COMPUTER SCIENCE

PRACTICAL FILE

XII

~vanshav bhalla

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))

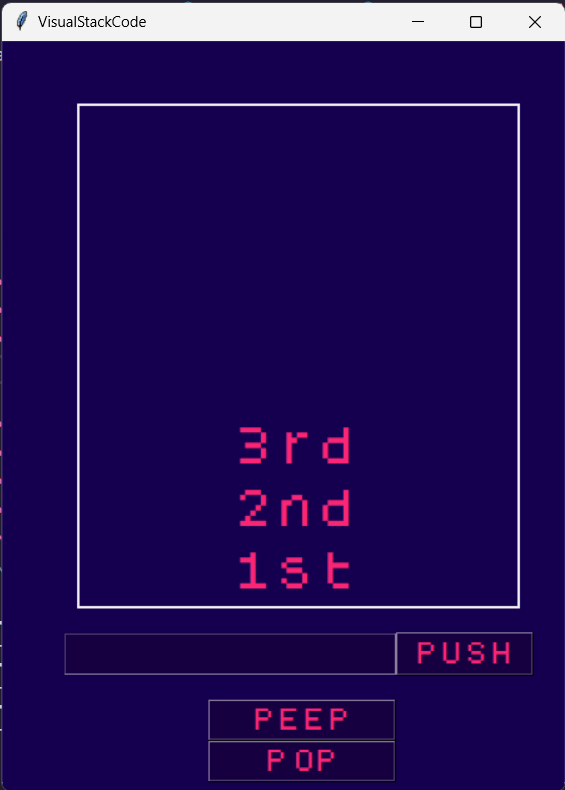
*self*.stack\_canvas.create\_text(330, 85, *fill*="#FF0000", *text*="N", *font*=("Dogica", 25))

*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":

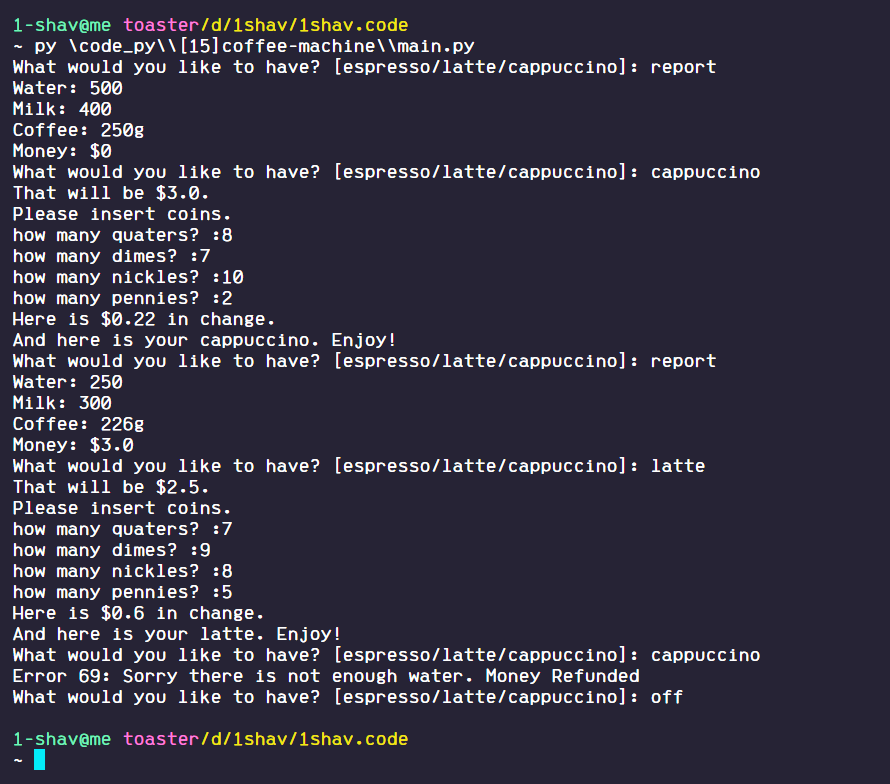
            if check\_resources(menu["cappuccino"]["ingredients"]["water"], menu["cappuccino"]["ingredients"]["milk"], menu["cappuccino"]["ingredients"]["coffee"]):

                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: A program that converts English string into a list of NATO Phonetic Alphabets by reading a csv of all letters and their code.

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

import pandas

data = pandas.read\_csv("phonetic\_alphabet.csv")

data\_dicti = {row.letter:row.code for (index, row) in data.iterrows()}

while True:

    try:

        stringii = input("Please input the string you want code for: ")

        resultant = [data\_dicti[letter] for letter in stringii.upper()]

    except KeyError:

        print("Please input only english letters!")

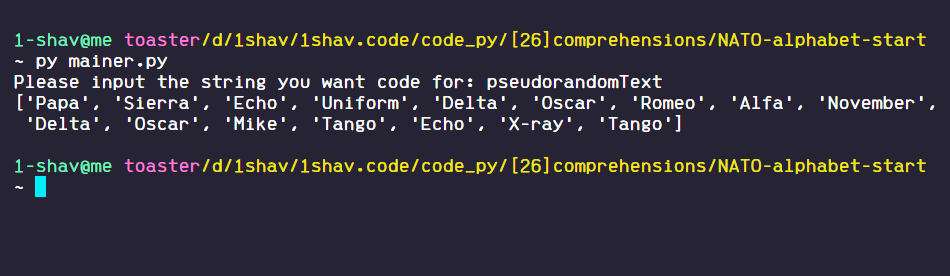
        continue

    else:

        print(resultant)

        break

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



Program 4: A GUI program to convert miles into kilometers.

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

import tkinter as tk

khidki = tk.Tk()

khidki.title("Miles To Km")

khidki.minsize(*width*=300, *height*=150)

khidki.config(*padx*=50, *pady*=35)

inputational = tk.Entry()

inputational.insert(tk.END, "0")

inputational.config(*width*=10)

inputational.grid(*row*=0, *column*=1)

miles\_morales = tk.Label(*text*="Miles", *padx*=7, *pady*=5)

miles\_morales.grid(*row*=0, *column*=2)

labhel = tk.Label(*text*="is equal to", *padx*=7, *pady*=2)

labhel.grid(*row*=1, *column*=0)

km\_morales = tk.Label(*text*="0", *padx*=5, *pady*=2)

km\_morales.grid(*row*=1, *column*=1)

labhel2 = tk.Label(*text*="Km", *padx*=5, *pady*=2)

labhel2.grid(*row*=1, *column*=2)

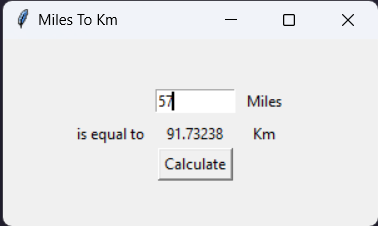
def convert():

    km\_morales.config(*text*=f"{int(inputational.get()) \* 1.60934}")

khatka = tk.Button(*text*="Calculate", *command*=convert)

khatka.grid(*row*=2, *column*=1)

khidki.mainloop()

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

Program 5: A GUI pomodoro timer app that helps increase productivity.

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

import tkinter as tk

import math

from playsound import playsound

*# ---------------------------- CONSTANTS ------------------------------- #*

PINK = "#F2C9F0"

PURPLE = "#395D73"

GREEN = "#2D4859"

YELLOW = "#f7f5dd"

BLUE = "#1B1959"

FONT\_NAME = "Dogica"

WORK\_MIN = 35

SHORT\_BREAK\_MIN = 5

LONG\_BREAK\_MIN = 20

REPS = 0

CHECKS = 0

TIMER = None

STARTED = False

STOPPED = False

TIMER\_RN = 0

*# ---------------------------- TIMER RESET ------------------------------#*

def reset\_timer():

    global REPS

    global CHECKS

    global STARTED

    global STOPPED

    screen.after\_cancel(TIMER)

    REPS = 0

    CHECKS = 0

    config\_check\_marks()

    timed\_canvas.itemconfig(timer\_txt, *text*="00:00")

    STARTED = False

    start.config(*text*="start", *command*=check\_started)

    STOPPED = False

*# ---------------------------- TIMER MECHANISM ------------------------- #*

def check\_started():

    global STARTED

    global STOPPED

    if STOPPED == True:

        STOPPED = False

        restart\_timer()

        start.config(*text*="stop", *command*=stop\_timer)

    if STARTED == False:

        STARTED = True

        start\_timer()

        start.config(*text*="stop", *command*=stop\_timer)

def start\_timer():

    global REPS

    REPS += 1

    work\_sec = WORK\_MIN \* 60

    short\_break\_sec = SHORT\_BREAK\_MIN \* 60

    long\_break\_sec = LONG\_BREAK\_MIN \* 60

    if REPS % 8 == 0:

        count\_down(long\_break\_sec)

    elif REPS % 2 == 0:

        count\_down(short\_break\_sec)

    else:

        count\_down(work\_sec)

def restart\_timer():

    count\_down(TIMER\_RN)

def stop\_timer():

    global STOPPED

    STOPPED = True

    start.config(*text*="start", *command*=check\_started)

    screen.after\_cancel(TIMER)

*# ---------------------------- COUNTDOWN MECHANISM --------------------- #*

def count\_down(*count*):

    global CHECKS

    global TIMER

    global TIMER\_RN

    count\_min = math.floor(*count* / 60)

    formatted\_min = "{:02d}".format(count\_min)

    count\_sec = int(*count* % 60)

    formatted\_sec = "{:02d}".format(count\_sec)

    timed\_canvas.itemconfig(timer\_txt, *text*=f"{formatted\_min}:{formatted\_sec}")

    TIMER\_RN = *count*

    if *count* > 0:

        TIMER = screen.after(1000, count\_down, *count* - 1)

    else:

        playsound("code\_py/[28]more\_\_gui/pawmodoro/sound.mp3")

        start\_timer()

        if REPS % 2 == 0:

            CHECKS += 1

            config\_check\_marks()

def config\_check\_marks():

        check\_marks.config(*text*="-" \* CHECKS)

*# ---------------------------- UI SETUP ------------------------------- #*

screen = tk.Tk()

screen.title("Pawmodoro")

screen.config(*padx*=100, *pady*=75, *bg*=PINK)

logo\_canvas = tk.Canvas(*width*=460, *height*=300, *bg*=PINK, *highlightthickness*=0)

logo = tk.PhotoImage(*file*="code\_py/[28]more\_\_gui/pawmodoro/logooo.png")

logo\_canvas.create\_image(230, 150, *image*=logo)

logo\_canvas.grid(*row*=0, *column*=1)

timed\_canvas = tk.Canvas(*width*=260, *height*=50, *bg*=PINK, *highlightthickness*=0)

timer\_txt = timed\_canvas.create\_text(130, 25, *text*="00:00", *font*=(FONT\_NAME, 40, "bold"), *fill*="white")

timed\_canvas.grid(*row*=1, *column*=1)

start = tk.Button(*text*="start", *command*=check\_started, *font*=(FONT\_NAME, 10), *fg*=YELLOW, *bg*=PURPLE)

start.grid(*row*=2, *column*=0)

reset = tk.Button(*text*="reset", *command*=reset\_timer, *font*=(FONT\_NAME, 10), *fg*=YELLOW, *bg*=PURPLE)

reset.grid(*row*=2, *column*=2)

check\_marks = tk.Label(*font*=(FONT\_NAME, 17),*fg*=GREEN, *bg*=PINK)

check\_marks.grid(*row*=3, *column*=1)

screen.mainloop()

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



Program 6: A GUI task manager app with features to add new task, delete a task, and update a task.

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

import tkinter as tk

from tkinter import messagebox

def add\_task():

task = entry\_task.get()

if task:

listbox\_tasks.insert(tk.END, task)

entry\_task.delete(0, tk.END)

else:

messagebox.showwarning("Warning", "Please enter a task.")

def delete\_task():

try:

selected\_task = listbox\_tasks.curselection()[0]

listbox\_tasks.delete(selected\_task)

except:

messagebox.showwarning("Warning", "Please select a task to delete.")

def update\_task():

try:

selected\_task = listbox\_tasks.curselection()[0]

updated\_task = entry\_task.get()

listbox\_tasks.delete(selected\_task)

listbox\_tasks.insert(selected\_task, updated\_task)

entry\_task.delete(0, tk.END)

except:

messagebox.showwarning("Warning", "Please select a task to update.")

window = tk.Tk()

window.title("Task Manager")

window.geometry("400x400")

window.configure(bg='#54494B')

label = tk.Label(window, text="Enter task:", bg='#54494B', fg='#F1F7ED')

entry\_task = tk.Entry(window, bg='#CCDDE2', fg='#594236')

add\_button = tk.Button(window, text="Add Task", command=add\_task, bg='#91C7B1', fg='#594236')

delete\_button = tk.Button(window, text="Delete Task", command=delete\_task, bg='#B33951', fg='#F1F7ED')

update\_button = tk.Button(window, text="Update Task", command=update\_task, bg='#E3D081', fg='#594236')

listbox\_tasks = tk.Listbox(window, bg='#CCDDE2', fg='#594236')

label.pack(pady=(10, 0))

entry\_task.pack(pady=(0, 10), padx=20, fill='both')

add\_button.pack(pady=(0, 10), fill='both')

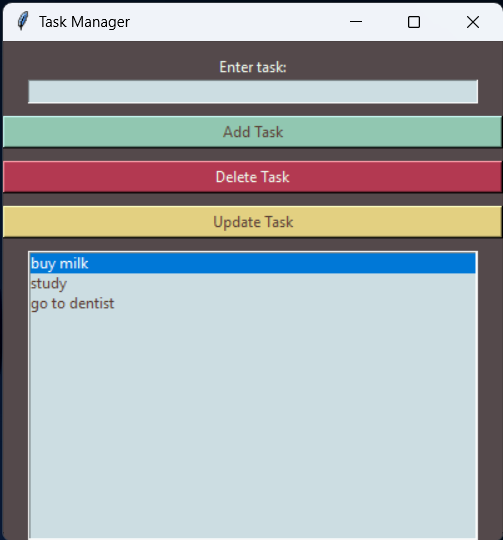
delete\_button.pack(pady=(0, 10), fill='both')

update\_button.pack(pady=(0, 10), fill='both')

listbox\_tasks.pack(padx=20, pady=0, fill='both', expand=True)

window.mainloop()

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



Program 7: A GUI flash card app that helps you learn French.

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

import tkinter as tk

from tkinter import ttk

import pandas as pd

from random import choice

*# ---------------------------------- CONSTANTS -----------------------------------#*

BACKGROUND\_COLOR = "#B1DDC6"

BLACK\_SOMETHINF = "#1B1E23"

OFF\_WHITE = "#FAF9F6"

*# ---------------------------------- UI SETUP ------------------------------------#*

class App(tk.Tk):

    def \_\_init\_\_(*self*):

        super().\_\_init\_\_()

*#self.root = tk.Tk("FlashyCards")*

*self*.configure(*bg*=BACKGROUND\_COLOR, *padx*=30, *pady*=20)

*self*.title("FlashyCards")

*self*.geometry("600x500+500+250")

*#--------------------------------------english\_frame\_related-------------#*

*self*.eng\_frame = tk.Frame(*self*)

*self*.eng\_canvas = tk.Canvas(*self*.eng\_frame, *bg*=BACKGROUND\_COLOR, *width*=540, *height*=360, *highlightthickness*=0)

*self*.card\_eng = tk.PhotoImage(*file*="code\_py/[31]capstone\_flashcard/images/card\_back\_re.png")

*self*.eng\_canvas.create\_image(270, 180, *image*=*self*.card\_eng)

*self*.eng\_canvas.create\_text(263, 99, *text*="ENGLISH", *font*=("Brass Mono", 20, "italic"), *fill*=OFF\_WHITE)

*self*.english\_text = *self*.eng\_canvas.create\_text(263, 170, *text*="word", *font*=("Ariel", 47, "bold"), *fill*=OFF\_WHITE)

*self*.eng\_canvas.pack()

*self*.eng\_frame.grid(*column*=0, *row*=0, *columnspan*=2)

*#----------------------------------french\_frame\_related------------------#*

*self*.fre\_frame = tk.Frame(*self*)

*self*.fre\_canvas = tk.Canvas(*self*.fre\_frame, *bg*=BACKGROUND\_COLOR, *width*=540, *height*=360, *highlightthickness*=0)

*self*.card\_fre = tk.PhotoImage(*file*="code\_py/[31]capstone\_flashcard/images/card\_front\_re.png")

*self*.fre\_canvas.create\_image(270, 180, *image*=*self*.card\_fre)

*self*.fre\_canvas.create\_text(263, 99, *text*="FRENCH", *font*=("Brass Mono", 20, "italic") ,*fill*=BLACK\_SOMETHINF)

*self*.french\_text = *self*.fre\_canvas.create\_text(263, 170, *text*="word", *font*=("Ariel", 47, "bold"), *fill*=BLACK\_SOMETHINF)

*self*.fre\_canvas.pack()

*self*.fre\_frame.grid(*column*=0, *row*=0, *columnspan*=2)

*#-----------------------------------button\_related-----------------------#*

*self*.right\_image = tk.PhotoImage(*file*="code\_py/[31]capstone\_flashcard/images/right\_ree.png")

*self*.wrong\_image = tk.PhotoImage(*file*="code\_py/[31]capstone\_flashcard/images/wrong\_ree.png")

*self*.right\_button = tk.Button(*image*=*self*.right\_image,*borderwidth*=0, *highlightthickness*=0, *command*=*self*.he\_knows)

*self*.right\_button.grid(*column*=0, *row*=1)

*self*.wrong\_button = tk.Button(*image*=*self*.wrong\_image, *borderwidth*=0, *highlightthickness*=0, *command*=*self*.hes\_dumb)

*self*.wrong\_button.grid(*column*=1, *row*=1)

*#------------------------------------important\_ig-----------------------#*

        try:

            with open(*file*="code\_py/[31]capstone\_flashcard/data/words\_to\_learn.csv") as data:

                data = pd.read\_csv(data)

        except FileNotFoundError:

            with open(*file*="code\_py/[31]capstone\_flashcard/data/french\_words.csv") as data:

                data = pd.read\_csv(data)

*self*.data\_pairs = data.to\_dict(*orient*="records")

*self*.current\_frame = *self*.fre\_frame

*self*.waiter = *self*.after(3000, *self*.upar\_niche)

*self*.refresh\_word()

*#---------------------------functions-------------------------------#*

    def get\_word(*self*):

*self*.current\_pair = choice(*self*.data\_pairs)

        french\_word = *self*.current\_pair["French"]

        english\_word = *self*.current\_pair["English"]

        return french\_word, english\_word

    def he\_knows(*self*):

*self*.data\_pairs.remove(*self*.current\_pair)

        updated\_data = pd.DataFrame(*self*.data\_pairs)

        with open(*file*="code\_py/[31]capstone\_flashcard/data/words\_to\_learn.csv", *mode*="w") as new\_data\_file:

            updated\_data.to\_csv(new\_data\_file, *index*=False)

*self*.refresh\_word()

    def hes\_dumb(*self*):

*self*.refresh\_word()

    def refresh\_word(*self*):

        french\_word, english\_word = *self*.get\_word()

*self*.fre\_canvas.itemconfig(*self*.french\_text, *text*=french\_word)

*self*.eng\_canvas.itemconfig(*self*.english\_text, *text*=english\_word)

*self*.next\_card()

    def next\_card(*self*):

*self*.after\_cancel(*self*.waiter)

*self*.current\_frame = *self*.fre\_frame

*self*.fre\_frame.tkraise()

*self*.waiter = *self*.after(3000, *self*.upar\_niche)

    def upar\_niche(*self*):

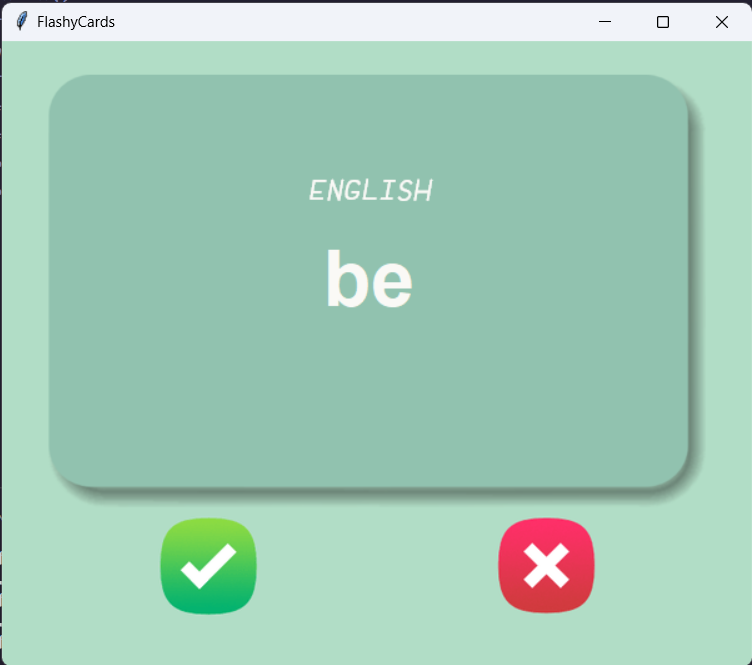
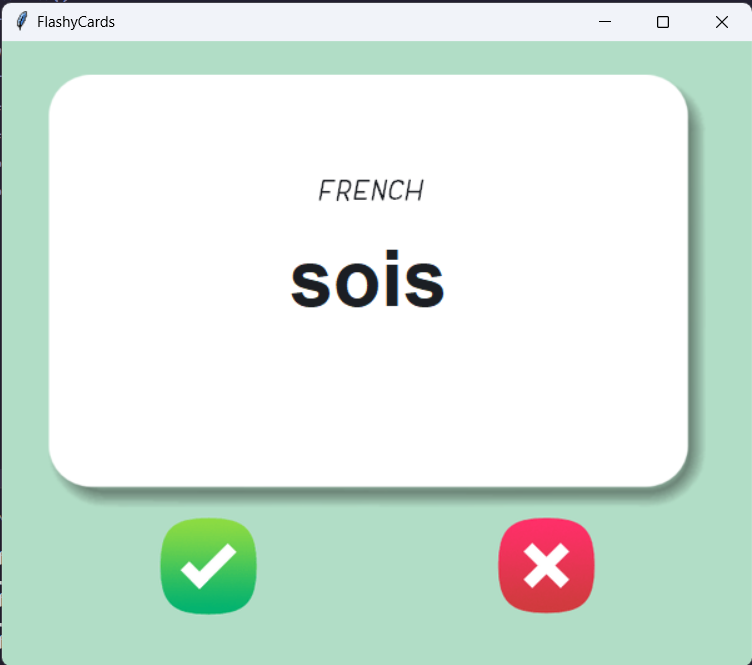
*self*.current\_frame = *self*.eng\_frame

*self*.eng\_frame.tkraise()

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

    application = App()

    application.mainloop()

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

Program 8: A rock-paper-scissor program.

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

dicti = {

    "rock":'''

    \_\_\_\_\_\_\_

---'   \_\_\_\_)

      (\_\_\_\_\_)

      (\_\_\_\_\_)

      (\_\_\_\_)

---.\_\_(\_\_\_)

''',

    "paper":'''

    \_\_\_\_\_\_\_

---'   \_\_\_\_)\_\_\_\_

          \_\_\_\_\_\_)

          \_\_\_\_\_\_\_)

         \_\_\_\_\_\_\_)

---.\_\_\_\_\_\_\_\_\_\_)

''',

    "scissor":'''

    \_\_\_\_\_\_\_

---'   \_\_\_\_)\_\_\_\_

          \_\_\_\_\_\_)

       \_\_\_\_\_\_\_\_\_\_)

      (\_\_\_\_)

---.\_\_(\_\_\_)

'''

}

choices = ["rock", "paper", "scissor"]

import random

import os

def compare(*user*, *comp*):

    if *user* == "rock":

        if *comp* == "rock":

            print("seems to be a DRAW. Try again loser!")

        elif *comp* == "paper":

            print("the roboto WON and YOU LOSE! this is a battle of pure skill mister!")

        elif *comp* == "scissor":

            print("apparently you... WON?! your luck seemed to work huh!")

    elif *user* == "paper":

        if *comp* == "rock":

            print("apparently you... WON?! your luck seemed to work huh!")

        elif *comp* == "paper":

            print("seems to be a DRAW. Try again loser!")

        elif *comp* == "scissor":

            print("the roboto WON and YOU LOSE! this is a battle of pure skill mister!")

    elif *user* == "scissor":

        if *comp* == "rock":

            print("the roboto WON and YOU LOSE! this is a battle of pure skill mister!")

        elif *comp* == "paper":

            print("apparently you... WON?! your luck seemed to work huh!")

        elif *comp* == "scissor":

            print("seems to be a DRAW. Try again loser!")

while True:

    player\_choice = input(f"User Choice :: ")

    computer\_choice = random.choice(choices)

    if player\_choice not in choices:

        print(f"Please choose a valid attack from {choices}")

        continue

    else:

        print(f"You chose :: {dicti[player\_choice]}")

        print(f"The roboto chose :: {dicti[computer\_choice]}")

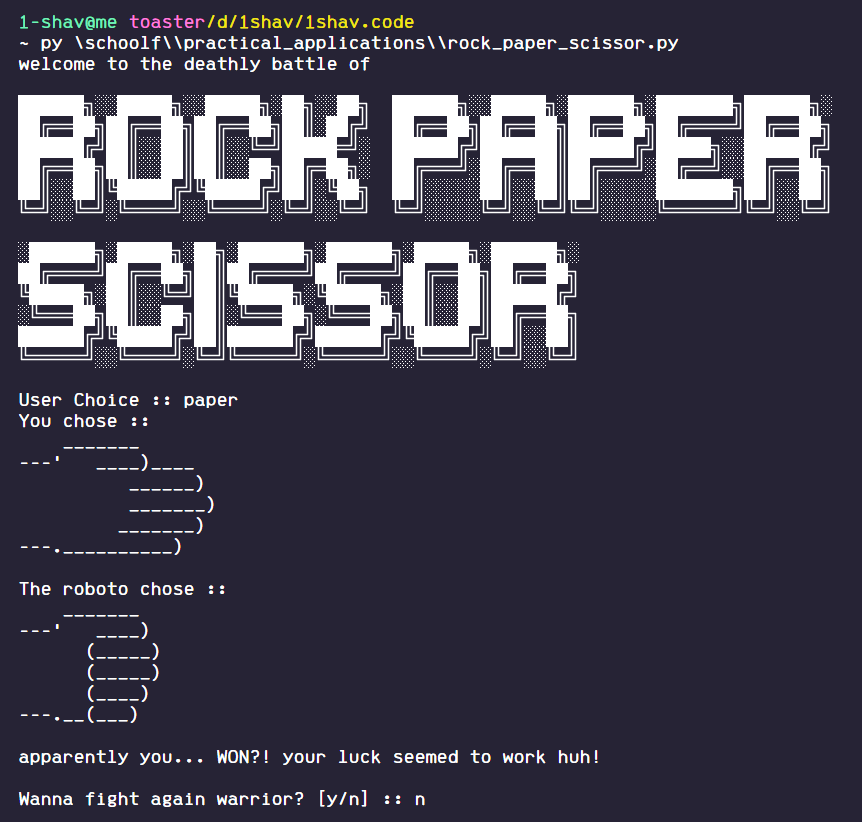
        compare(player\_choice, computer\_choice)

    if input("\nWanna fight again warrior? [y/n] :: ").lower() != "y":

        break

    else:

        os.system("cls")

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

Program 9: A hangman game program.

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

import random

from extra import words\_list, hangman\_word, hangman\_ascii

import os

def game():

    print(hangman\_word)

    choosen\_word = random.choice(words\_list)

    display = []

    for \_ in choosen\_word:

        display.append("\_")

    lives = 7

    while lives != 0 and "\_" in display:

        print(f"\n{display}")

        user\_guess = input("Please type a letter: ").lower()

        if user\_guess not in choosen\_word:

            lives -= 1

            print(f"\nThat letter is not present in the word\nlives left {lives}\n{hangman\_ascii[7 - lives]}")

            continue

        for position in range(len(display)):

            if choosen\_word[position] == user\_guess:

                display[position] = user\_guess

        print(f"\nGreat, you guessed right\nlives remaining {lives}\n{hangman\_ascii[7 - lives]}")

    if lives == 0:

        print(f"\nGAME OVER, You lost all your lives\nThe correct word was {choosen\_word}")

    else:

        print(f"\nGAME OVER, You won and correctly guessed {choosen\_word}")

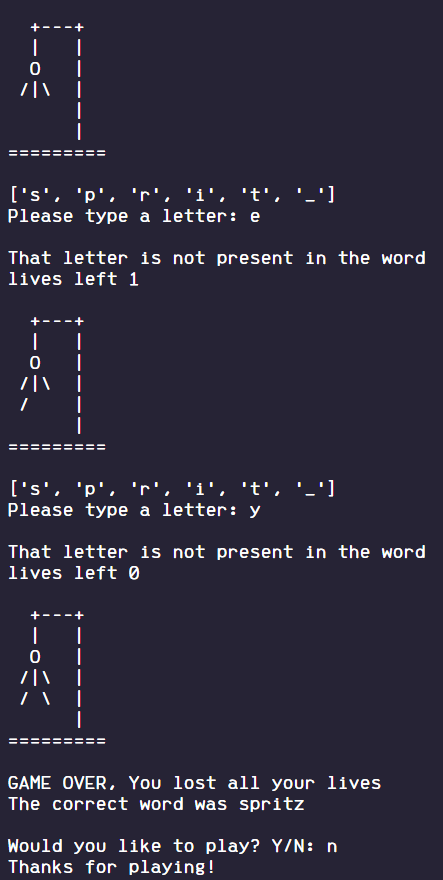
game()

while input("\nWould you like to play? Y/N: ").upper() == "Y":

    os.system("cls")

    game()

print("Thanks for playing!")

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

Program 10: A basic Caesar cipher encoder / decoder program based on shift-based substitution.

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

alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

def caeser(*data\_tuple*):

    caeser\_direction, input\_text, shift\_amount = *data\_tuple*[0], *data\_tuple*[1], *data\_tuple*[2]

    output = ""

    if caeser\_direction == "decode":

        shift\_amount \*= -1

    for char in input\_text:

        if char in alphabet:

            position = alphabet.index(char)

            new\_position = position + shift\_amount

            output += alphabet[new\_position]

        else:

            output += char

    print(f'The {caeser\_direction}d text is "{output}"')

def gechyodata():

    direction = input("Type 'encode' to encrypt, type 'decode' to decrypt :: ").lower()

    while direction not in ["encode", "decode"]:

        print("Please only type encode/decode!")

        direction = input("Type 'encode' to encrypt, type 'decode' to decrypt :: ").lower()

    text = input("Type your message :: ").lower()

    shift = int(input("Type the shift number :: "))

    shift %= 26

    return direction, text, shift

caeser(gechyodata())

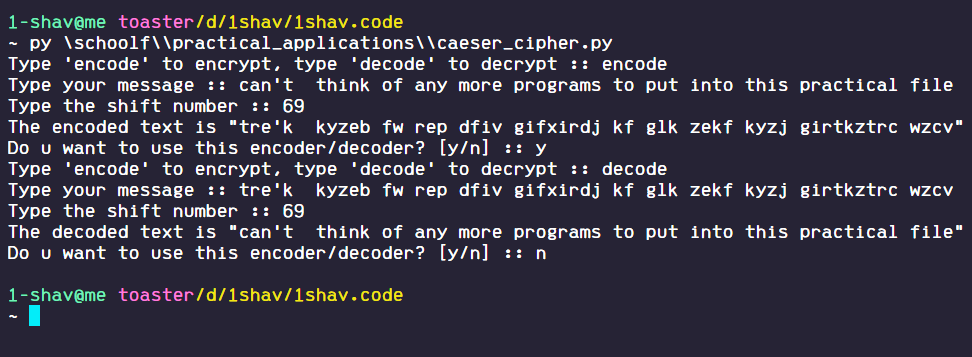
while True:

    play = input("Do u want to use this encoder/decoder? [y/n] :: ").lower()

    if play != "y":

        break

    caeser(gechyodata())

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

Program 11: A python script that tells informs the user if it is going to rain today or not.

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

import requests

url = "http://api.weatherapi.com/v1/forecast.json"

parameters = {"q":(28.59,77.08),

              "key":”API\_KEY”

              }

response = requests.get(url, *params*=parameters)

response.raise\_for\_status()

data = response.json()

willRain = data["forecast"]["forecastday"][0]["day"]["daily\_will\_it\_rain"]

willSnow = data["forecast"]["forecastday"][0]["day"]["daily\_will\_it\_snow"]

if willRain:

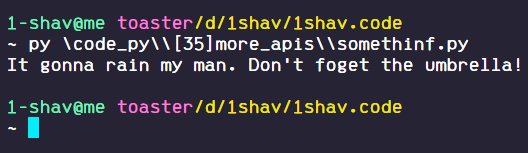
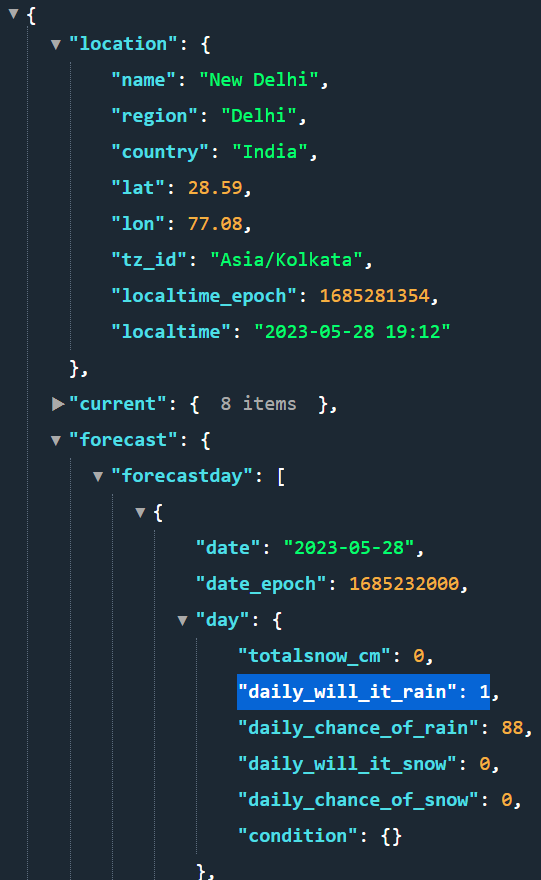
    print("It gonna rain my man. Don't foget the umbrella!")

elif willSnow:

    print("It gonna snow my man. Don't forget the umbrella!")

else:

    print(f"It ain't gonna rain. No need for the umbrella!")

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

Program 12: A GUI program that displays random kanye quotes.

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

from tkinter import \*

import requests

def get\_quote():

    url = "https://api.kanye.rest"

    response = requests.get(*url*=url)

    response.raise\_for\_status()

    data = response.json()

    canvas.itemconfig(quote\_text, *text*=data["quote"])

window = Tk()

window.title("Kanye Says...")

window.config(*padx*=50, *pady*=50)

canvas = Canvas(*width*=300, *height*=414)

background\_img = PhotoImage(*file*="background.png")

canvas.create\_image(150, 207, *image*=background\_img)

quote\_text = canvas.create\_text(150, 207, *text*="Kanye Quote Goes HERE", *width*=250, *font*=("Arial", 30, "bold"), *fill*="white")

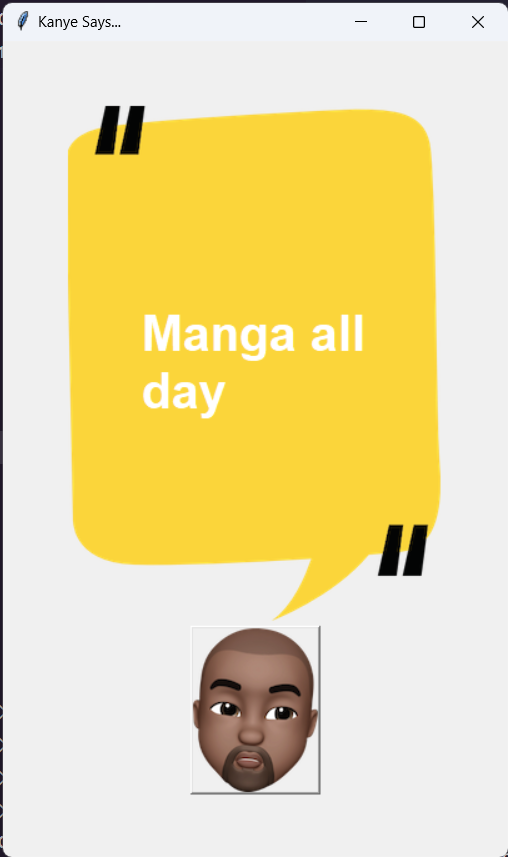
canvas.grid(*row*=0, *column*=0)

kanye\_img = PhotoImage(*file*="kanye.png")

kanye\_button = Button(*image*=kanye\_img, *highlightthickness*=0, *command*=get\_quote)

kanye\_button.grid(*row*=1, *column*=0)

window.mainloop()

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

Program 13: A simple calculator program.

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

def add(*n1*, *n2*):

    return *n1* + *n2*

def subtract(*n1*, *n2*):

    return *n1* - *n2*

def multiply(*n1*, *n2*):

    return *n1* \* *n2*

def divide(*n1*, *n2*):

    return *n1* / *n2*

operations = {

    "+": add,

    "-": subtract,

    "\*": multiply,

    "/": divide

}

def ganana():

*"""The actual calculator function."""*

    num1 = float(input("Please enter the first number:: "))

    joined = ", ".join(operations.keys())

    while True:

        operation\_symbol = input(f"Choose an operator from the following [{joined}] :: ")

        num2 = float(input("Please enter the next number :: "))

        function = operations[operation\_symbol]

        answer = function(num1, num2)

        print(f"{num1} {operation\_symbol} {num2} = {answer}")

        continu = input(f"Choice [y->continue calculating with {answer} / n->calculate somethinf else / exit->do I really have to tell you what's gonna happen?] :: ").lower()

        if continu == "y":

            num1 = answer

        elif continu == "n":

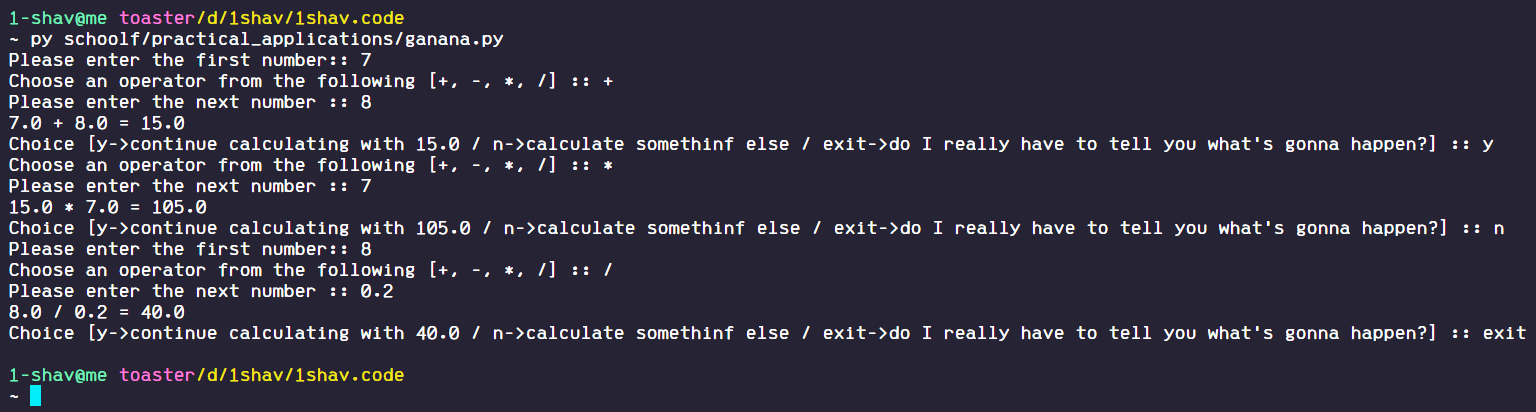
            ganana()

            break

        else:

            break

ganana()

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

Program 14: Tic-Tac-Toe implemented in python.

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

import random

import os

def display\_board(*board*):

    os.system('cls')

    print(" "+*board*[0]+' | '+*board*[1]+' | '+*board*[2])

    print("\_\_\_ "\*3)

    print("\n "+*board*[3]+' | '+*board*[4]+' | '+*board*[5])

    print("\_\_\_ "\*3)

    print("\n "+*board*[6]+' | '+*board*[7]+' | '+*board*[8])

def player\_input():

    marker = ''

    while not (marker == 'X' or marker == 'O'):

        marker = input('Player 1\n\tDo you want to be X or O? :: ').upper()

    if marker == 'X':

        return ('X', 'O')

    elif marker == "O":

        return ('O', 'X')

def place\_marker(*board*, *marker*, *position*):

*board*[*position* - 1] = *marker*

def win\_check(*board*, *mark*):

    return ((*board*[0] == *mark* and *board*[1] == *mark* and *board*[2] == *mark*) or

            (*board*[3] == *mark* and *board*[4] == *mark* and *board*[5] == *mark*) or

            (*board*[6] == *mark* and *board*[7] == *mark* and *board*[8] == *mark*) or

            (*board*[0] == *mark* and *board*[3] == *mark* and *board*[6] == *mark*) or

            (*board*[1] == *mark* and *board*[4] == *mark* and *board*[7] == *mark*) or

            (*board*[2] == *mark* and *board*[5] == *mark* and *board*[8] == *mark*) or

            (*board*[0] == *mark* and *board*[4] == *mark* and *board*[8] == *mark*) or

            (*board*[2] == *mark* and *board*[4] == *mark* and *board*[6] == *mark*))

def is\_board\_full(*board*):

    return ' ' not in *board*

def game():

    board = [' ' for \_ in range(9)]

    display\_board(board)

    player1, player2 = player\_input()

    turn = random.randint(1,2)

    if turn == 1:

        print(f'Player 1 [{player1}] goes first')

    else:

        print(f'Player 2 [{player2}] goes first')

    while True:

        if not is\_board\_full(board):

            if turn == 1:

                pos = int(input(f'Player 1 [{player1}]\n\tChoose a position :: '))

                while pos not in range(1,10) or board[pos - 1] != " ":

                    print("INVALID POSITION!")

                    pos = int(input(f'Player 1 [{player1}]\n\tChoose a position :: '))

                place\_marker(board, player1, pos)

                display\_board(board)

                if win\_check(board, player1):

                    print(f'Player 1 [{player1}] wins!')

                    break

                turn = 2

            else:

                pos = int(input(f'Player 2 [{player2}]\n\tChoose a position  :: '))

                while pos not in range(1,10) or board[pos - 1] != " ":

                    print("INVALID POSITION!")

                    pos = int(input(f'Player 2 [{player2}]\n\tChoose a position :: '))

                place\_marker(board, player2, pos)

                display\_board(board)

                if win\_check(board, player2):

                    print(f'Player 2 [{player2}] wins!')

                    break

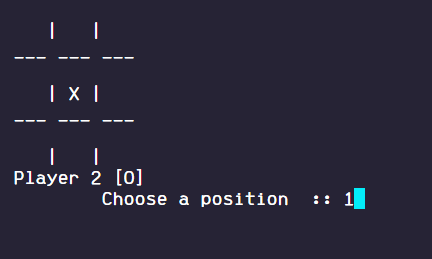
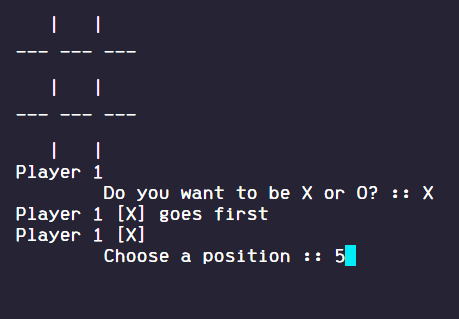
                turn = 1

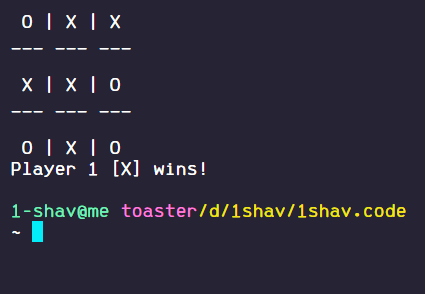
        else:

            print("IT'S A DRAW!")

            break

game()

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



Program 15: A higher-lower game that prompts user to guess which account has more followers on Instagram.

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

from art import logo, vs

from game\_data import data

from random import choice

from os import system

score = 0

chosen\_A = choice(data)

while True:

    print(f"{logo}\nCurrent score: {score}")

    chosen\_B = choice(data)

    print(f"Compare A: {chosen\_A['name']}, a {chosen\_A['description']}, from {chosen\_A['country']}")

    print(vs)

    print(f"Against B: {chosen\_B['name']}, a {chosen\_B['description']}, from {chosen\_B['country']}")

    answer = input("\nWho has more followers? Type 'A' or 'B': ").lower()

    def real\_winner(*a*, *b*):

        if int(*a*["follower\_count"]) > int(*b*["follower\_count"]):

            return "a"

        else:

            return "b"

    if answer == real\_winner(chosen\_A,  chosen\_B):

        score += 1

        chosen\_A = chosen\_B

        system("cls")

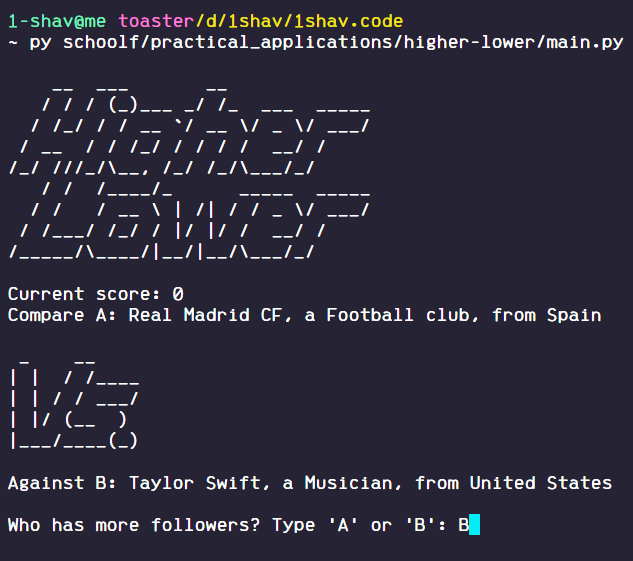
    else:

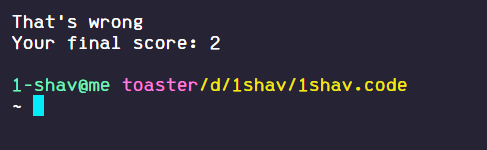
        system("cls")

        print(f"That's wrong\nYour final score: {score}")

        break

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





Program 16: MySQL program in python to create database, use it and create a table.

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

import mysql.connector as mysql

connection = mysql.connect(host="localhost", user="root", password="the\_pass")

print("CONNECTION ESTABLISHED SUCCESSFULLY")

cursor = connection.cursor()

cursor.execute("CREATE DATABASE IF NOT EXISTS school")

print("DATABASE CREATED SUCCESSFULLY")

cursor.execute("USE school")

print("DATABASE SELECTED SUCCESSFULLY")

cursor.execute("CREATE TABLE IF NOT EXISTS VIII\_B(name VARCHAR(20), maths INT(3), science INT(3), english INT(3), AI INT(3), total FLOAT(3))")

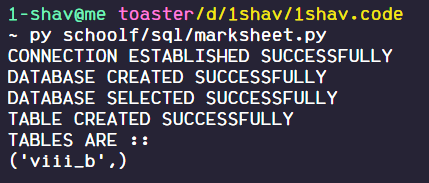
print("TABLE CREATED SUCCESSFULLY")

cursor.execute("SHOW TABLES")

print("TABLES ARE :: ")

for i in cursor:

print(i)

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

Program 17: MySQL program in python to add entries to the table created.

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

while wanna:=input("DO YOU WANT TO ENTER MARKS OF STUDENT :: ").lower():

if wanna != "y":

break

name = input("ENTER THE NAME OF STUDENT :: ")

maths = int(input("ENTER THE MARKS OF MATHS :: "))

science = int(input("ENTER THE MARKS OF SCIENCE :: "))

english = int(input("ENTER THE MARKS OF ENGLISH :: "))

AI = int(input("ENTER THE MARKS OF AI :: "))

total = maths + science + english + AI

cursor.execute(f"INSERT INTO VIII\_B(name, maths, science, english, AI, total) VALUES('{name}',{maths},{science},{english},{AI},{total})")

cursor.execute("SELECT \* FROM VIII\_B")

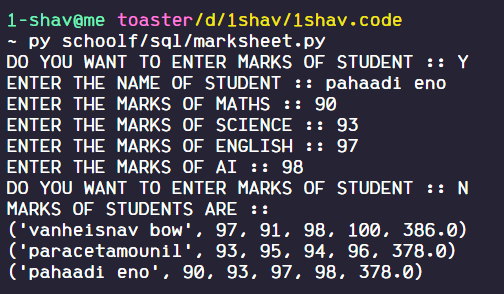
print("MARKS OF STUDENTS ARE :: ")

table\_entries = cursor.fetchall()

for i in table\_entries:

print(i)

connection.commit()

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

Program 18: MySQL program in python to alter a table by adding a column.

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

cursor.execute("ALTER TABLE viii\_b ADD roll\_num INT(3) PRIMARY KEY AUTO\_INCREMENT")

cursor.execute("SELECT \* FROM VIII\_B")

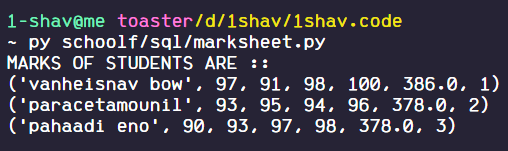
print("MARKS OF STUDENTS ARE :: ")

table\_entries = cursor.fetchall()

for i in table\_entries:

print(i)

connection.commit()

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

Program 19: A todo list application made with MySQL in Python

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

import mysql.connector as mysql

conn = mysql.connect(host="localhost", user="root", password="1-177155-400")

cursor = conn.cursor()

cursor.execute("USE school")

cursor.execute("CREATE TABLE IF NOT EXISTS todo (id INT AUTO\_INCREMENT PRIMARY KEY, task VARCHAR(255))")

def add\_task(task):

query = "INSERT INTO todo (task) VALUES (%s)"

cursor.execute(query, (task,))

conn.commit()

def view\_tasks():

cursor.execute("SELECT \* FROM todo")

tasks = cursor.fetchall()

if tasks:

for task in tasks:

print(f"Task ID: {task[0]}, Task: {task[1]}")

else:

print("No tasks found.")

def update\_task(task\_id, new\_task):

query = "UPDATE todo SET task = %s WHERE id = %s"

cursor.execute(query, (new\_task, task\_id))

conn.commit()

def delete\_task(task\_id):

query = "DELETE FROM todo WHERE id = %s"

cursor.execute(query, (task\_id,))

conn.commit()

while True:

print("\nTodo List Options:")

print("1. Add Task")

print("2. View Tasks")

print("3. Update Task")

print("4. Delete Task")

print("5. Quit")

choice = input("Enter your choice: ")

if choice == '1':

task = input("Enter the task: ")

add\_task(task)

print("Task added!")

elif choice == '2':

print("Tasks:")

view\_tasks()

elif choice == '3':

task\_id = int(input("Enter the Task ID to update: "))

new\_task = input("Enter the new task: ")

update\_task(task\_id, new\_task)

print("Task updated!")

elif choice == '4':

task\_id = int(input("Enter the Task ID to delete: "))

delete\_task(task\_id)

print("Task deleted!")

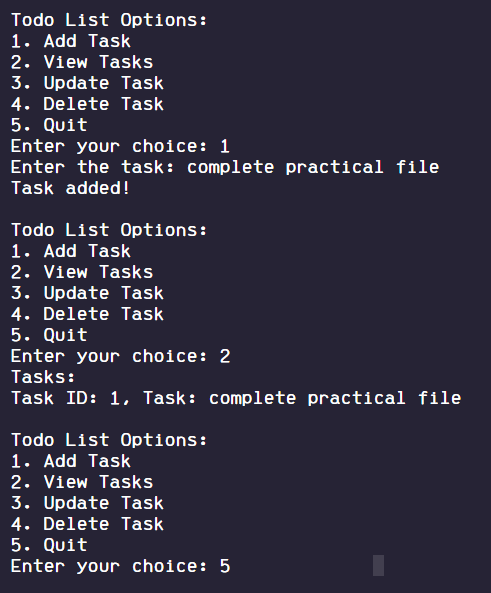
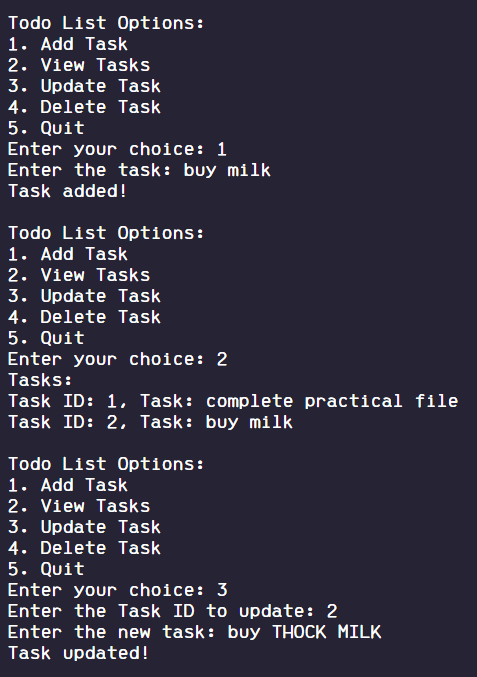
elif choice == '5':

break

cursor.close()

conn.close()

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



Program 20: Simple Password Manager made in MySQL using Python.

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

import mysql.connector as mysql

conn = mysql.connect(host="localhost", user="root", password="1-177155-400")

cursor = conn.cursor()

cursor.execute("USE school ")

cursor.execute("CREATE TABLE IF NOT EXISTS passwords (id INT AUTO\_INCREMENT PRIMARY KEY, website VARCHAR(255), username VARCHAR(255), password VARCHAR(255))")

def add\_password(website, username, password):

query = "INSERT INTO passwords (website, username, password) VALUES (%s, %s, %s)"

cursor.execute(query, (website, username, password))

conn.commit()

def view\_passwords():

cursor.execute("SELECT \* FROM passwords")

passwords = cursor.fetchall()

if passwords:

for entry in passwords:

print(f"Website: {entry[1]}, Username: {entry[2]}, Password: {entry[3]}")

else:

print("No passwords stored.")

def search\_password(website):

query = "SELECT \* FROM passwords WHERE website = %s"

cursor.execute(query, (website,))

password = cursor.fetchone()

if password:

print(f"Website: {password[1]}, Username: {password[2]}, Password: {password[3]}")

else:

print(f"No password found for {website}.")

def delete\_password(website):

query = "DELETE FROM passwords WHERE website = %s"

cursor.execute(query, (website,))

conn.commit()

print(f"Password entry for {website} deleted.")

while True:

print("\nPassword Manager Options:")

print("1. Add Password")

print("2. View Passwords")

print("3. Search Password")

print("4. Delete Password")

print("5. Quit")

choice = input("Enter your choice: ")

if choice == '1':

website = input("Enter the website: ")

username = input("Enter the username: ")

password = input("Enter the password: ")

add\_password(website, username, password)

print("Password added!")

elif choice == '2':

print("Stored Passwords:")

view\_passwords()

elif choice == '3':

website = input("Enter the website to search: ")

search\_password(website)

elif choice == '4':

website = input("Enter the website to delete: ")

delete\_password(website)

elif choice == '5':

break

cursor.close()

conn.close()

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

