COMPUTER SCIENCE

PRACTICAL FILE

XII

Program 1: A GUI program to visualize stack concepts like push, peep, and pop.

---------------------code------------------------

import tkinter as tk

from tkinter import ttk

class App(tk.Tk):

    def \_\_init\_\_(*self*):

        super().\_\_init\_\_()

*self*.title("VisualStackCode")

*self*.geometry("450x600+500+50")

*self*.configure(*bg*="#150050", *padx*=50, *pady*=50)

*self*.stack = ("1st", "2nd", "3rd")

*self*.handy\_stack = list(*self*.stack)

*self*.config\_stack = []

*self*.stack\_canvas = tk.Canvas(*bg*="#150050", *width*=350, *height*=400)

*self*.show\_stack(375)

*self*.stack\_canvas.grid(*column*=0, *row*=0, *columnspan*=2)

*self*.push\_entry = tk.Entry(*bg*="#160040", *font*=("Dogica", 22), *highlightthickness*=0, *fg*="#0E8388", *width*=9)

*self*.push\_entry.grid(*column*=0, *row*=1, *pady*=20, *sticky*="E")

*self*.push\_butt = tk.Button(*bg*="#160040", *fg*="#FB2576", *text*="PUSH", *font*=("Dogica", 15), *activeforeground*="#03C988", *activebackground*="#160040", *pady*=2, *command*=*self*.push)

*self*.push\_butt.grid(*column*=1, *row*=1, *sticky*="W")

*self*.peep\_butt = tk.Button(*bg*="#160040", *fg*="#FB2576", *text*="PEEP", *font*=("Dogica", 15), *activeforeground*="#03C988", *activebackground*="#160040", *width*=7, *command*=*self*.peep)

*self*.peep\_butt.grid(*column*=0, *row*=2, *sticky*="E")

*self*.pop\_butt = tk.Button(*bg*="#160040", *fg*="#FB2576", *text*="POP", *font*=("Dogica", 15), *activeforeground*="#03C988", *activebackground*="#160040", *width*=7, *command*=*self*.poop)

*self*.pop\_butt.grid(*column*=0, *row*=3, *sticky*="E")

    def show\_stack(*self*, *y*):

        if len(*self*.handy\_stack) != 0:

*self*.txt = *self*.stack\_canvas.create\_text(175, *y*, *fill*="#FB2576", *activefill*="#03C988", *text*=(*self*.handy\_stack[0]), *font*=("Dogica", 25))

*self*.config\_stack.append(*self*.txt)

*self*.handy\_stack.pop(0)

*y* -= 50

*self*.show\_stack(*y*)

    def peep(*self*):

*self*.stack\_canvas.itemconfig(*self*.config\_stack[-1], *fill*="#FF0000", *activefill*="#0E8388")

    def push(*self*):

        if len(*self*.stack) <8:

            push\_txt = *self*.push\_entry.get()

            if push\_txt != "":

*self*.handy\_stack = list(*self*.stack)

*self*.handy\_stack.append(push\_txt)

*self*.stack = tuple(*self*.handy\_stack)

*self*.stack\_canvas.delete("all")

*self*.config\_stack = []

*self*.show\_stack(375)

        else:

*self*.stack\_canvas.create\_text(331, 375, *fill*="#FF0000", *text*="W", *font*=("Dogica", 25))

*self*.stack\_canvas.create\_text(326, 325, *fill*="#FF0000", *text*="O", *font*=("Dogica", 25))

*self*.stack\_canvas.create\_text(330, 275, *fill*="#FF0000", *text*="L", *font*=("Dogica", 25))

*self*.stack\_canvas.create\_text(330, 225, *fill*="#FF0000", *text*="F", *font*=("Dogica", 25))

*self*.stack\_canvas.create\_text(330, 175, *fill*="#FF0000", *text*="R", *font*=("Dogica", 25))

*self*.stack\_canvas.create\_text(330, 125, *fill*="#FF0000", *text*="E", *font*=("Dogica", 25))

*self*.stack\_canvas.create\_text(330, 75, *fill*="#FF0000", *text*="V", *font*=("Dogica", 25))

*self*.stack\_canvas.create\_text(326, 25, *fill*="#FF0000", *text*="O", *font*=("Dogica", 25))

    def poop(*self*):

*self*.handy\_stack = list(*self*.stack)

        if len(*self*.handy\_stack) != 0:

            try:

*self*.handy\_stack.pop()

            except IndexError:

                return None

            else:

*self*.stack\_canvas.delete("all")

*self*.stack = tuple(*self*.handy\_stack)

*self*.show\_stack(375)

        else:

*self*.stack\_canvas.create\_text(334, 365, *fill*="#FF0000", *text*="W", *font*=("Dogica", 25))

*self*.stack\_canvas.create\_text(330, 325, *fill*="#FF0000", *text*="O", *font*=("Dogica", 25))

*self*.stack\_canvas.create\_text(330, 285, *fill*="#FF0000", *text*="L", *font*=("Dogica", 25))

*self*.stack\_canvas.create\_text(330, 245, *fill*="#FF0000", *text*="F", *font*=("Dogica", 25))

*self*.stack\_canvas.create\_text(330, 205, *fill*="#FF0000", *text*="R", *font*=("Dogica", 25))

*self*.stack\_canvas.create\_text(330, 165, *fill*="#FF0000", *text*="E", *font*=("Dogica", 25))

*self*.stack\_canvas.create\_text(330, 125, *fill*="#FF0000", *text*="D", *font*=("Dogica", 25))

*self*.stack\_canvas.create\_text(330, 85, *fill*="#FF0000", *text*="N", *font*=("Dogica", 25))

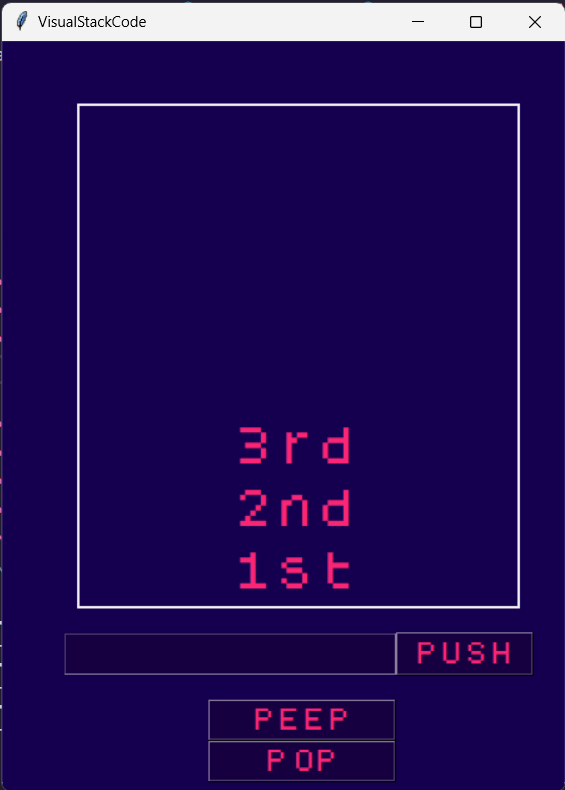
*self*.stack\_canvas.create\_text(330, 45, *fill*="#FF0000", *text*="U", *font*=("Dogica", 25))

if \_\_name\_\_=="\_\_main\_\_":

    app = App()

    app.mainloop()

----------------------output---------------------



Program 2: A text file handler with functions to read file, append to file, count characters in file, and update file.

---------------------code------------------------

class Menu():

    def \_\_init\_\_(*self*):

*self*.file\_path = input("File Path :: ")

    def read\_file(*self*):

        with open(*self*.file\_path, "r") as data:

            print(data.read())

    def add\_to\_file(*self*):

        with open(*self*.file\_path, "a") as data:

            to\_add = (f'\n{input(r"What to add [/q to stop writing]:: ")}')

            while not to\_add.endswith("/q"):

                to\_add += (f'\n{input("next line :: ")}')

            data.write(to\_add.removesuffix("/q"))

            print("FILE SUCCESSFULLY UPDATED!")

    def count\_char(*self*):

        to\_count = input("What to count :: ")

        with open(*self*.file\_path, "r") as data:

            file\_data = data.read()

            counter = 0

            for char in file\_data:

                if char == to\_count:

                    counter += 1

        print(f'no. of "{to\_count}" = {counter}')

    def swap\_stuff(*self*):

        to\_swap = input("What to swap :: ")

        swap\_with = input("Swap with :: ")

        with open(*self*.file\_path, "r") as data:

            file\_data = data.read()

        if to\_swap in file\_data:

            new\_data = file\_data.replace(to\_swap, swap\_with)

            with open(*self*.file\_path, "w") as data:

                data.write(new\_data)

            print(f"FILE SUCCESSFULLY UPDATED!")

        else:

            print(f'ERROR :: "{to\_swap}" doesn\'t exist in the file!')

if \_\_name\_\_ == "\_\_main\_\_":

    menu = Menu()

    print("~WELCOME TO TEXT FILE HANDLER~")

    available\_fns = {

        "1": "read\_file",

        "2": "add\_to\_file",

        "3": "count\_char",

        "4": "swap\_stuff",

    }

    print(f"~AVAILABLE FUNCTIONS~\n\t{available\_fns}")

    while True:

        wanna = input("What wanna do user? :: ")

        if wanna == "exit":

            break

        elif wanna in ["fns", "show fns", "available fns", "get\_fns"]:

            print(f"~AVAILABLE FUNCTIONS~\n\t{available\_fns}")

        elif (fnc:=available\_fns.get(wanna, "null"))!="null":

            menu.\_\_getattribute\_\_(fnc)()

        else:

            print("~INVALID INPUT~")

----------------------output---------------------



Program 3: A CLI coffee machine program.

---------------------code------------------------

menu = {

    "espresso": {

        "ingredients": {

            "water": 50,

            "milk": 0,

            "coffee": 18,

        },

        "cost": 1.5,

    },

    "latte": {

        "ingredients": {

            "water": 200,

            "milk": 150,

            "coffee": 24,

        },

        "cost": 2.5,

    },

    "cappuccino": {

        "ingredients": {

            "water": 250,

            "milk": 100,

            "coffee": 24,

        },

        "cost": 3.0,

    }

}

resources = {

    "water": 500,

    "milk": 400,

    "coffee": 250,

    "money" : 0

}

def check\_resources(*water*, *milk*, *coffee*):

*"""Checks if enough resources are available for making coffee"""*

    if resources["water"] >= *water* and resources["milk"] >= *milk* and resources["coffee"] >= *coffee*:

        return True

    else:

        return False

def recieve\_money(*coffee*):

*"""Recieves coins and returns actual amount recieved"""*

    print(f"That will be ${menu[*coffee*]['cost']}.\nPlease insert coins.")

    quaters = int(input("how many quaters? :"))

    dimes = int(input("how many dimes? :"))

    nickles = int(input("how many nickles? :"))

    pennies = int(input("how many pennies? :"))

    money\_received = (quaters \* 0.25) + (dimes \* 0.10) + (nickles \* 0.05) + (pennies \* 0.01)

    return money\_received

def update\_resources(*coffee*, *amount\_received*):

*"""Checks if money recieved is enough, refunds if it is excess and updates all variables"""*

    cost = menu[*coffee*]["cost"]

    if cost == *amount\_received*:

        print(f"Here is your {*coffee*}. Enjoy!")

        resources["water"] -= menu[*coffee*]["ingredients"]["water"]

        resources["milk"] -= menu[*coffee*]["ingredients"]["milk"]

        resources["coffee"] -= menu[*coffee*]["ingredients"]["coffee"]

        resources["money"] += cost

    elif cost < *amount\_received*:

        money\_returned = *amount\_received* - cost

        print(f"Here is ${round(money\_returned, 2)} in change.\nAnd here is your {*coffee*}. Enjoy!")

        resources["water"] -= menu[*coffee*]["ingredients"]["water"]

        resources["milk"] -= menu[*coffee*]["ingredients"]["milk"]

        resources["coffee"] -= menu[*coffee*]["ingredients"]["coffee"]

        resources["money"] += cost

        return True

    elif cost > *amount\_received*:

        print("Sorry that's not enough money. Money Refunded.")

        return False

def whats\_short(*water*, *milk*, *coffee*,):

*"""Checks what is not sufficient"""*

    if *water* > resources["water"]:

        print("Error 69: Sorry there is not enough water. Money Refunded")

    elif *milk* > resources["milk"]:

        print("Error 99: Sorry there is not enough milk. Money Refunded")

    elif *coffee* > resources["coffee"]:

        print("Error 09: Sorry there is not enough coffee. Money Refunded")

while True:

    choice = input("What would you like to have? [espresso/latte/cappuccino]: ").lower()

    if choice == "off":

        break

    elif choice == "report":

        print(f"Water: {resources['water']}\nMilk: {resources['milk']}\nCoffee: {resources['coffee']}g\nMoney: ${round(resources['money'], 2)}")

    else:

        if choice == "espresso":

            if check\_resources(menu["espresso"]["ingredients"]["water"], menu["espresso"]["ingredients"]["milk"], menu["espresso"]["ingredients"]["coffee"]):

                amount = recieve\_money(choice)

                update\_resources(choice, amount)

            else:

                whats\_short(menu["espresso"]["ingredients"]["water"], menu["espresso"]["ingredients"]["milk"], menu["espresso"]["ingredients"]["coffee"])

        elif choice == "latte":

            if check\_resources(menu["latte"]["ingredients"]["water"], menu["latte"]["ingredients"]["milk"], menu["latte"]["ingredients"]["coffee"]):

                amount = recieve\_money(choice)

                update\_resources(choice, amount)

            else:

                whats\_short(menu["latte"]["ingredients"]["water"], menu["latte"]["ingredients"]["milk"], menu["latte"]["ingredients"]["coffee"])

        elif choice == "cappuccino":

            if check\_resources(menu["cappuccino"]["ingredients"]["water"], menu["cappuccino"]["ingredients"]["milk"], menu["cappuccino"]["ingredients"]["coffee"]):

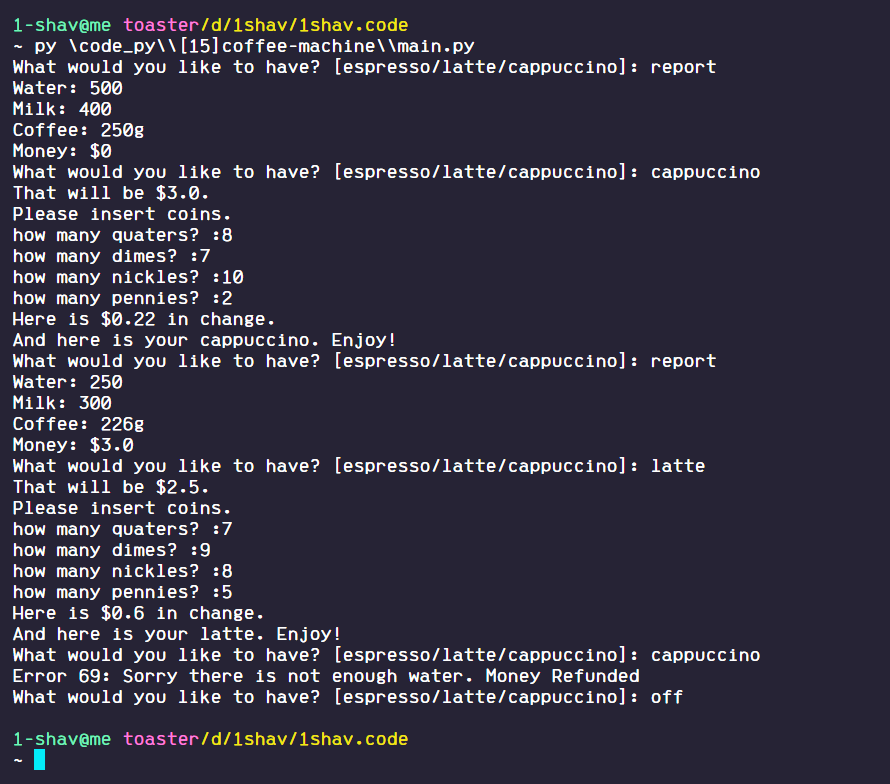
                amount = recieve\_money(choice)

                update\_resources(choice, amount)

            else:

                whats\_short(menu["cappuccino"]["ingredients"]["water"], menu["cappuccino"]["ingredients"]["milk"], menu["cappuccino"]["ingredients"]["coffee"])

----------------------output---------------------



Program 3: