_m.get()

total\_bill += quantity \* 20

if n\_cb.get() == 1 and db\_n.get() > 0:

quantity = db\_n.get()

total\_bill += quantity \* 20

# Set the total bill to the Entry widget

t.set(total\_bill)

# Check if the entered amount is sufficient

if i >= total\_bill:

# Calculate and set the change

change = i - total\_bill

s1.set(change)

else:

# Show an error message if the amount is insufficient

tmg.showerror('ERROR', "Insufficient amount")

def admin\_access():

password = simpledialog.askstring("Password", "Enter Admin Password:", show='\*')

if password == "admin":

remaining\_quantity = {

"French Fries": db\_a.get(),

"Softy": db\_b.get(),

"Cup Cake": db\_c.get(),

"Dora Cake": db\_d.get(),

"Cold Coffee": db\_e.get(),

"Hot Coffee": db\_f.get(),

"Green Tea": db\_g.get(),

"Tea": db\_h.get(),

"Ice Tea": db\_i.get(),

"Coca Cola": db\_j.get(),

"Pepsi": db\_k.get(),

"Mirinda": db\_l.get(),

"Lays": db\_m.get(),

"Kurkure": db\_n.get(),

}

remaining\_text = "\n".join([f"{product}: {quantity}" for product, quantity in remaining\_quantity.items()])

tmg.showinfo("Remaining Quantity", f"Remaining Quantity:\n{remaining\_text}")

else:

tmg.showerror("Admin Access", "Access Denied! Incorrect Password.")

Button(rt2, text="Submit", bg="white", font=(40), fg="black", command=clicked).place(x=300, y=650)

Button(rt2, text="Cancel", bg="white", font=(40), fg="black", command=cancel1).place(x=450, y=650)

Button(rt2, text="Admin", bg="white", font=(40), fg="black", command=admin\_access).place(x=200, y=650)

try:

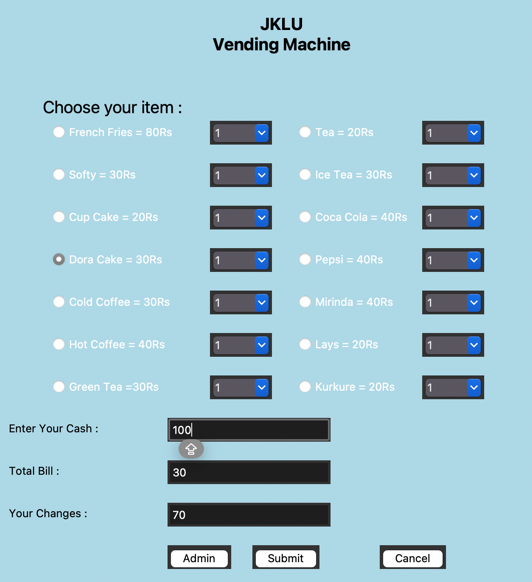
rt2.iconbitmap("C:\\Users\\sharm.LENOVO\\Downloads\\jklu.ico")

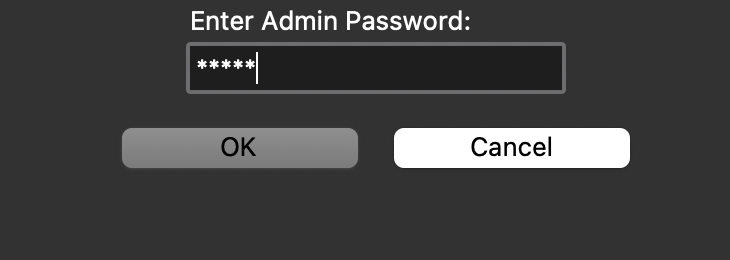
except:

raise

rt2.mainloop()

OBSERVATION –

  
- selecting our product, enter your cash amount, then submit.



-access the admin and check to check your stock



- As seen in the above interface you can see the remaining quantity

RESULTS –

The provided Python script utilizes Tkinter to construct a graphical user interface (GUI) simulating a vending machine system. Upon execution, it launches a window titled "JKLU Vending Machine" featuring a light blue backdrop. Within this interface, users are presented with various item options, including French Fries, Softy, Cup Cake, and more, each accompanied by their respective prices. Complementing this, combo boxes facilitate the selection of item quantities. A designated entry field allows users to input the amount of cash they wish to spend on their purchase. Upon submitting their selection, the script calculates the total bill based on the chosen items and their quantities. Additionally, it computes any change owed to the user if the entered cash exceeds the total bill. Error handling mechanisms promptly notify users if their cash input is insufficient or if invalid selections are made. Furthermore, an "Admin" button grants access to administrative functionalities, prompting users to input a password. Upon successful authentication, the system reveals the remaining quantities of each item in the vending machine. Finally, the script enters the main event loop, enabling continuous user interaction until the window is closed.

LEARNING –

* GUI Development: Building a graphical user interface (GUI) teaches you about designing user-friendly interfaces that users can interact with. Tkinter is a great library for beginners to learn GUI development in Python.
* Event Handling: Handling user interactions such as button clicks (e.g., Submit, Cancel) and item selections (e.g., radio buttons) requires understanding event-driven programming.
* Data Management: Managing data such as quantities of items, total bill, and cash input requires knowledge of data structures and manipulation techniques.
* Conditional Logic: Implementing logic for calculating the total bill, determining if the entered cash is sufficient, and providing appropriate feedback (e.g., error messages) involves using conditional statements.
* Error Handling: Dealing with errors or invalid inputs (e.g., insufficient cash) and providing meaningful error messages enhances the user experience.
* User Feedback: Providing feedback to users (e.g., displaying total bill, remaining quantity of items) is essential for a smooth user experience.
* Admin Functionality: Implementing admin functionalities, such as accessing privileged information (e.g., remaining quantity of items) with password protection, adds another layer of complexity to the project.
* Software Testing: Testing the application thoroughly to ensure all functionalities work as expected and handling edge cases (e.g., negative quantities, invalid inputs) improves your testing skills.
* Documentation and Commenting: Writing clear and concise comments and documentation helps others understand your code and is a good practice for collaboration and maintainability.
* Project Organization: Structuring your code into functions and modules improves readability, maintainability, and scalability of your project.