COMPUTER SCIENCE PROJECT

Soumorup Chakrabarti
XII-A

ACKNOWLEDGEMENT

In the accomplishment of this project successfully, I would like to thank all the people who have been concerned with this project.it would not have been possible without the kind support and help of many individuals and organizations. I take this opportunity to express my gratitude to all of them.

Primarily I would thank God for providing me with everything that I required for completing this project. Then I would thank our principal Mr. Alok Katdare and computer science in charge Mrs. Sunita Arora, who gave me the golden opportunity to do this wonderful project, which also helped me in doing a lot of research and I came to know many new things. Their support and suggestions proved valuable in enabling the successful completion of this project.

My sincere thanks to my parents, friends and all those who have been instrumental in the success of this project within the limited time frame.

TABLE OF CONTENTS

4 INTRODUCTION

5 AIM

6 PROJECT OUTLINE

7 HARDWARE AND SOFTWARE

8 PROCESS FLOW

9 SOURCE CODE

81 OUTPUT

88 FUTURE SCOPE

89BIBLIOGRAPHY


Ministry of Education
Ministry of Information and Broadcasting
Ministry of MSME
Ministry of Women and Child Development
Ministry of Commerce & Industry
Ministry of Textiles
सत्यमेव जयते
Ministry of Education's INNOVATION CELL (GOVERNMENT OF INDIA)
Toycathon 2021
UCID: TOYCATHON/2021/Fun Academy/
CERTIFICATE
Winner
This Certificate is awarded to
Soumorup Chakrabarti
of team Fun Academy
The Winner of 'Toycathon, 2021'.
Heartiest Congratulations!
Prof. Anil Sahasrabudhe
Chairman
AICTE
Dr. Abhay Jere
Chief Innovation Officer
Ministry of Education Innovation Cell
Dr. Mohit Gambhir
Innovation Director
Ministry of Education Innovation Cell

Reliance Foundation School, Koparkhairane

CERTIFICATE

This is to certify that _____, of class 12, AISSCE Roll No._____ of Reliance Foundation School, Koparkhairane has successfully completed the _____ Project titled _____

during the academic session 2021- 2022.

Signature of Internal Examiner: Signature of External Examiner:

School Seal:

INTRODUCTION

Adolescents generally have a shorter attention span and as a result, are easily distracted. They tend to engage in a multitude of activities which divert their attention from their academics. These include but are not limited to browsing social media and playing video games. As a result, we have come up with an application via. which academics can be implemented in the form of entertaining video games.

Our goal is to make our application an interactive and fun filled medium through which students can learn academic concepts whilst keeping themselves entertained. Video games are an art form as well, and whilst integrated with academic subjects such as mathematics and science, truly emphasise on educational learning. Thus, by our application, we also attempt to remove the popular misconception in most parents' minds that video games only affect children negatively.

AIM

Elaborating on our goals and ideals, we've aimed at:

- Creating education-based learning games.
- Integrating tinkering with coding.
- Innovating fun games for making complex mathematical operations made easy for students.
- Brainstorming the creative young minds to learn, to reason and to excel in logic-based subjects.
- Integrating coding with mathematics, science, and logic.

PROJECT OUTLINE

1. **Puzzle Mania**: A multiple-choice game wherein, A 4x4 puzzle game with 15 segments of a larger image is displayed. The user slides the segments using the integrated controls in order to unscramble the image. The images are usually related to Indian history, geography and general knowledge.

2. **Chemistry Wizard**: A multiple-choice game wherein the user is asked questions with regards to chemistry and the elements of the Periodic Table. This game has been made intuitive and interactive with pictures and GFX.

3. **Numero Snake**: A special variant of snake with mathematical twists and turns which is sure to give you a good time.

4. **Graphical Genius**: A game that tests your statistical skills by making you analyse a graph and deduct conclusions from the same

.5. **Money Bender**: Another multiple-choice oriented game which involves Micro and Macroeconomics principles with the added pressure of time.

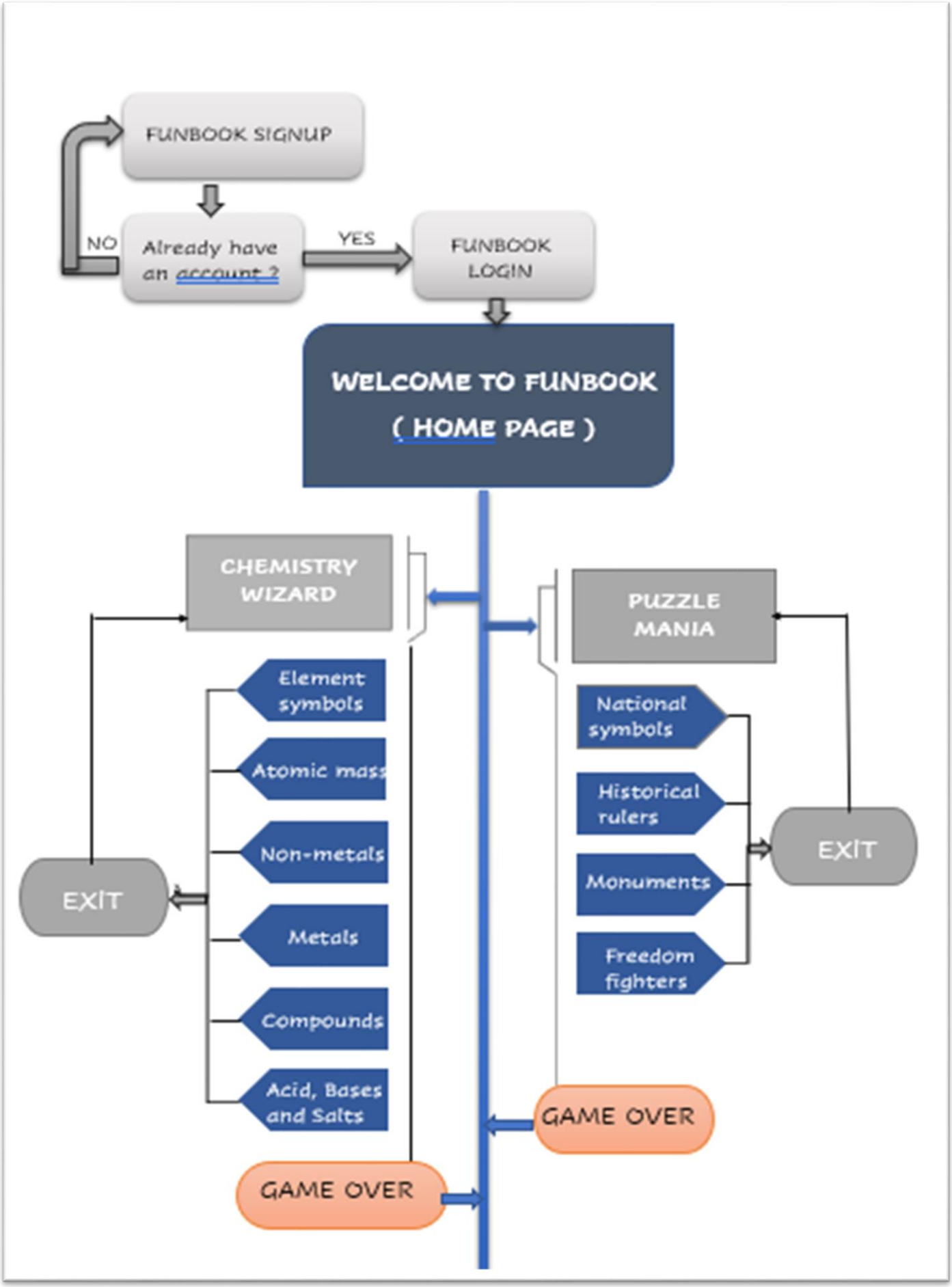
HARDWARE

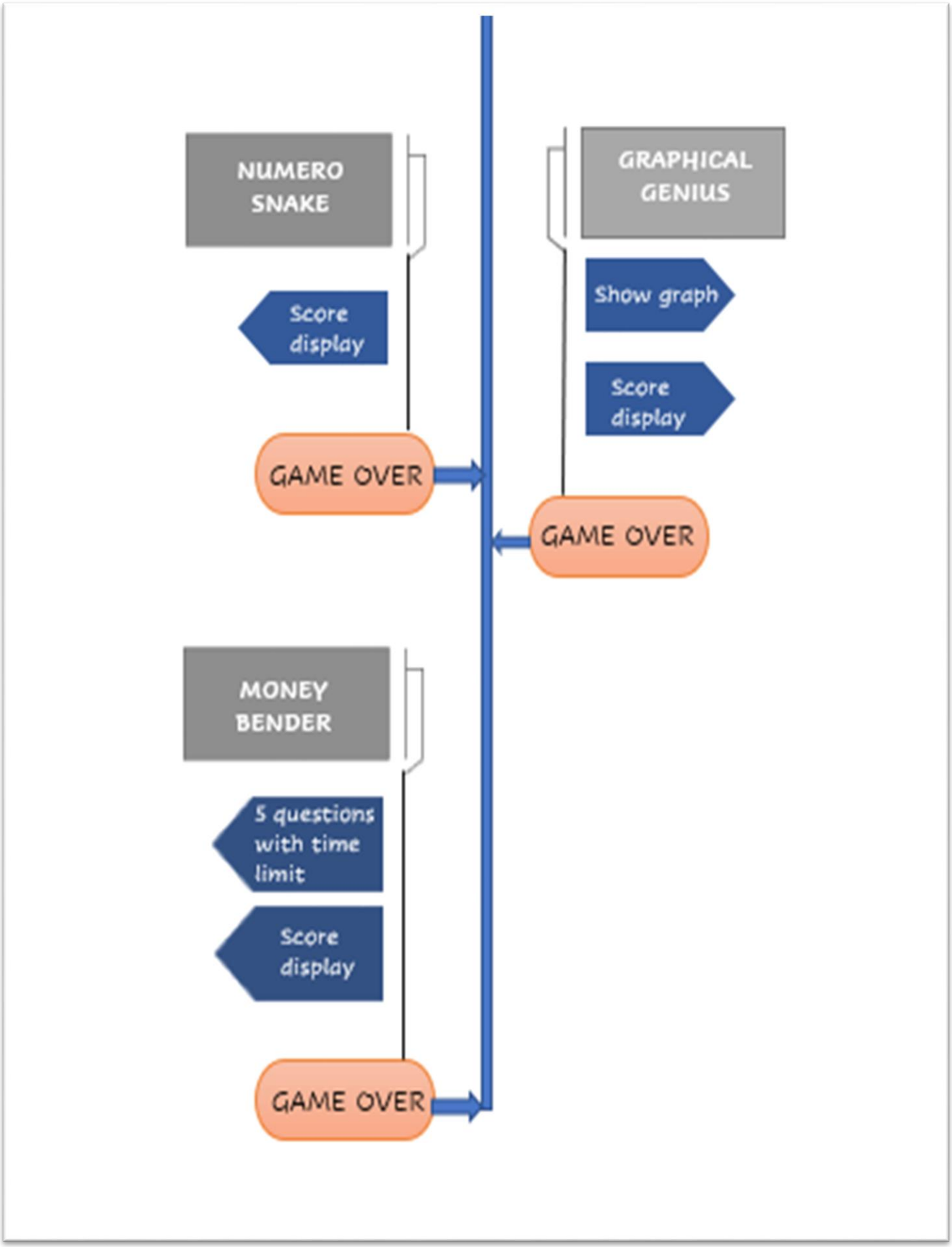
- **Device name:** ASUS LAPTOP-TUVJ2VDQ
- **Processor:** Intel(R) Core(TM) i7-8565U
- **Installed RAM:** 8.00 GB (7.86 GB usable)

SOFTWARE

- **FOR CODING** : Python in IDLE
- **GUI** : Tkinter and turtle
- **BACKEND DATABASE** : MySQL

PROCESS FLOW




NUMERO SNAKE
Score display
GAME OVER
GRAPHICAL GENIUS
Show graph
Score display
GAME OVER
MONEY BENDER
5 questions with time limit
Score display
GAME OVER

SOURCE CODE

1)Homepage Code

```
import tkinter as tk
from tkinter import *
from tkinter import messagebox
import os
global root_1
global win1
global win2
global win3
global win4
global username
import random
import mysql.connector as mc
import time
username="word"
global username_1
def mainscreen1():
    win=Tk()
    win.geometry("720x400")
    l=Label(win,text=" welcome to funbook ")
    l.pack()
def first_game():
    #global win1
    #win1=Tk()
    os.system('GAME_5_UPDATED.py')
```

```
def second_game():
    #global win2
    #win2=Tk()
    os.system('5_GAME_QUIZ.py')

def third_game():
    #global win3
    #win3=Tk()
    #os.system('1_GAME_SNAKE.py')
    import GAME_SNAKE
    obj=mc.connect(host='localhost',user='root',password='Prerna@1004',database='game')
    cursor=obj.cursor()
    print(username_1,'permanganate')
    cursor.execute('select * from snake')
    r1=cursor.fetchall()
    print(r1)
    b=0
    for i in r1:

        if (username_1)==i[0] and i[1]<GAME_SNAKE.prerna:
            b=1
            break
        elif username_1==i[0] and i[1]>GAME_SNAKE.prerna:
            b=2
            break
    if b==1:
        variable=("update snake set high_score={} where name='{}'")
        cursor.execute(variable.format(GAME_SNAKE.prerna,username_1))
        obj.commit()
        cursor.execute('select * from snake')
        print(cursor.fetchall(),"signal")
```

```
    elif b!=2:
        cursor.execute("insert into snake values('{}','{}')".format(username_1,(GAME_SNAKE.prerna)))
        obj.commit()
def fourth_game():
    #global win4
    #win4=Tk()
    os.system('bar_graph.py')
def fifth_game():
    #global win4
    #win4=Tk()
    os.system('quiz_final.py')
def home_page():
    global root_1
    root_1 = tk.Toplevel()

    global bg_1,icon1,icon2,icon3,icon4,icon5
    bg_1 = PhotoImage(file="background.png")
    icon1=PhotoImage(file="memory.png")
    icon2=PhotoImage(file="chemlogo.png")
    icon3=PhotoImage(file="snake.png")
    icon4=PhotoImage(file="bar.png")
    icon5=PhotoImage(file="money.png")

    canvas_1 = Canvas( root_1, width = 720, height = 400)
    canvas_1.pack(fill = "both", expand = True)
    canvas_1.create_image( 0, 0, image = bg_1,anchor = NW)
    button_1 = Button( root_1, width=90,height=100,text = "GAME
1",command=first_game,image=icon1,font=("Arial Bold",15),bg="yellow",fg="red")
    button_2 = Button( root_1, width=90,height=100,text = "GAME
2",command=second_game,image=icon2,font=("Arial Bold",15),bg="yellow",fg="red")
    button_3 = Button( root_1, width=90,height=100,text = "GAME
3",command=third_game,image=icon3,font=("Arial Bold",15),bg="yellow",fg="red")
```

```
    button_4 = Button( root_1, width=90,height=100,text = "GAME
4",command=fourth_game,image=icon4,font=("Arial Bold",15),bg="yellow",fg="red")
    button_5 = Button( root_1, width=90,height=100,text = "GAME
5",command=fifth_game,image=icon5,font=("Arial Bold",15),bg="yellow",fg="red")
    #button_6 = Button( root_1, width=90,height=100,text =
"LEADERBOARD",command=leaderboard,font=("Arial Bold",15),bg="yellow",fg="red")
    button1_canvas = canvas_1.create_window( 100, 250, anchor = "nw", window = button_1)
    button2_canvas = canvas_1.create_window( 205, 250, anchor = "nw", window = button_2)
    button3_canvas = canvas_1.create_window( 310, 250, anchor = "nw", window = button_3)
    button4_canvas = canvas_1.create_window( 415, 250, anchor = "nw", window = button_4)
    button5_canvas = canvas_1.create_window( 520, 250, anchor = "nw", window = button_5)
   # button6_canvas = canvas_1.create_window( 100, 400, anchor = "nw", window =button_6)
    root_1.mainloop()
'''def leaderboard():
    root = Tk()
    # specify size of window.
    #root.geometry("250x170")

    # Create text widget and specify size.
    #T = Text(root, height = 5, width = 52)

    # Create label
##    l = Label(root, text = "Fact of the Day")
##    l.config(font =("Courier", 14))

    Fact = """A man can be arrested in
    Italy for wearing a skirt in public."""
    # Create button for next text.
    #T.pack()
    # Insert The Fact.
    #T.insert(tk.END, Fact)
```

```
    cursor.execute("select * from snake order by high_score desc;")
    r_4=cursor.fetchall()
    #for i in r_4:
    for i in r_4:
        l = Label(root, text = i)
        l.config(font =("Courier", 14))
        l.pack()
    tk.mainloop()'''
def loginPage(logdata):
    sup.destroy()
    global login
    login = Tk()

    user_name = StringVar()
    password = StringVar()

    login_canvas = Canvas(login,width=720,height=440,bg="dark blue")
    login_canvas.pack()

    login_frame = Frame(login_canvas,bg="light sea green")
    login_frame.place(relwidth=0.8,relheight=0.8,relx=0.1,rely=0.1)

    heading = Label(login_frame,text="Funbook Login",fg="black",bg="light sea green")
    heading.config(font=('garamond 40'))
    heading.place(relx=0.2,rely=0.1)
    #USER NAME
    ulabel = Label(login_frame,text="Username",fg='black',bg='light sea green')
    ulabel.place(relx=0.21,rely=0.4)
    uname = Entry(login_frame,bg='white',fg='black',textvariable = user_name)
    uname.config(width=42)
```

```
    uname.place(relx=0.31,rely=0.4)
    #PASSWORD
    plabel = Label(login_frame,text="Password",fg='black',bg='light sea green')
    plabel.place(relx=0.215,rely=0.5)
    pas = Entry(login_frame,bg='white',fg='black',show="*",textvariable = password)
    pas.config(width=42)
    pas.place(relx=0.31,rely=0.5)
    def check():
        global username_1
        for a,b,c,d in logdata:
            if b == uname.get() and c == pas.get():
                username_1=uname.get()
                print(username_1)
                #login.destroy()
                home_page()
                break
        else:
            error = Label(login_frame,text="Wrong Username or Password!",fg='black',bg='light sea
green')
            error.place(relx=0.37,rely=0.7)


    #LOGIN BUTTON
    log =
Button(login_frame,text='Login',padx=5,pady=5,width=5,command=check,bg='green',fg='black')
    #sp = Button(sup_frame,text='SignUp',padx=5,pady=5,width=5,command =
addUserToDataBase,bg='green')
    log.configure(width = 15,height=1, activebackground = "#33B5E5", relief = FLAT)
    log.place(relx=0.4,rely=0.6)
    login.mainloop()
def signUpPage():
    #root.destroy()
    global sup
```

```
sup = Tk()
fname = StringVar()
uname = StringVar()
passW = StringVar()
country = StringVar()
sup_canvas = Canvas(sup,width=720,height=440,bg="dark blue")
sup_canvas.pack()

sup_frame = Frame(sup_canvas,bg="light sea green")
sup_frame.place(relwidth=0.8,relheight=0.8,relx=0.1,rely=0.1)

heading = Label(sup_frame,text="Funbook SignUp",fg="black",bg="light sea green")
heading.config(font=('garamond 40'))
heading.place(relx=0.2,rely=0.1)

#full name
flabel = Label(sup_frame,text="Full Name",fg='black',bg="light sea green")
flabel.place(relx=0.21,rely=0.4)
fname = Entry(sup_frame,bg='white',fg='black',textvariable = fname)
fname.config(width=42)
fname.place(relx=0.31,rely=0.4)

#username
ulabel = Label(sup_frame,text="Username",fg='black',bg="light sea green")
ulabel.place(relx=0.21,rely=0.5)
user = Entry(sup_frame,bg='white',fg='black',textvariable = uname)
user.config(width=42)
user.place(relx=0.31,rely=0.5)
#password
plabel = Label(sup_frame,text="Password",fg='black',bg="light sea green")
plabel.place(relx=0.215,rely=0.6)
```

```
pas = Entry(sup_frame,bg='white',fg='black',show="*",textvariable = passW)
pas.config(width=42)
pas.place(relx=0.31,rely=0.6)
#country
clabel = Label(sup_frame,text="Country",fg='black',bg="light sea green")
clabel.place(relx=0.215,rely=0.7)
c = Entry(sup_frame,bg='white',fg='black',textvariable = country)
c.config(width=42)
c.place(relx=0.31,rely=0.7)
def addUserToDataBase():
    a=0
    global username
    fullname = fname.get()
    username = user.get()
    password = pas.get()
    country = c.get()

    conn = mc.connect(host='localhost',user='root',password='Prerna@1004',database='quiz')
    create = conn.cursor()
    create.execute('select * from usersignup')
    r=create.fetchall()

    for i in r:
        if (username) in i[1]:
            a=1
            break
    if a==1:
        messagebox.showerror("ERROR","Username already exists")
        sup.destroy()
        signUpPage()
    else:
```

```
        create.execute('CREATE TABLE IF NOT EXISTS userSignUp(FULLNAME text, USERNAME
text,PASSWORD text,COUNTRY text)')

        mySql_insert_query = """INSERT INTO userSignUp (fullname,username,password,country)
VALUES (%s, %s, %s, %s) """

        record =(fullname,username,password,country)

        create.execute(mySql_insert_query, record)

        conn.commit()


      #create.execute("INSERT INTO userSignUp VALUES
(?,?,?,?)",(fullname,username,password,country))

      #conn.commit()

      create.execute('SELECT * FROM userSignUp')

      z=create.fetchall()

      print(z)

#       L2.config(text="Username is "+z[0][0]+"\nPassword is "+z[-1][1])

      conn.close()

      loginPage(z)

   def gotoLogin():

      conn = mc.connect(host='localhost',user='root',password='Prerna@1004',database='quiz')

      create = conn.cursor()

      conn.commit()

      create.execute('SELECT * FROM userSignUp')

      z=create.fetchall()

      loginPage(z)

   #signup BUTTON

   sp = Button(sup_frame,text='SignUp',padx=5,pady=5,width=5,command =
addUserToDataBase,bg='green')

   sp.configure(width = 15,height=1, activebackground = "#33B5E5", relief = FLAT)

   sp.place(relx=0.4,rely=0.8)

   log = Button(sup_frame,text='Already have a Account?',padx=5,pady=5,width=5,command =
gotoLogin,bg="light sea green",fg='black')

   log.configure(width = 16,height=1, activebackground = "#33B5E5", relief = FLAT)

   log.place(relx=0.4,rely=0.9)
```

```
    sup.mainloop()
def start():
    signUpPage()
    #mainscreen()
start()
```

2)Snake Game

```
import turtle as T
import mysql.connector as mc
obj=mc.connect(host='localhost',user='root',password='Prerna@1004',database='game')
obj_1=mc.connect(host='localhost',user='root',password='Prerna@1004',database='quiz')
cursor=obj.cursor()
cursor_1=obj_1.cursor()

import tkinter.messagebox
import FINAL_HOMEPAGE_TOYCATHON
import random
from random import randrange
from random import randint
from freegames import square, vector
import pygame
global F1
global F2
global F3
global F4
global F5
global ans
global prerna
from FINAL_HOMEPAGE_TOYCATHON import username_1
print(username_1)
prerna=0
F1,F2,F3,F4,F5,ans=0,0,0,0,0,0
pen = T.Turtle()
pen.speed(0)
pen.shape("square")
pen.color("white")
pen.penup()
```

```
pen.hideturtle()

##pen.goto(0, 260)

##pen.write("Score: 0 High Score: {}".format(high_score), align="center", font=("Courier", 24, "normal"))

#print('INSTRUCTIONS:\n\n 1.The snake moves horizontally and vertically \n\n 2.There is a questions displayed on the top left corner and there are various food boxes on which different numbers are printed\n\n 3.The correct answer to the questions is printed on one of those food boxes\n\n 4.The snake is supposed to eat the food in which the correct nawer for the questions is printed \n\n 5.Any consumption of the food containing wrong answer for the questions ends the game \n\n 6.In case of a division question ,the answer contains only the whole number part and not the decimal.make sure you notice this \n\n ALL THE BEST !')

T.bgcolor('#00cc00')

#T.bgpic('grass.jpg')


def add():
    x=randint(-20,20)
    y=randint(-20,20)


    global ans
    ans=x+y
    global F1
    F1=randint(-200,200)
    global F2
    F2=randint(-200,200)
    global F3
    F3=randint(-200,200)
    global F4
    F4=randint(-200,200)
    global F5
    F5=randint(-200,200)
    #print("Find",x,"+",y,"=?")
    x1=str(x)
    y1=str(y)
```

```
    global prerna
    pen.goto(-315,240)
    pen.clear()
    pen.write("Find "+x1+"+"+"("+y1+")"+"=?",font=("Arial", 20, "normal"))
    pen.goto(75,240)
    pen.write("Your Score:",font=("Arial", 20, "normal"))
def sub():
    x=randint(-20,20)
    y=randint(-20,20)

    global ans
    ans=x-y
    global F1
    F1=randint(-200,200)
    global F2
    F2=randint(-200,200)
    global F3
    F3=randint(-200,200)
    global F4
    F4=randint(-200,200)
    global F5
    F5=randint(-200,200)
    #print("Find",x,"-",y,"=?")
    x1=str(x)
    y1=str(y)
    global prerna

    pen.goto(-315,240)
    pen.clear()
    pen.write("Find "+x1+"-"+"("+y1+")"+"=?",font=("Arial", 20, "normal"))
    pen.goto(75,240)
```

```
    pen.write("Your score:"+str(prerna)+"",font=("Arial", 20, "normal"))
def mult():
    x=randint(-20,20)
    y=randint(-20,20)

    global F1
    F1=randint(-200,200)
    global F2
    F2=randint(-200,200)
    global F3
    F3=randint(-200,200)
    global F4
    F4=randint(-200,200)
    global F5
    F5=randint(-200,200)
    global ans
    ans=x*y
    #print("Find",x,"x",y,"=?")
    x1=str(x)
    y1=str(y)
    global prerna
    pen.goto(-315,240)
    pen.clear()
    pen.write("Find "+x1+"x"+"("+y1+")"+"=?",font=("Arial", 20, "normal"))
    pen.goto(75,240)
    pen.write("Your score:"+str(prerna)+"",font=("Arial", 20, "normal"))
def div():
    x=randint(-20,20)
    y=randint(-20,20)

    global ans
```

```
    if x%y==0:
        ans=(x//y)
    else:
        ans=round((x/y),2)
    global F1
    F1=randint(-200,200)
    global F2
    F2=randint(-200,200)
    global F3
    F3=randint(-200,200)
    global F4
    F4=randint(-200,200)
    global F5
    F5=randint(-200,200)
    #print("Find",x,"/",y,"=?")
    x1=str(x)
    y1=str(y)
    global prerna

    pen.goto(-315,240)
    pen.clear()
    pen.write("Find "+x1+"/"+"("+y1+")"+"=?",font=("Arial", 20, "normal"))
    pen.goto(75,240)
    pen.write("Your score:"+str(prerna)+"",font=("Arial", 20, "normal"))
Ran=randint(1,5)
if Ran==1:
    add()
elif Ran==2:
    sub()
elif Ran==3:
```

```
    mult()
elif Ran==4:
    div()



count,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12=0,(randrange(-20,20)*10),(randrange(-
20,20)*10),(randrange(-20,20)*10),(randrange(-20,20)*10),(randrange(-20,20)*10),(randrange(-
20,20)*10),(randrange(-20,20)*10),(randrange(-20,20)*10),(randrange(-20,20)*10),(randrange(-
20,20)*10),(randrange(-20,20)*10),(randrange(-20,20)*10)


f1,f2,f3,f4,f5,f6 = vector(c1, c2),vector(c3,c4),vector(c5, c6),vector(c7, c8),vector(c9, c10),vector(c11,
c12)
snake = [vector(10, 0)]
aim = vector(0, -10)


def change(x, y):
    "Change snake direction."
    aim.x = x
    aim.y = y


def inside(head):
    "Return True if head inside boundaries."
    return -335 < head.x < 320 and -280 < head.y < 280


def move():
    "Move snake forward one segment."
    head = snake[-1].copy()
    head.move(aim)
    global prerna
    global username
    if not inside(head) or head in snake or head==f1 or head==f2 or head==f3 or head==f4 or
head==f5:
        square(head.x, head.y, 15, 'red')
```

```
    #exit()


    #sunita ma'am
    ##from FINAL_HOMEPAGE_TOYCATHON import username
    #print(username)
    #cursor.execute("insert into snake values("+username+","+str(prerna)+")")
    tkinter.messagebox.showinfo("Game Over","GAME OVER!\nThank you for playing")
    T.update()
    return
snake.append(head)

if head == f6:
    #global prerna
    prerna=prerna+10
    #print('Snake:', len(snake))
    Ran=randint(1,5)
    if Ran==1:
        add()
    elif Ran==2:
        sub()
    elif Ran==3:
        mult()
    elif Ran==4:
        div()

    f1.x = randrange(-20, 20) * 10
    f1.y = randrange(-20, 20) * 10
    f2.x = randrange(-20, 20) * 10
    f2.y = randrange(-20, 20) * 10
    f3.x = randrange(-20, 20) * 10
    f3.y = randrange(-20, 20) * 10
```

```
    f4.x = randrange(-20, 20) * 10
    f4.y = randrange(-20, 20) * 10
    f5.x = randrange(-20, 20) * 10
    f5.y = randrange(-20, 20) * 10
    f6.x = randrange(-20, 20) * 10
    f6.y = randrange(-20, 20) * 10

else:
    snake.pop(0)

T.clear()
for body in snake:
    square(body.x, body.y, 20, 'yellow')

s=randint(-100,100)
square(f1.x, f1.y, 20, 'crimson')
square(f2.x, f2.y, 20, 'crimson')
square(f3.x, f3.y, 20, 'crimson')
square(f4.x, f4.y, 20, 'crimson')
square(f5.x, f5.y, 20, 'crimson')
square(f6.x, f6.y, 20, 'crimson')

pen.goto(f6.x,f6.y)
pen.write(ans,font=("Arial", 12, "normal"))
pen.goto(f5.x,f5.y)
pen.write(F5,font=("Arial", 12, "normal"))
pen.goto(f4.x,f4.y)
pen.write(F4,font=("Arial", 12, "normal"))
pen.goto(f3.x,f3.y)
pen.write(F3,font=("Arial", 12, "normal"))
pen.goto(f2.x,f2.y)
```

```
    pen.write(F2,font=("Arial", 12, "normal"))
    pen.goto(f1.x,f1.y)
    pen.write(F1,font=("Arial", 12, "normal"))
    T.update()
    T.ontimer(move, 100)
T.hideturtle()
T.tracer(False)
T.listen()
T.onkey(lambda: change(10, 0), 'Right')
T.onkey(lambda: change(-10, 0), 'Left')
T.onkey(lambda: change(0, 10), 'Up')
T.onkey(lambda: change(0, -10), 'Down')
move()
T.done()
```

3)Money Bender

```
import tkinter as tk
from tkinter import *
import random
import sqlite3
import time
def easy():
    root.destroy()
    global e
    e = Tk()
    easy_canvas = Canvas(e,width=800,height=500,bg="crimson")
    easy_canvas.pack()

    easy_frame = Frame(easy_canvas,bg="yellow")
    easy_frame.place(relwidth=0.8,relheight=0.8,relx=0.1,rely=0.1)
    def countDown():
        check = 0
        for k in range(120, 0, -1):

            if k == 1:
                check=-1
            timer.configure(text=k)
            easy_frame.update()
            time.sleep(1)

        timer.configure(text="Times up!")
        if check==-1:
            return (-1)
        else:
            return 0
    global score
```

```
    score = 0
# first question
    a=random.randrange(100,1000,10)
    b=random.randint(1,50)
    c=random.randint(1,10)

    A=str(a)
    B=str(b)
    C=str(c)

    f1=str(random.randrange(100,1000,1))
    f2=str(random.randrange(100,1000,1))
    f3=str(random.randrange(100,1000,1))
    f4=str((a*b*c)/100)
    f6=str(random.randrange(0,9,1))
    f7=str(random.randrange(0,9,1))
    f8=str(random.randrange(0,9,1))

#2nd question
    d1=random.randrange(6000,10000,100)
    d2=random.randrange(1000,5000,100)
    d3=random.randint(10,90)
    d4=str(d1)
    d5=str(d2)
    d6=str(d3)
    d7=random.randrange(1,5,1)
    d8=str(((d1+d2)//2)*((100+d3)//100))
    d9=str(random.randrange(1000,10000,10))
    d10=str(random.randrange(1000,10000,10))
    d11=str(random.randrange(1000,10000,10))
```

```
#3rd question
    c1=random.randint(10,50)
    c2=random.randrange(5000,50000,100)
    c3=random.randint(1,13)
    c4=str(c1)
    c5=str(c2)
    c6=str(c3)
    c7=str(random.randrange(5000,50000,100))
    c8=str(random.randrange(5000,50000,100))
    c9=str(random.randrange(5000,50000,100))
    c10=str(((100+c1)*c2)//(100-c1))
    c11=random.randrange(1,5,1)

#4th question
    e1=random.randint(20,50)
    e2=random.randrange(100,1000,10)
    e3=random.randint(1,13)
    e4=str(e1)
    e5=str(e2)
    e6=str(e3)
    e7=random.randrange(1,5,1)
    e8=str((e2//(e1-e3)))
    e9=str(random.randrange(10,51,1))
    e10=str(random.randrange(10,51,1))
    e11=str(random.randrange(10,51,1))

    X1=f1+'.'+f6
    X2=f2+'.'+f7,
    X3=f3+'.'+f8,
    X4=f4
    X5=[X1,X2,X3,X4]
```

```
X6=random.choice(X5)
X5.remove(X6)
X7=random.choice(X5)
X5.remove(X7)
X8=random.choice(X5)
X5.remove(X8)
X9=random.choice(X5)
X5.remove(X9)

Y5=[d8,d9,d10,d11]

Y6=random.choice(Y5)
Y5.remove(Y6)
Y7=random.choice(Y5)
Y5.remove(Y7)
Y8=random.choice(Y5)
Y5.remove(Y8)
Y9=random.choice(Y5)
Y5.remove(Y9)

Z5=[c7,c8,c9,c10]
Z6=random.choice(Z5)
Z5.remove(Z6)
Z7=random.choice(Z5)
Z5.remove(Z7)
Z8=random.choice(Z5)
Z5.remove(Z8)
Z9=random.choice(Z5)
Z5.remove(Z9)
```

```
W5=[e8,e9,e10,e11]
W6=random.choice(W5)
W5.remove(W6)
W7=random.choice(W5)
W5.remove(W7)
W8=random.choice(W5)
W5.remove(W8)
W9=random.choice(W5)
W5.remove(W9)
easyQ = [
        [
          'Q:Calculate the simple interest if principle amount is '+A+' ,\nrate ofinterest on the principal amount is '+B+' ,and the duration \n is '+C+' years?',
           X6,
           X7,
           X8,
           X9

        ] ,
        [
          'Q:The percentage profit earned by selling an article for Rs. '+d4+' \n is equal to the percentage loss incurred by selling the same\n  article for Rs. '+d5+' . At what price should the article be sold \n to make '+d6+' profit?' ,
           Y6,
           Y7,
           Y8,
           Y9

        ],
        [
          'Q:When a plot is sold for Rs. '+c5+', the owner loses '+c4+'%. \nAt what price must that plot be sold in order to gain '+c4+'%?',
```

```
            Z6,
            Z7,
            Z8,
            Z9
        ],
        [
            'Q:On selling '+e4+' balls at Rs '+e5+' there is a loss equal to the \n cost price of '+e6+' balls. The cost price of a ball is?' ,
            W6,
            W7,
            W8,
            W9
        ]
    ]
answer = [
    f4,
    d8,
    c10,
    e8
    ]

li = ['',0,1,2,3]
x = random.choice(li[1:])

ques = Label(easy_frame,text =easyQ[x][0],font="verdana 14",bg="yellow")
ques.place(relx=0.5,rely=0.2,anchor=CENTER)

var = StringVar()
a = Radiobutton(easy_frame,text=easyQ[x][1],font="verdana 12",value=easyQ[x][1],variable = var,bg="yellow")
a.place(relx=0.5,rely=0.42,anchor=CENTER)
```

```
    b = Radiobutton(easy_frame,text=easyQ[x][2],font="verdana 12",value=easyQ[x][2],variable =
var,bg="yellow")
    b.place(relx=0.5,rely=0.52,anchor=CENTER)

    c = Radiobutton(easy_frame,text=easyQ[x][3],font="verdana 12",value=easyQ[x][3],variable =
var,bg="yellow")
    c.place(relx=0.5,rely=0.62,anchor=CENTER)

    d = Radiobutton(easy_frame,text=easyQ[x][4],font="verdana 12",value=easyQ[x][4],variable =
var,bg="yellow")
    d.place(relx=0.5,rely=0.72,anchor=CENTER)



    li.remove(x)

    timer = Label(e)
    timer.place(relx=0.8,rely=0.82,anchor=CENTER)
    def display():

        if len(li) == 1:
            e.destroy()
            showMark(score)
        if len(li) == 2:
            nextQuestion.configure(text='End',command=calc)

        if li:
            x = random.choice(li[1:])
            ques.configure(text =easyQ[x][0])

            a.configure(text=easyQ[x][1],value=easyQ[x][1])

            b.configure(text=easyQ[x][2],value=easyQ[x][2])
```

```
            c.configure(text=easyQ[x][3],value=easyQ[x][3])
            d.configure(text=easyQ[x][4],value=easyQ[x][4])
            li.remove(x)
            print(li)
            y = countDown()
            if y == -1:
                display()
        def calc():
            global score
            print("var is:",var.get())
            print("answers are:",answer)
            if (var.get() in answer):
                score+=5
            display()
        submit = Button(easy_frame,command=calc,text="Submit")
        submit.place(relx=0.5,rely=0.82,anchor=CENTER)
        nextQuestion = Button(easy_frame,command=display,text="Next")
        nextQuestion.place(relx=0.87,rely=0.82,anchor=CENTER)

        y = countDown()
        if y == -1:
            display()
        e.mainloop()
    def showMark(mark):
        global sh
        sh = Tk()

        show_canvas = Canvas(sh,width=720,height=440,bg="#101357")
        show_canvas.pack()
```

```
    show_frame = Frame(show_canvas,bg="white")
    show_frame.place(relwidth=0.8,relheight=0.8,relx=0.1,rely=0.1)
    st = "Your score is "+str(mark)
    mlabel = Label(show_canvas,text=st,fg="black")
    mlabel.place(relx=0.5,rely=0.2,anchor=CENTER)
    sh.mainloop()
def start():
    global root
    root = Tk()
    canvas = Canvas(root,width = 450,height =320)
    canvas.grid(column = 0 , row = 1)
    img = PhotoImage(file="moneypic.png")
    canvas.create_image(50,10,image=img,anchor=NW)

    button = Button(root, text='Start',command = easy)
    button.configure(width = 102,height=2, activebackground = "#33B5E5", bg ='green', relief =
RAISED)
    button.grid(column = 0 , row = 2)
    root.mainloop()
start()
```

4) Graphical Genius

```
from matplotlib import pyplot as plt
import random
import statistics
import numpy as np
from tkinter import *
import tkinter.messagebox
from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg,NavigationToolbar2Tk)

global canvas
global toolbar
global root
global p1
global p2
global p3
global y1
y1=random.randrange(1,20)
global y2
y2=random.randrange(1,20)
global y3
y3=random.randrange(1,20)
global y4
y4=random.randrange(1,20)
global y5
y5=random.randrange(1,20)
global y7
global Y1
global Y
global left
```

```
global height
global tick_label
global a1
global a2
global a3
global a4
global a5
global a6
def cm_to_inch(value):
    return value/2.54
def graph():
    global p1
    global p2
    global p3
    p1=0
    p2=0
    p3=0
    global y1
    global y2
    global y3
    global y4
    global y5
    global y7
    global Y1
    global Y
    global left
    global height
    global tick_label
    global canvas
    global toolbar
```

```
    fig=plt.figure(figsize=(5,4))
    left = [1,2,3,4,5]
    height = [y1,y2,y3,y4,y5]
    tick_label = ['pista', 'butterscotch', 'mango', 'vanilla','chocolate' ]
    plt.yticks(np.arange(0,21,1))
    plt.bar(left, height, tick_label = tick_label,width = 0.6, color = ['green','yellow', 'red','blue','brown'])
    plt.xlabel('flavours of ice-cream')
    plt.ylabel('no. of students')
    plt.title('bar chart for students choosing different ice-creams')
    canvas = FigureCanvasTkAgg(fig,master = root)
    canvas.draw()

    # placing the canvas on the Tkinter window
    canvas.get_tk_widget().grid(row=8,column=1)

    # creating the Matplotlib toolbar
    #toolbar = NavigationToolbar2Tk(canvas, root)
    #toolbar.update()

    # placing the toolbar on the Tkinter window
    canvas.get_tk_widget().grid(row=8,column=2)

# the main Tkinter window
    #plt.show()

##########################################################################################
######

def result():
    global p1
    global p2
```

```
    global p3
    global y1
    global y2
    global y3
    global y4
    global y5
    global y7
    global Y1
    global Y
    global left
    global height
    global tick_label
    global a1
    global a2
    global a3
    global a4
    global a5
    global a6
    global height
    #a1=int(input('How many birch trees did arthur see:'))
##    print(a1.get())
##    print(type(a1))
##    print(y2)
##    print(type(y2))

    if int(a1.get())==(y2):
        print('correct!')
        p1=p1+5
        p2=p2+1
    else:
        print('wrong!')
```

```
    p3=p3+1
#a2=(input('Which tree did he see least:'))

A=min(height)
print(A)
print(type(A))
print(y1)
print(type(y1))
print(y2)
print(type(y2))

print(y3)
print(type(y3))

print(y4)
print(type(y4))

print(y5)
print(type(y5))




if A==y1:
    if (a2.get()).upper()==('chocolate').upper():
        print('correct!')
        p1=p1+5
        p2=p2+1
    else:
        print('wrong!')
        p3=p3+1
```

```
elif A==y2:
    if (a2.get()).upper()==('butterscotch').upper():
        print('correct!')
        p1=p1+5
        p2=p2+1
    else:
        print('wrong!')
        p3=p3+1
elif A==y3:
    if (a2.get()).upper()==('mango').upper():
        print('correct!')
        p1=p1+5
        p2=p2+1
    else:
        print('wrong!')
        p3=p3+1
elif A==y4:
    if (a2.get()).upper()==('vanilla').upper():
        print('correct!')
        p1=p1+5
        p2=p2+1
    else:
        print('wrong!')
        p3=p3+1
elif A==y5:
    if (a2.get()).upper()==('pista').upper():
        print('correct!')
        p1=p1+5
        p2=p2+1
    else:
        print('wrong!')
```

```
        p3=p3+1
else:
    print('invalid input')



#a4=int(input('how many cedar and pine trees together'))
if int(a4.get())==(y4+y5):
    print('correct!')
    p1=p1+5
    p2=p2+1
else:
    print('wrong!')
    p3=p3+1

#a5=int(input('If arthur counted 7 more pine treee how many did he total count:'))

if int(a5.get())==(y4+7):
    print('correct!')
    p1=p1+5
    p2=p2+1
else:
    print('wrong!')
    p3=p3+1



#a6=(input('Which tree did he see most:'))
B=max(height)
if B==y1:
    if (a6.get()).upper()==('chocolate').upper():
        print('correct!')
```

```
        p1=p1+5
        p2=p2+1
    else:
        print('wrong!')
        p3=p3+1
elif B==y2:
    if (a6.get()).upper()==('butterscotch').upper():
        print('correct!')
        p1=p1+5
        p2=p2+1
    else:
        print('wrong!')
        p3=p3+1
elif B==y3:
    if (a6.get()).upper()==('mango').upper():
        print('correct!')
        p1=p1+5
        p2=p2+1
    else:
        print('wrong!')
        p3=p3+1
elif B==y4:
    if (a6.get()).upper()==('vanilla').upper():
        print('correct!')
        p1=p1+5
        p2=p2+1
    else:
        print('wrong!')
        p3=p3+1
elif B==y5:
    if (a6.get()).upper()==('pista').upper():
```

```
            print('correct!')
            p1=p1+5
            p2=p2+1
        else:
            print('wrong!')
            p3=p3+1
    else:
        print('invalid input')
    print('Your total score is ',p1,' ,correct answers:',p2,' ,wrong answers:',p3)
    p1=str(p1)
    tkinter.messagebox.showinfo("Score Card","Your total score is:"+p1)

################################################################################
p1=0
p2=0
p3=0
y1=random.randrange(3,21,1)
y2=random.randrange(4,21,1)
y3=random.randrange(5,21,1)
y4=random.randrange(4,21,1)
y5=random.randrange(3,21,1)
y7=random.randint(3,20)
Y1=random.choice([y1,y2,y3,y4,y5])
Y=str(Y1)

root=Tk()
#root.attributes('-fullscreen', True)
root.configure(bg='orange')
root.title("Graphical genius")
root.geometry("1350x800")
```

```
a1=IntVar()

a2=StringVar()

a3=StringVar()

a4=IntVar()

a5=IntVar()

a6=StringVar()


#L1=Label(root,text="GRAPH GAME",bg='yellow',fg='red',font="lucida 15 bold").grid(columnspan=2)

b1=Button(root,text="SHOW GRAPH",command=graph,bg='crimson',fg='white',font="verdana 20
bold").grid(columnspan=1)


L2=Label(root,text='How many students liked butterscotch:',fg='dark blue',font='verdana 12
bold').grid(row=2,column=0)

T1=Entry(root,textvariable=a1).grid(row=2,column=1,padx=20,pady=10,ipadx=15,ipady=2)


L3=Label(root,text='Which flavour was least liked:',fg='dark blue',font='verdana 12
bold').grid(row=3,column=0)

T2=Entry(root,textvariable=a2).grid(row=3,column=1,padx=20,pady=10,ipadx=15,ipady=2)


##L4=Label(root,text="Which tree did they count "+Y+" of:",fg='green',font='lucida 15
bold').grid(row=4,column=0)

##T3=Entry(root,textvariable=a3).grid(row=4,column=1,padx=20,pady=10,ipadx=15,ipady=5)

##

L5=Label(root,text='how many students liked pista and vanilla together:',fg='dark
blue',font='verdana 12 bold').grid(row=5,column=0)

T4=Entry(root,textvariable=a4).grid(row=5,column=1,padx=20,pady=10,ipadx=15,ipady=2)


L6=Label(root,text='If 7 more vanilla choosers were counted,what is the total:',fg='dark
blue',font='verdana 12 bold').grid(row=6,column=0)

T5=Entry(root,textvariable=a5).grid(row=6,column=1,padx=20,pady=10,ipadx=15,ipady=2)


L7=Label(root,text='Which flavour was liked the most:',fg='dark blue',font='verdana 12
bold').grid(row=7,column=0)
```

```
T6=Entry(root,textvariable=a6).grid(row=7,column=1,padx=20,pady=10,ipadx=15,ipady=2)


b2=Button(root,text="Submit",command=result,bg='green',fg='white',font="verdana 12
bold").grid(columnspan=1)


root.mainloop()
```

5)Chemistry Wizard

```
from tkinter import *


def start_main_page():
    def start_game(args):
        main_window.destroy()
        if args == 1:
            from Options import Animals
            Animals.main()
        elif args == 2:
            from Options import Body_parts
            Body_parts.main()
        elif args == 3:
            from Options import Colour
            Colour.main()
        elif args == 4:
            from Options import Fruit
            Fruit.main()
        elif args == 5:
            from Options import Shapes
            Shapes.main()
        elif args == 6:
            from Options import Vegetable
            Vegetable.main()
        elif args == 7:
```

```
        from Options import Vehicles
        Vehicles.main()

def option():

    lab_img1 = Button(
        main_window,
        image=img1,
        bg='#e6fff5',
        border=0,
        justify='center',

    )
    sel_btn1 = Button(
        text="Element symbols",
        width=18,
        borderwidth=8,
        font=("", 18),
        fg="#000000",
        bg="#99ffd6",
        cursor="hand2",
        command=lambda: start_game(1),
    )

    sel_btn2 = Button(
        text="Atomic mass",
        width=18,
        borderwidth=8,
        font=("", 18),
        fg="#000000",
        bg="#99ffd6",
```

```
    cursor="hand2",
    command=lambda: start_game(2),
)

sel_btn3 = Button(
    text="Non-metals",
    width=18,
    borderwidth=8,
    font=("", 18),
    fg="#000000",
    bg="#99ffd6",
    cursor="hand2",
    command=lambda: start_game(3),
)

sel_btn4 = Button(
    text="Metals",
    width=18,
    borderwidth=8,
    font=("", 18),
    fg="#000000",
    bg="#99ffd6",
    cursor="hand2",
    command=lambda: start_game(4),
)

sel_btn5 = Button(
    text="Compounds",
    width=18,
    borderwidth=8,
    font=("", 18),
```

```
        fg="#000000",
        bg="#99ffd6",
        cursor="hand2",
        command=lambda: start_game(5),
    )

    sel_btn6 = Button(
        text="Acids,bases and salts",
        width=18,
        borderwidth=8,
        font=("", 18),
        fg="#000000",
        bg="#99ffd6",
        cursor="hand2",
        command=lambda: start_game(6),
    )

##      sel_btn7 = Button(
##          text="Vehicles",
##          width=18,
##          borderwidth=8,
##          font=("", 18),
##          fg="#000000",
##          bg="#99ffd6",
##          cursor="hand2",
##          command=lambda: start_game(7),
##      )
    lab_img1.grid(row=0, column=0, padx=20)
    sel_btn1.grid(row=1, column=4, pady=(10, 0), padx=50, )
    sel_btn2.grid(row=2, column=4, pady=(10, 0), padx=50, )
    sel_btn3.grid(row=3, column=4, pady=(10, 0), padx=50, )
```

```
    sel_btn4.grid(row=4, column=4, pady=(10, 0), padx=50, )
    sel_btn5.grid(row=5, column=4, pady=(10, 0), padx=50, )
    sel_btn6.grid(row=6, column=4, pady=(10, 0), padx=50, )
    #sel_btn7.grid(row=6, column=4, pady=(10, 0), padx=50, )

def show_option():
    start_btn.destroy()

    lab_img.destroy()
    option()

main_window = Tk()

main_window.geometry("500x500+400+100")
main_window.resizable(0, 0)
main_window.title("chemistry wizard")
main_window.configure(background="#e6fff5")
#main_window.iconbitmap(r'quizee_logo_.ico')

img0 = PhotoImage(file="chemwizard1.png")
img1 = PhotoImage(file="back.png")

lab_img = Label(
    main_window,
    image=img0,
    bg='#e6fff5',
)
lab_img.pack(pady=(70, 0))

start_btn = Button(
    main_window,
```

```
        text="Start",
        width=18,
        borderwidth=8,
        fg="#000000",
        bg="#99ffd6",
        font=("", 13),
        cursor="hand2",
        command=show_option,
    )
    start_btn.pack(pady=(40, 20))

    main_window.mainloop()


start_main_page()
```

6)Puzzle Mania

```
from tkinter import *


def start_main_page():
    def start_game(args):
        main_window.destroy()
        if args == 1:
            import random
            import tkinter as tk
            import tkinter.messagebox
            from PIL import Image, ImageTk
            import random
            #tk.geometry(1000x1000)
            MAX_BOARD_SIZE = 600
```

```
files=['elephant.jpeg','mango.jpeg','tiger.jpeg','lion.jpeg','peacock.jpeg']
a=random.choice(files)
class Application(tk.Frame):

    def __init__(self, image, board_grid=4):
        tk.Frame.__init__(self)
        self.grid()
        self.board_grid = board_grid if board_grid > 2 else 3
        self.load_image(image)
        self.steps = 0
        self.create_widgets()
        self.create_events()
        self.create_board()
        self.show()

    def load_image(self, image):
        image = Image.open(image)
        board_size = min(image.size)
        if image.size[0] != image.size[1]:
            image = image.crop((0, 0, board_size, board_size))
        if board_size > MAX_BOARD_SIZE:
            board_size = MAX_BOARD_SIZE
            image = image.resize((board_size, board_size), Image.ANTIALIAS)
        self.image = image
        self.board_size = board_size
        self.piece_size = self.board_size / self.board_grid

    def create_widgets(self):
        args = dict(width=self.board_size, height=self.board_size)
        self.canvas = tk.Canvas(self, **args)
        self.canvas.grid(row=0,column=0)
```

```
        #self.canvas = tk.Canvas(width=300, height=300, bg='white')
        #self.canvas.grid()

        load = Image.open(a)
        render = ImageTk.PhotoImage(load)

        widget = tk.Label(self, image=render, fg='white', bg='black')
        widget.image = render
        widget.place(x=0,y=0)
        widget.grid(row=0,column=1)
        #self.canvas.create_window(100, 100, window=widget)

    def create_events(self):
        self.canvas.bind_all('<KeyPress-Up>', self.slide)
        self.canvas.bind_all('<KeyPress-Down>', self.slide)
        self.canvas.bind_all('<KeyPress-Left>', self.slide)
        self.canvas.bind_all('<KeyPress-Right>', self.slide)
        self.canvas.bind_all('<KeyPress-h>', self.slide)
        self.canvas.bind_all('<KeyPress-j>', self.slide)
        self.canvas.bind_all('<KeyPress-k>', self.slide)
        self.canvas.bind_all('<KeyPress-l>', self.slide)
        self.canvas.bind_all('<KeyPress-H>', self.help)

    def help(self, event):
        if getattr(self, '_img_help_id', None) is None:
            self._img_help = ImageTk.PhotoImage(self.image)
            self._img_help_id = self.canvas.create_image(0, 0,
                    image=self._img_help, anchor=tk.NW)
        else:
            state = self.canvas.itemcget(self._img_help_id, 'state')
```

```
        state = 'hidden' if state == '' else ''
        self.canvas.itemconfigure(self._img_help_id, state=state)

    def slide(self, event):
        pieces = self.get_pieces_around()
        if event.keysym in ('Up', 'k') and pieces['bottom']:
            self._slide(pieces['bottom'], pieces['center'],
                        (0, -self.piece_size))
        if event.keysym in ('Down', 'j') and pieces['top']:
            self._slide(pieces['top'], pieces['center'],
                        (0, self.piece_size))
        if event.keysym in ('Left', 'h') and pieces['right']:
            self._slide(pieces['right'], pieces['center'],
                        (-self.piece_size, 0))
        if event.keysym in ('Right', 'l') and pieces['left']:
            self._slide(pieces['left'], pieces['center'],
                        (self.piece_size, 0))
        self.check_status()

    def _slide(self, from_, to, coord):
        self.canvas.move(from_['id'], *coord)
        to['pos_a'], from_['pos_a'] = from_['pos_a'], to['pos_a']
        self.steps += 1

    def get_pieces_around(self):
        pieces = {'center': None,
                  'right' : None,
                  'left'  : None,
                  'top'   : None,
                  'bottom': None}
        for piece in self.board:
```

```
            if not piece['visible']:
                pieces['center'] = piece
                break
        x0, y0 = pieces['center']['pos_a']
        for piece in self.board:
            x1, y1 = piece['pos_a']
            if y0 == y1 and x1 == x0 + 1:
                pieces['right'] = piece
            if y0 == y1 and x1 == x0 - 1:
                pieces['left'] = piece
            if x0 == x1 and y1 == y0 - 1:
                pieces['top'] = piece
            if x0 == x1 and y1 == y0 + 1:
                pieces['bottom'] = piece
        return pieces

    def create_board(self):
        self.board = []
        for x in range(self.board_grid):
            for y in range(self.board_grid):
                x0 = x * self.piece_size
                y0 = y * self.piece_size
                x1 = x0 + self.piece_size
                y1 = y0 + self.piece_size
                image = ImageTk.PhotoImage(
                        self.image.crop((x0, y0, x1, y1)))
                piece = {'id'     : None,
                         'image'  : image,
                         'pos_o'  : (x, y),
                         'pos_a'  : None,
                         'visible': True}
```

```
                self.board.append(piece)
        self.board[-1]['visible'] = False

    def check_status(self):
        for piece in self.board:
            if piece['pos_a'] != piece['pos_o']:
                return
        title = 'Ganaste!'
        message = 'Lo resolviste en %d movidas!' % self.steps
        tkMessageBox.showinfo(title, message)

    def show(self):
        random.shuffle(self.board)
        index = 0
        for x in range(self.board_grid):
            for y in range(self.board_grid):
                self.board[index]['pos_a'] = (x, y)
                if self.board[index]['visible']:
                    x1 = x * self.piece_size
                    y1 = y * self.piece_size
                    image = self.board[index]['image']
                    id = self.canvas.create_image(
                            x1, y1, image=image, anchor=tk.NW)
                    self.board[index]['id'] = id
                index += 1


if __name__ == '__main__':
    from optparse import OptionParser
    parser = OptionParser(description="Sliding puzzle")
    parser.add_option('-g', '--board-grid', type=int, default=4,
```

```
                help="(the minimum value is 3)")
        parser.add_option('-i', '--image', type=str, default=a,
                help="path to image")
        args, _ = parser.parse_args()

        if args.board_grid < 3:
           args.board_grid = 3
           print("Warning: using 3 for board-grid")

        app = Application(args.image, args.board_grid)
        app.master.title('Sliding puzzle')
        app.mainloop()

    #from Options import kingss
    #kingss.main()

elif args == 2:
        import random
        import tkinter as tk
        import tkinter.messagebox
        from PIL import Image, ImageTk
        import random
        #tk.geometry(1000x1000)
        MAX_BOARD_SIZE = 600
        files=['pic_1.jpeg','pic_3.jpeg','pic_5.jpeg']
        a=random.choice(files)
        class Application(tk.Frame):

           def __init__(self, image, board_grid=4):
              tk.Frame.__init__(self)
              self.grid()
```

```
        self.board_grid = board_grid if board_grid > 2 else 3
        self.load_image(image)
        self.steps = 0
        self.create_widgets()
        self.create_events()
        self.create_board()
        self.show()

    def load_image(self, image):
        image = Image.open(image)
        board_size = min(image.size)
        if image.size[0] != image.size[1]:
            image = image.crop((0, 0, board_size, board_size))
        if board_size > MAX_BOARD_SIZE:
            board_size = MAX_BOARD_SIZE
            image = image.resize((board_size, board_size), Image.ANTIALIAS)
        self.image = image
        self.board_size = board_size
        self.piece_size = self.board_size / self.board_grid

    def create_widgets(self):
        args = dict(width=self.board_size, height=self.board_size)
        self.canvas = tk.Canvas(self, **args)
        self.canvas.grid(row=0,column=0)

        #self.canvas = tk.Canvas(width=300, height=300, bg='white')
        #self.canvas.grid()

        load = Image.open(a)
        render = ImageTk.PhotoImage(load)
```

```
    widget = tk.Label(self, image=render, fg='white', bg='black')
    widget.image = render
    widget.place(x=0,y=0)
    widget.grid(row=0,column=1)
    #self.canvas.create_window(100, 100, window=widget)

def create_events(self):
    self.canvas.bind_all('<KeyPress-Up>', self.slide)
    self.canvas.bind_all('<KeyPress-Down>', self.slide)
    self.canvas.bind_all('<KeyPress-Left>', self.slide)
    self.canvas.bind_all('<KeyPress-Right>', self.slide)
    self.canvas.bind_all('<KeyPress-h>', self.slide)
    self.canvas.bind_all('<KeyPress-j>', self.slide)
    self.canvas.bind_all('<KeyPress-k>', self.slide)
    self.canvas.bind_all('<KeyPress-l>', self.slide)
    self.canvas.bind_all('<KeyPress-H>', self.help)

def help(self, event):
    if getattr(self, '_img_help_id', None) is None:
        self._img_help = ImageTk.PhotoImage(self.image)
        self._img_help_id = self.canvas.create_image(0, 0,
                image=self._img_help, anchor=tk.NW)
    else:
        state = self.canvas.itemcget(self._img_help_id, 'state')
        state = 'hidden' if state == '' else ''
        self.canvas.itemconfigure(self._img_help_id, state=state)

def slide(self, event):
    pieces = self.get_pieces_around()
    if event.keysym in ('Up', 'k') and pieces['bottom']:
        self._slide(pieces['bottom'], pieces['center'],
```

```
                (0, -self.piece_size))
        if event.keysym in ('Down', 'j') and pieces['top']:
            self._slide(pieces['top'], pieces['center'],
                (0, self.piece_size))
        if event.keysym in ('Left', 'h') and pieces['right']:
            self._slide(pieces['right'], pieces['center'],
                (-self.piece_size, 0))
        if event.keysym in ('Right', 'l') and pieces['left']:
            self._slide(pieces['left'], pieces['center'],
                (self.piece_size, 0))
        self.check_status()

    def _slide(self, from_, to, coord):
        self.canvas.move(from_['id'], *coord)
        to['pos_a'], from_['pos_a'] = from_['pos_a'], to['pos_a']
        self.steps += 1

    def get_pieces_around(self):
        pieces = {'center': None,
                  'right' : None,
                  'left'  : None,
                  'top'   : None,
                  'bottom': None}
        for piece in self.board:
            if not piece['visible']:
                pieces['center'] = piece
                break
        x0, y0 = pieces['center']['pos_a']
        for piece in self.board:
            x1, y1 = piece['pos_a']
            if y0 == y1 and x1 == x0 + 1:
```

```
            pieces['right'] = piece
        if y0 == y1 and x1 == x0 - 1:
            pieces['left'] = piece
        if x0 == x1 and y1 == y0 - 1:
            pieces['top'] = piece
        if x0 == x1 and y1 == y0 + 1:
            pieces['bottom'] = piece
    return pieces

def create_board(self):
    self.board = []
    for x in range(self.board_grid):
        for y in range(self.board_grid):
            x0 = x * self.piece_size
            y0 = y * self.piece_size
            x1 = x0 + self.piece_size
            y1 = y0 + self.piece_size
            image = ImageTk.PhotoImage(
                  self.image.crop((x0, y0, x1, y1)))
            piece = {'id'     : None,
                     'image'  : image,
                     'pos_o'  : (x, y),
                     'pos_a'  : None,
                     'visible': True}
            self.board.append(piece)
    self.board[-1]['visible'] = False

def check_status(self):
    for piece in self.board:
        if piece['pos_a'] != piece['pos_o']:
            return
```

```
            title = 'Ganaste!'
            message = 'Lo resolviste en %d movidas!' % self.steps
            tkMessageBox.showinfo(title, message)

        def show(self):
            random.shuffle(self.board)
            index = 0
            for x in range(self.board_grid):
                for y in range(self.board_grid):
                    self.board[index]['pos_a'] = (x, y)
                    if self.board[index]['visible']:
                        x1 = x * self.piece_size
                        y1 = y * self.piece_size
                        image = self.board[index]['image']
                        id = self.canvas.create_image(
                                x1, y1, image=image, anchor=tk.NW)
                        self.board[index]['id'] = id
                    index += 1


    if __name__ == '__main__':
        from optparse import OptionParser
        parser = OptionParser(description="Sliding puzzle")
        parser.add_option('-g', '--board-grid', type=int, default=4,
                          help="(the minimum value is 3)")
        parser.add_option('-i', '--image', type=str, default=a,
                          help="path to image")
        args, _ = parser.parse_args()

        if args.board_grid < 3:
            args.board_grid = 3
```

```
            print("Warning: using 3 for board-grid")

        app = Application(args.image, args.board_grid)
        app.master.title('Sliding puzzle')
        app.mainloop()
elif args == 3:
    import random
    import tkinter as tk
    import tkinter.messagebox
    from PIL import Image, ImageTk
    import random
    #tk.geometry(1000x1000)
    MAX_BOARD_SIZE = 600
    files=['s_2.jpeg','s_1.jpeg','s_3.jpeg','s_4.jpeg','s_5.jpeg',]
    a=random.choice(files)
    class Application(tk.Frame):

        def __init__(self, image, board_grid=4):
            tk.Frame.__init__(self)
            self.grid()
            self.board_grid = board_grid if board_grid > 2 else 3
            self.load_image(image)
            self.steps = 0
            self.create_widgets()
            self.create_events()
            self.create_board()
            self.show()

        def load_image(self, image):
            image = Image.open(image)
            board_size = min(image.size)
```

```
        if image.size[0] != image.size[1]:
            image = image.crop((0, 0, board_size, board_size))
        if board_size > MAX_BOARD_SIZE:
            board_size = MAX_BOARD_SIZE
            image = image.resize((board_size, board_size), Image.ANTIALIAS)
        self.image = image
        self.board_size = board_size
        self.piece_size = self.board_size / self.board_grid

    def create_widgets(self):
        args = dict(width=self.board_size, height=self.board_size)
        self.canvas = tk.Canvas(self, **args)
        self.canvas.grid(row=0,column=0)

        #self.canvas = tk.Canvas(width=300, height=300, bg='white')
        #self.canvas.grid()

        load = Image.open(a)
        render = ImageTk.PhotoImage(load)

        widget = tk.Label(self, image=render, fg='white', bg='black')
        widget.image = render
        widget.place(x=0,y=0)
        widget.grid(row=0,column=1)
        #self.canvas.create_window(100, 100, window=widget)

    def create_events(self):
        self.canvas.bind_all('<KeyPress-Up>', self.slide)
        self.canvas.bind_all('<KeyPress-Down>', self.slide)
        self.canvas.bind_all('<KeyPress-Left>', self.slide)
        self.canvas.bind_all('<KeyPress-Right>', self.slide)
```

```
    self.canvas.bind_all('<KeyPress-h>', self.slide)
    self.canvas.bind_all('<KeyPress-j>', self.slide)
    self.canvas.bind_all('<KeyPress-k>', self.slide)
    self.canvas.bind_all('<KeyPress-l>', self.slide)
    self.canvas.bind_all('<KeyPress-H>', self.help)

def help(self, event):
    if getattr(self, '_img_help_id', None) is None:
        self._img_help = ImageTk.PhotoImage(self.image)
        self._img_help_id = self.canvas.create_image(0, 0,
                image=self._img_help, anchor=tk.NW)
    else:
        state = self.canvas.itemcget(self._img_help_id, 'state')
        state = 'hidden' if state == '' else ''
        self.canvas.itemconfigure(self._img_help_id, state=state)

def slide(self, event):
    pieces = self.get_pieces_around()
    if event.keysym in ('Up', 'k') and pieces['bottom']:
        self._slide(pieces['bottom'], pieces['center'],
                (0, -self.piece_size))
    if event.keysym in ('Down', 'j') and pieces['top']:
        self._slide(pieces['top'], pieces['center'],
                (0, self.piece_size))
    if event.keysym in ('Left', 'h') and pieces['right']:
        self._slide(pieces['right'], pieces['center'],
                (-self.piece_size, 0))
    if event.keysym in ('Right', 'l') and pieces['left']:
        self._slide(pieces['left'], pieces['center'],
                (self.piece_size, 0))
    self.check_status()
```

```
def _slide(self, from_, to, coord):
    self.canvas.move(from_['id'], *coord)
    to['pos_a'], from_['pos_a'] = from_['pos_a'], to['pos_a']
    self.steps += 1

def get_pieces_around(self):
    pieces = {'center': None,
              'right' : None,
              'left'  : None,
              'top'   : None,
              'bottom': None}
    for piece in self.board:
        if not piece['visible']:
            pieces['center'] = piece
            break
    x0, y0 = pieces['center']['pos_a']
    for piece in self.board:
        x1, y1 = piece['pos_a']
        if y0 == y1 and x1 == x0 + 1:
            pieces['right'] = piece
        if y0 == y1 and x1 == x0 - 1:
            pieces['left'] = piece
        if x0 == x1 and y1 == y0 - 1:
            pieces['top'] = piece
        if x0 == x1 and y1 == y0 + 1:
            pieces['bottom'] = piece
    return pieces

def create_board(self):
    self.board = []
```

```
    for x in range(self.board_grid):
        for y in range(self.board_grid):
            x0 = x * self.piece_size
            y0 = y * self.piece_size
            x1 = x0 + self.piece_size
            y1 = y0 + self.piece_size
            image = ImageTk.PhotoImage(
                    self.image.crop((x0, y0, x1, y1)))
            piece = {'id'     : None,
                     'image'  : image,
                     'pos_o'  : (x, y),
                     'pos_a'  : None,
                     'visible': True}
            self.board.append(piece)
    self.board[-1]['visible'] = False

def check_status(self):
    for piece in self.board:
        if piece['pos_a'] != piece['pos_o']:
            return
    title = 'Ganaste!'
    message = 'Lo resolviste en %d movidas!' % self.steps
    tkMessageBox.showinfo(title, message)

def show(self):
    random.shuffle(self.board)
    index = 0
    for x in range(self.board_grid):
        for y in range(self.board_grid):
            self.board[index]['pos_a'] = (x, y)
            if self.board[index]['visible']:
```

```
                    x1 = x * self.piece_size
                    y1 = y * self.piece_size
                    image = self.board[index]['image']
                    id = self.canvas.create_image(
                          x1, y1, image=image, anchor=tk.NW)
                    self.board[index]['id'] = id
                index += 1


    if __name__ == '__main__':
        from optparse import OptionParser
        parser = OptionParser(description="Sliding puzzle")
        parser.add_option('-g', '--board-grid', type=int, default=4,
                          help="(the minimum value is 3)")
        parser.add_option('-i', '--image', type=str, default=a,
                          help="path to image")
        args, _ = parser.parse_args()

        if args.board_grid < 3:
            args.board_grid = 3
            print("Warning: using 3 for board-grid")

        app = Application(args.image, args.board_grid)
        app.master.title('Sliding puzzle')
        app.mainloop()
elif args == 4:
    import random
    import tkinter as tk
    import tkinter.messagebox
    from PIL import Image, ImageTk
    import random
```

```
#tk.geometry(1000x1000)
MAX_BOARD_SIZE = 600
files=['sarojini.jpeg','bhagat.jpeg','azad.jpeg','lala.jpeg','subhash.jpeg']
a=random.choice(files)
class Application(tk.Frame):

    def __init__(self, image, board_grid=4):
        tk.Frame.__init__(self)
        self.grid()
        self.board_grid = board_grid if board_grid > 2 else 3
        self.load_image(image)
        self.steps = 0
        self.create_widgets()
        self.create_events()
        self.create_board()
        self.show()

    def load_image(self, image):
        image = Image.open(image)
        board_size = min(image.size)
        if image.size[0] != image.size[1]:
            image = image.crop((0, 0, board_size, board_size))
        if board_size > MAX_BOARD_SIZE:
            board_size = MAX_BOARD_SIZE
            image = image.resize((board_size, board_size), Image.ANTIALIAS)
        self.image = image
        self.board_size = board_size
        self.piece_size = self.board_size / self.board_grid

    def create_widgets(self):
        args = dict(width=self.board_size, height=self.board_size)
```

```
        self.canvas = tk.Canvas(self, **args)
        self.canvas.grid(row=0,column=0)

        #self.canvas = tk.Canvas(width=300, height=300, bg='white')
        #self.canvas.grid()

        load = Image.open(a)
        render = ImageTk.PhotoImage(load)

        widget = tk.Label(self, image=render, fg='white', bg='black')
        widget.image = render
        widget.place(x=0,y=0)
        widget.grid(row=0,column=1)
        #self.canvas.create_window(100, 100, window=widget)

    def create_events(self):
        self.canvas.bind_all('<KeyPress-Up>', self.slide)
        self.canvas.bind_all('<KeyPress-Down>', self.slide)
        self.canvas.bind_all('<KeyPress-Left>', self.slide)
        self.canvas.bind_all('<KeyPress-Right>', self.slide)
        self.canvas.bind_all('<KeyPress-h>', self.slide)
        self.canvas.bind_all('<KeyPress-j>', self.slide)
        self.canvas.bind_all('<KeyPress-k>', self.slide)
        self.canvas.bind_all('<KeyPress-l>', self.slide)
        self.canvas.bind_all('<KeyPress-H>', self.help)

    def help(self, event):
        if getattr(self, '_img_help_id', None) is None:
            self._img_help = ImageTk.PhotoImage(self.image)
            self._img_help_id = self.canvas.create_image(0, 0,
                    image=self._img_help, anchor=tk.NW)
```

```
        else:
            state = self.canvas.itemcget(self._img_help_id, 'state')
            state = 'hidden' if state == '' else ''
            self.canvas.itemconfigure(self._img_help_id, state=state)

    def slide(self, event):
        pieces = self.get_pieces_around()
        if event.keysym in ('Up', 'k') and pieces['bottom']:
            self._slide(pieces['bottom'], pieces['center'],
                        (0, -self.piece_size))
        if event.keysym in ('Down', 'j') and pieces['top']:
            self._slide(pieces['top'], pieces['center'],
                        (0, self.piece_size))
        if event.keysym in ('Left', 'h') and pieces['right']:
            self._slide(pieces['right'], pieces['center'],
                        (-self.piece_size, 0))
        if event.keysym in ('Right', 'l') and pieces['left']:
            self._slide(pieces['left'], pieces['center'],
                        (self.piece_size, 0))
        self.check_status()

    def _slide(self, from_, to, coord):
        self.canvas.move(from_['id'], *coord)
        to['pos_a'], from_['pos_a'] = from_['pos_a'], to['pos_a']
        self.steps += 1

    def get_pieces_around(self):
        pieces = {'center': None,
                  'right' : None,
                  'left'  : None,
                  'top'   : None,
```

```
            'bottom': None}
    for piece in self.board:
        if not piece['visible']:
            pieces['center'] = piece
            break
    x0, y0 = pieces['center']['pos_a']
    for piece in self.board:
        x1, y1 = piece['pos_a']
        if y0 == y1 and x1 == x0 + 1:
            pieces['right'] = piece
        if y0 == y1 and x1 == x0 - 1:
            pieces['left'] = piece
        if x0 == x1 and y1 == y0 - 1:
            pieces['top'] = piece
        if x0 == x1 and y1 == y0 + 1:
            pieces['bottom'] = piece
    return pieces


def create_board(self):
    self.board = []
    for x in range(self.board_grid):
        for y in range(self.board_grid):
            x0 = x * self.piece_size
            y0 = y * self.piece_size
            x1 = x0 + self.piece_size
            y1 = y0 + self.piece_size
            image = ImageTk.PhotoImage(
                    self.image.crop((x0, y0, x1, y1)))
            piece = {'id'     : None,
                     'image'  : image,
                     'pos_o'  : (x, y),
```

```
                    'pos_a'  : None,
                    'visible': True}
            self.board.append(piece)
        self.board[-1]['visible'] = False

    def check_status(self):
        for piece in self.board:
            if piece['pos_a'] != piece['pos_o']:
                return
        title = 'Ganaste!'
        message = 'Lo resolviste en %d movidas!' % self.steps
        tkMessageBox.showinfo(title, message)

    def show(self):
        random.shuffle(self.board)
        index = 0
        for x in range(self.board_grid):
            for y in range(self.board_grid):
                self.board[index]['pos_a'] = (x, y)
                if self.board[index]['visible']:
                    x1 = x * self.piece_size
                    y1 = y * self.piece_size
                    image = self.board[index]['image']
                    id = self.canvas.create_image(
                            x1, y1, image=image, anchor=tk.NW)
                    self.board[index]['id'] = id
                index += 1


if __name__ == '__main__':
    from optparse import OptionParser
```

```
            parser = OptionParser(description="Sliding puzzle")
            parser.add_option('-g', '--board-grid', type=int, default=4,
                              help="(the minimum value is 3)")
            parser.add_option('-i', '--image', type=str, default=a,
                              help="path to image")
            args, _ = parser.parse_args()

            if args.board_grid < 3:
               args.board_grid = 3
               print("Warning: using 3 for board-grid")

            app = Application(args.image, args.board_grid)
            app.master.title('Sliding puzzle')
            app.mainloop()

    elif args == 5:
        from Options import Shapes
        Shapes.main()
    elif args == 6:
        from Options import Vegetable
        Vegetable.main()
    elif args == 7:
        from Options import Vehicles
        Vehicles.main()

def option():

    lab_img1 = Button(
        main_window,
        image=img1,
        bg='#FF69B4',
```

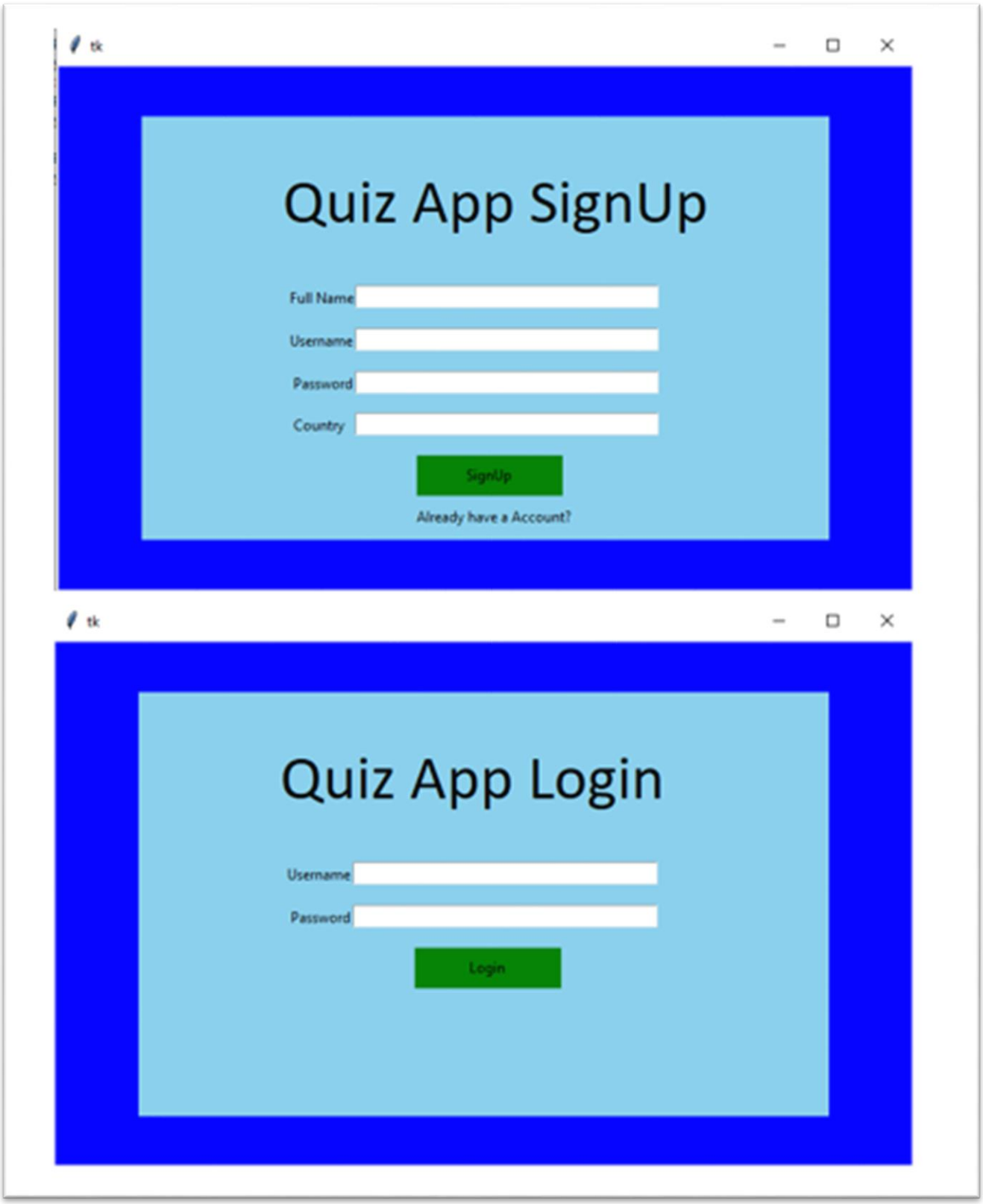
```
    border=0,
    justify='center',

)
sel_btn1 = Button(
    text="National symbols",
    width=18,
    borderwidth=8,
    font=("", 18),
    fg="#000000",
    bg="#DDA0DD",
    cursor="hand2",
    command=lambda: start_game(1),
)

sel_btn2 = Button(
    text="Historical rulers",
    width=18,
    borderwidth=8,
    font=("", 18),
    fg="#000000",
    bg="#DDA0DD",
    cursor="hand2",
    command=lambda: start_game(2),
)

sel_btn3 = Button(
    text="monuments",
    width=18,
    borderwidth=8,
    font=("", 18),
```

```
    fg="#000000",
    bg="#DDA0DD",
    cursor="hand2",
    command=lambda: start_game(3),
)

sel_btn4 = Button(
    text="Freedom fighters",
    width=18,
    borderwidth=8,
    font=("", 18),
    fg="#000000",
    bg="#DDA0DD",
    cursor="hand2",
    command=lambda: start_game(4),
)

sel_btn5 = Button(
    text="Compounds",
    width=18,
    borderwidth=8,
    font=("", 18),
    fg="#000000",
    bg="#99ffd6",
    cursor="hand2",
    command=lambda: start_game(5),
)

sel_btn6 = Button(
    text="Acids,bases and salts",
    width=18,
```

```
        borderwidth=8,
        font=("", 18),
        fg="#000000",
        bg="#99ffd6",
        cursor="hand2",
        command=lambda: start_game(6),
    )
    lab_img1.grid(row=0, column=0, padx=20)
    sel_btn1.grid(row=1, column=1, pady=(10, 0), padx=50, )
    sel_btn2.grid(row=2, column=1, pady=(10, 0), padx=50, )
    sel_btn3.grid(row=3, column=1, pady=(10, 0), padx=50, )
    sel_btn4.grid(row=4, column=1, pady=(10, 0), padx=50, )
    #sel_btn5.grid(row=5, column=1, pady=(10, 0), padx=50, )
    #sel_btn1.grid(row=6, column=1, pady=(10, 0), padx=50, )
    #sel_btn7.grid(row=6, column=1, pady=(10, 0), padx=50, )

def show_option():
    start_btn.destroy()

    lab_img.destroy()
    option()

main_window = Tk()

main_window.geometry("500x500+400+100")
main_window.resizable(0, 0)
main_window.title("PUZZLE MANIA")
main_window.configure(background="#FF69B4")
#main_window.iconbitmap(r'quizee_logo_.ico')

img0 = PhotoImage(file="puzzle_cover.png")
```

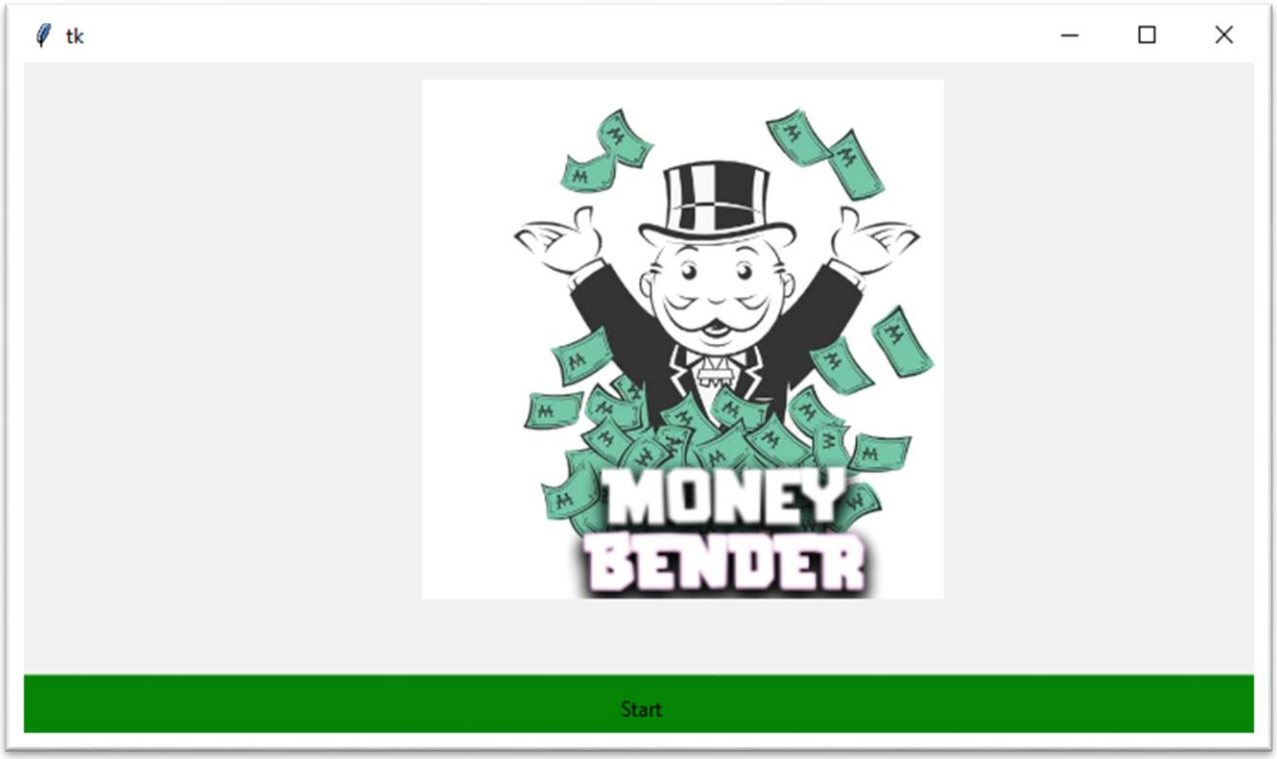
```
    img1 = PhotoImage(file="back.png")

    lab_img = Label(
        main_window,
        image=img0,
        bg='#FF69B4',
    )
    lab_img.pack(pady=(45, 0))

    start_btn = Button(
        main_window,
        text="Start",
        width=18,
        borderwidth=8,
        fg="#000000",
        bg="#DDA0DD",
        font=("", 13),
        cursor="hand2",
        command=show_option,
    )
    start_btn.pack(pady=(40, 20))
    main_window.mainloop()
start_main_page()
```

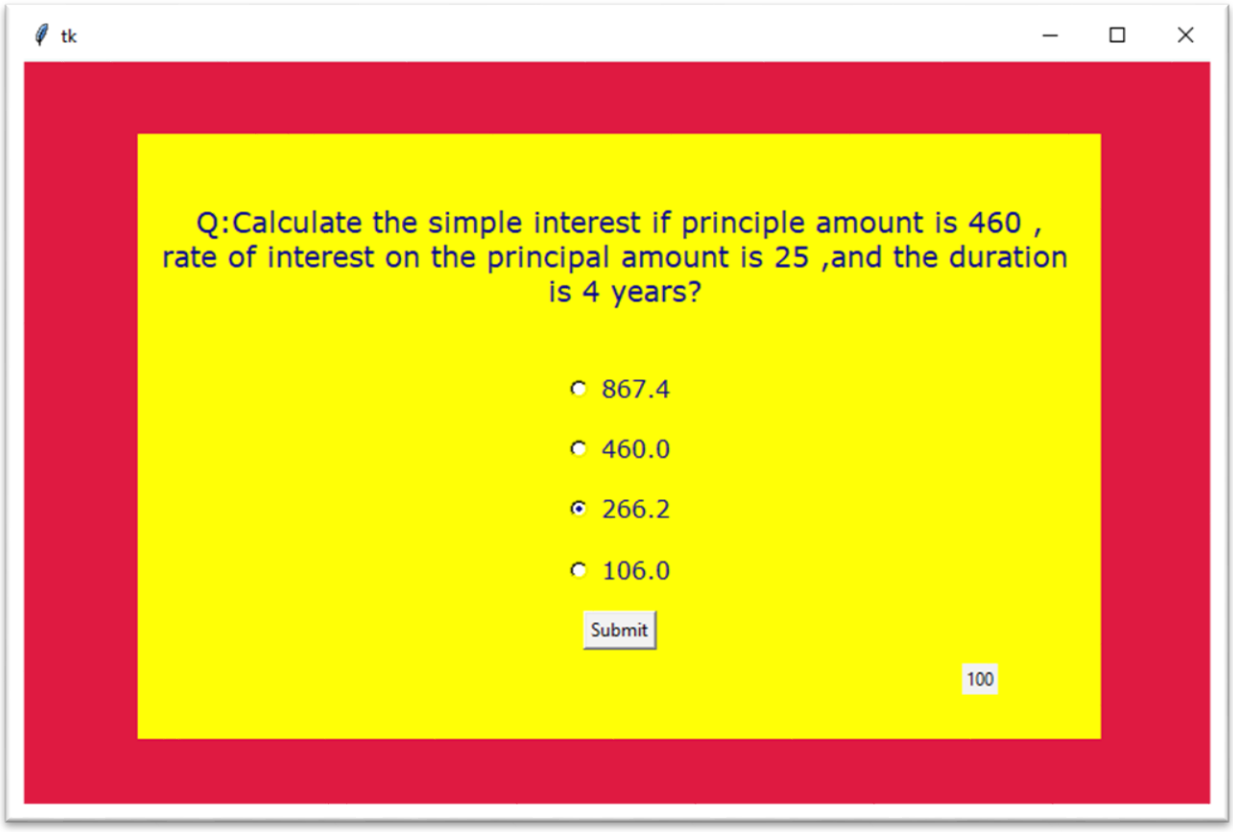
OUTPUT

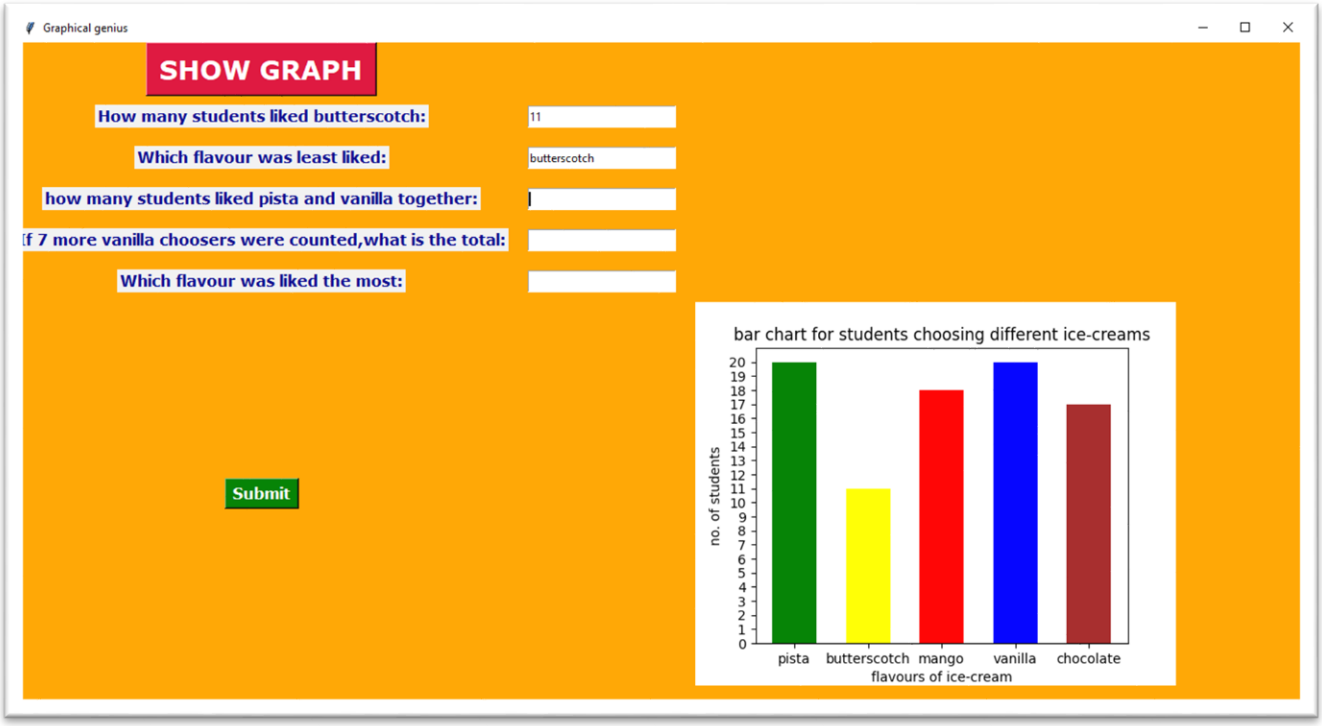

tk
Quiz App SignUp
Full Name
Username
Password
Country
SignUp
Already have a Account?
tk
Quiz App Login
Username
Password
Login

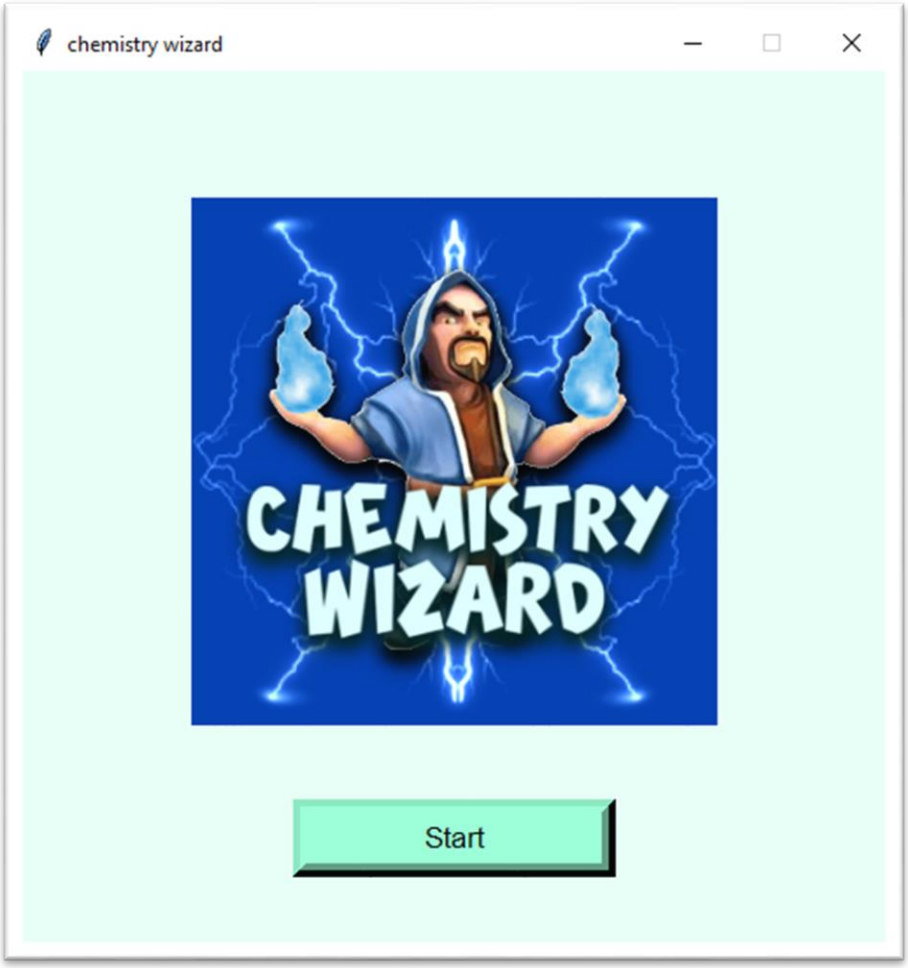

tk
WELCOME TO FUNBOOK!

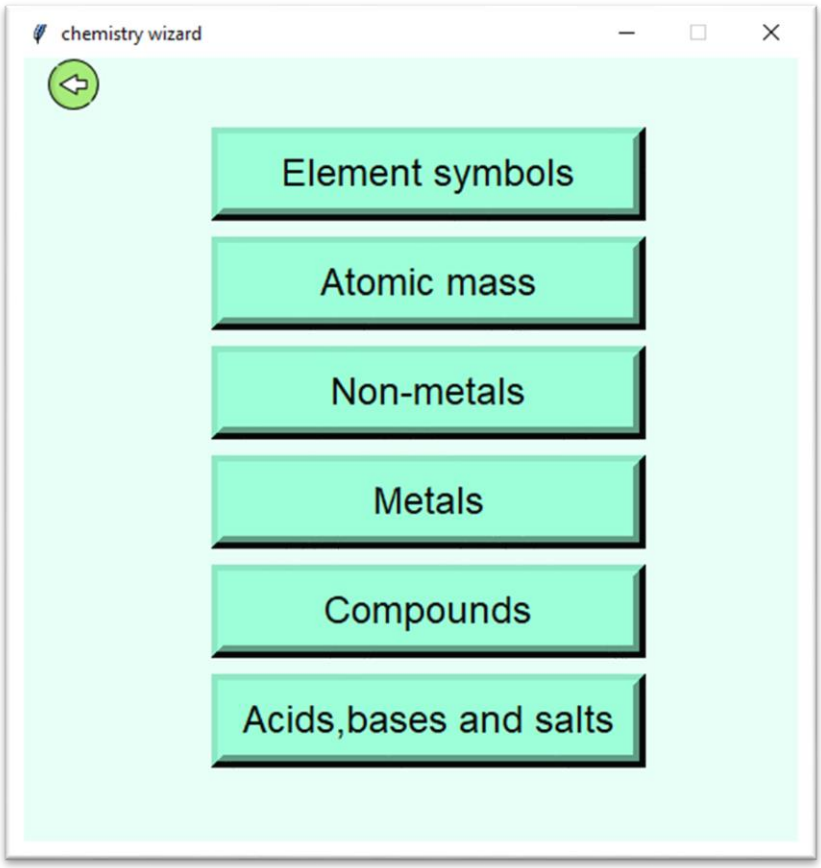

Python Turtle Graphics
Find 5/(-2)=?
Your score:30
-2.5
69
-14
7
32
126

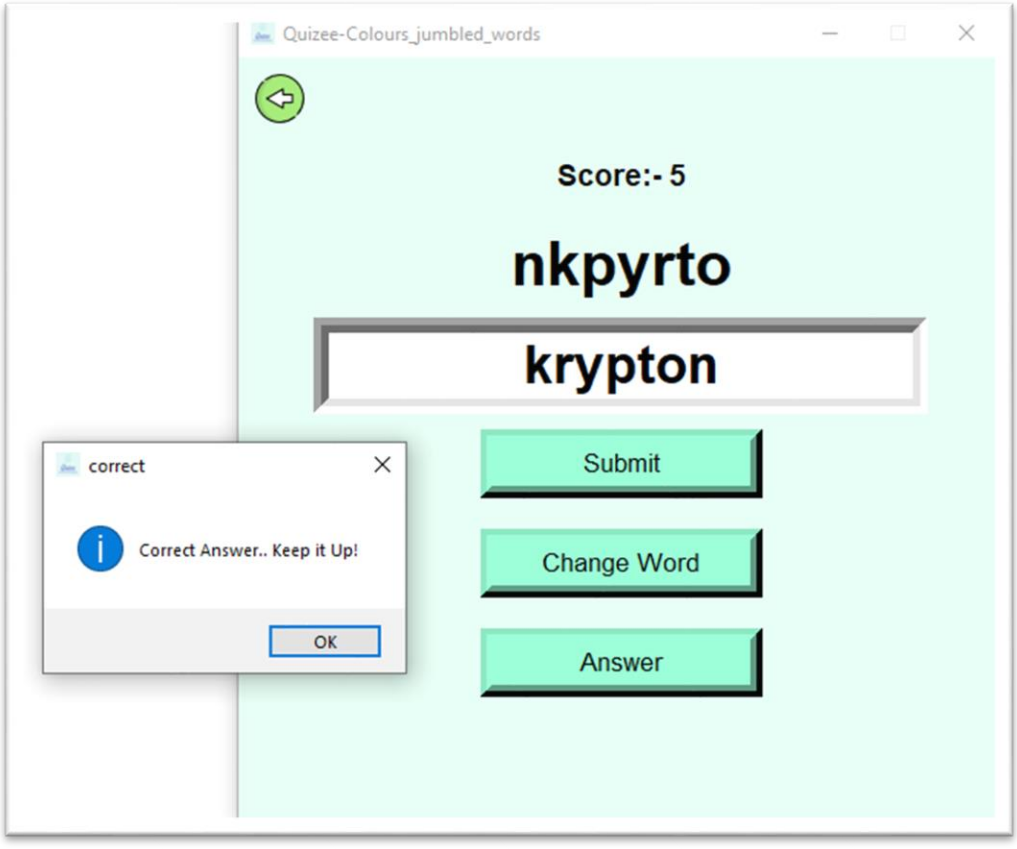

tk
MONEY
BENDER
Start

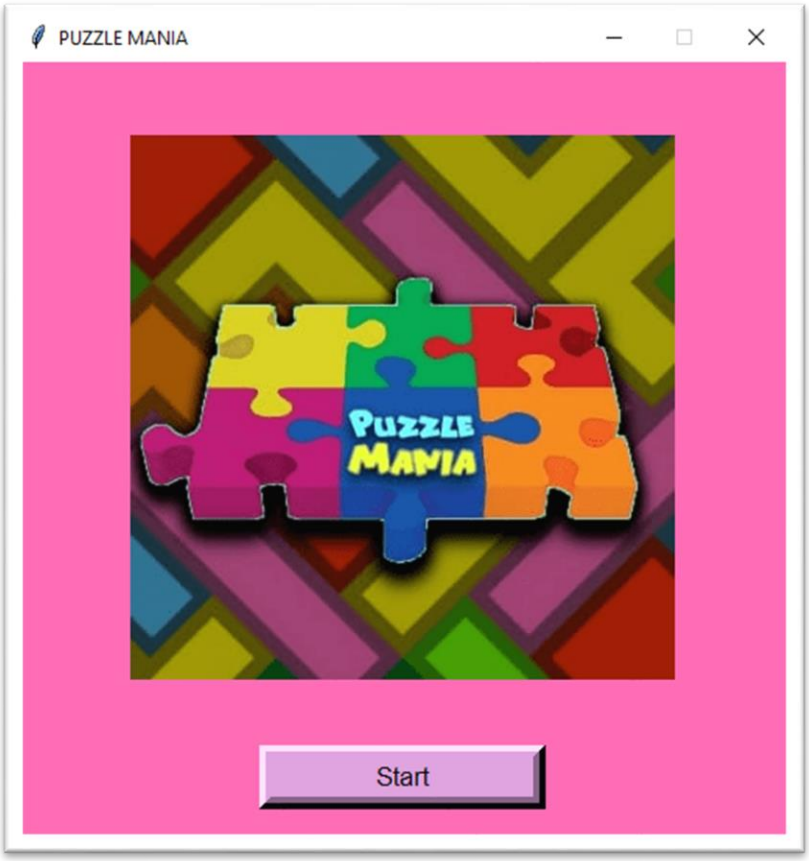

tk
Q:Calculate the simple interest if principle amount is 460 , rate of interest on the principal amount is 25 ,and the duration is 4 years?
867.4
460.0
266.2
106.0
Submit
100


Graphical genius
SHOW GRAPH
How many students liked butterscotch:
11
Which flavour was least liked:
butterscotch
how many students liked pista and vanilla together:
If 7 more vanilla choosers were counted,what is the total:
Which flavour was liked the most:
Submit
bar chart for students choosing different ice-creams
no. of students
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
pista
butterscotch
mango
vanilla
chocolate
flavours of ice-cream

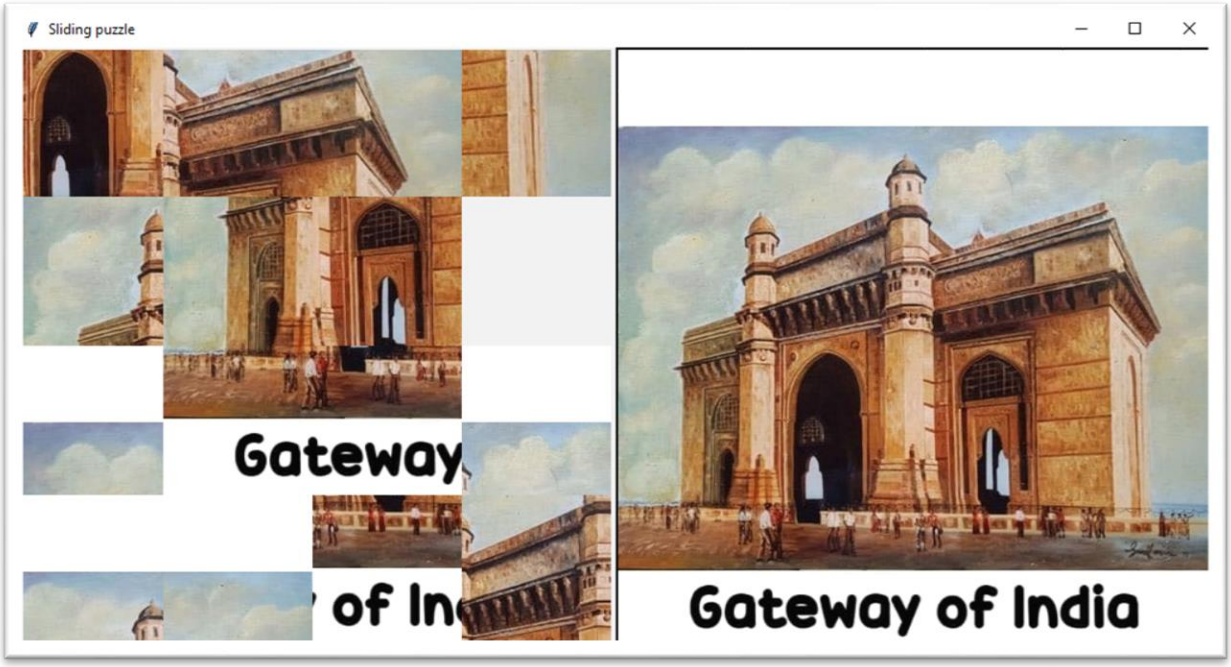

chemistry wizard
CHEMISTRY
WIZARD
Start

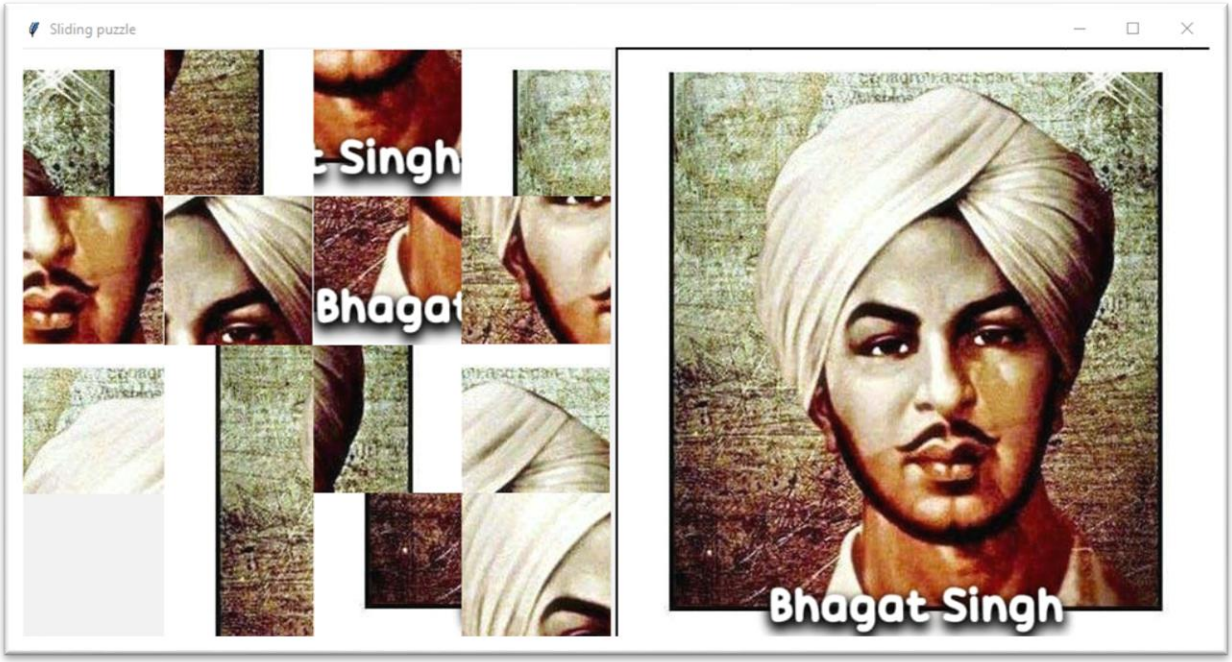

chemistry wizard
Element symbols
Atomic mass
Non-metals
Metals
Compounds
Acids,bases and salts


Quizee-Colours_jumbled_words
Score:- 5
nkpyrto
krypton
Submit
Change Word
Answer
correct
Correct Answer.. Keep it Up!
OK


PUZZLE MANIA
PUZZLE
MANIA
Start


PUZZLE MANIA
National symbols
Historical rulers
monuments
Freedom fighters


Sliding puzzle
Gateway
of In
Gateway of India


Sliding puzzle
t Singh
Bhagat
Bhagat Singh

FUTURE SCOPE

- For teachers, video games with educational value will act as relevant material for engaging their students.
- The games allow students to learn new concepts at their own pace without having a constant overlook from parents and teachers.
- In the future, once the covid pandemic goes and the normal scenario resumes the students will most probably get a lot of homework, which involves the tedious process of writing pages after pages.

Thus, if some of the homework is given through our project, then the student will not only be interested in doing his homework on this platform, but will willingly do it .

BIBLIOGRAPHY

www.canva.com

www.geeksforgeeks.org

www.stackoverflow.com

www.github.com