# Programming Project

Module Code : FC712

Class/Group: COMP GROUP 1

Module Title: Programming

Assessment Title: Programming Project

Tutor Name: Jack Parkinson

Student GUID Number: 2785448

Date of Submission: 17 July 2022

I confirm that this assignment is my own work.

Where I/we have referred to academic sources, I have provided in-text citations and included the sources in the final reference list.

Contents

[Programming Project 1](#_Toc108130848)

[Section 1.1 3](#_Toc108130849)

[Section 1.2 4](#_Toc108130850)

[Section 1.3 5](#_Toc108130851)

[Section 1.4 6](#_Toc108130852)

[Section 2.1 7](#_Toc108130853)

[Section 2.2 8](#_Toc108130854)

[Section 2.3 9](#_Toc108130855)

[Section 2.4 10](#_Toc108130856)

[Section 3.1 11](#_Toc108130857)

[Section 3.2 19](#_Toc108130858)

[References 66](#_Toc108130859)

## Section 1.1

Diagram

Description automatically generated

Diagram

Description automatically generated

## Section 1.2

The program is made by python and flask for validating and creating webpages. Including HTML and CSS forms for designing webpages. Each form has different attribute, each validates different type of data. This includes, radio fields, Boolean fields, text area fields as well as string fields. Each of these validations are made for specific webpages, in functions. The functions are made in a thing called app routes; these app routes will connect the location of the webpages. As well as connect each python program to the webpages and the HTML pages. The HTML program is stored in templates and connected to the python program by a method called render templates. There is also a function to store the time the data has been inputted into a text file alongside the answers. Each time a function in the app route gets validated then at that exact moment the data will be taken and stored in a text file, these can be answers from a radio field or text area fields or string fields. As well as once the data gets validated then the webpage will direct you to another webpage using “Url\_for”. The Boolean fields are only used for terms and conditions. The program also has a security key for secure reasons as to not get malware or anything. To access each route, you will need a request called “POST”, again this is also for security reasons. The pages will keep redirecting the user until the user has reached the final page where the survey has finally been recorded and finished. The webpages are designed using mainly CSS for background images and placing and HTML to place labels and more. Jinja is also used to add python code to HTML forms so that validation can occur. The text area field validation has an extra validation inside it called length, this part validates how many characters the user has inputted. There is a minimum and maximum value to the validation and an error will occur if the user has surpassed or hasn’t reached the criteria. Adding to that the text area field can be customised from the python program as shown by the variable “render\_kw”. This variable customises the text area field big box. The Boolean field only validates if a box has been ticked. The radio field has multiple options to tick the answer from compared to the Boolean field. An error will occur if none of the boxes has been ticked in either a Boolean field or radio field. The string field only validates if data has been inputted in the box.

## 

## Section 1.3

I had to use a method called trial and error as well as debugging to fix issues in the program. I also had to use google to fix many issues, however the internet was not as suitable to get information to fix issues in flask as most of the information has been outdated. Therefore, I had to use trial and error which was very time consuming. In this process I had to try new things out for the program to work, this led to much more errors than predicted. I also had to make functions for each page to check weather each function would work properly. Additionally, I had to keep erasing data in files as I kept on testing the website for it to function in a suitable manner. Another time-consuming factor was always checking on HTML and CSS forms to design the website, this meant I had to keep repeating the program to see weather the design was nice. This required me to save and restart the program. A problem had also raised where the CSS would not load unless I had erased my cache, this took too much of my time as I had to reset my program and erase my cache to run 1 part of the program. I also had to apply jinja to my code which was very confusing, and this had let to the structure of the code becoming much more confusing. Multiple files were created to test each program before applying to the main file. The results at the end were successful as the file had run properly as a nice, designed survey, and the validations were also successful despite having main issues with jinja.

## Section 1.4

The first web page will ask the user their GIC number, email, and name. If the user does not type a single data to either or any of the fields, a validation will occur, and the user wouldn’t be able to proceed to the next page as an error will show up. Additionally, if the user doesn’t provide the correct format of email or the correct format of P number a error will show up. However, if the user types of data in all the fields, then they would be able to proceed and the information as well as the time will be stored in a separated text file. The user will then go to the next page where they would have to accept the terms and conditions and the Boolean validator will appear for that to happen, if the user does not tick the box, an error will be presented. But if they tick the box, they will proceed to the radio field part of the program where multiple questions with answers will be presented. Once the user has answered the questions, it would be stored in another separate file, and they can move to the next webpage. If they don’t answer the questions fully an error will once again be presented. The next webpage has 2 questions which are essay type of questions, meaning text area field validators are presented in this part. If the user does not type the required number of characters in the field or surpass the maximum allowed characters, then another error would be presented. But if the user has satisfied the validation the data will be stored in a separate file, and they would move on to the final webpage. In the final webpage, few words are presented to thank the user for participating in the survey. I believe I had done well in implementing the validations successfully into my program. Additionally, I believe the survey looks very nice and it’s working properly like a normal questionnaire would. However, an improvement I would make is to make the data be erased in the file if the user does not accept the terms and conditions as well as validate the email properly so a false email will not be provided.

## Section 2.1

Diagram

Description automatically generatedDiagram

Description automatically generated

Diagram

Description automatically generated

Chart

Description automatically generated

## Section 2.2

The foundation of the python code is made by Tkinter. Some of the image’s act as button as shown in the starting screen itself, the images are also resized to fit the starting screen. The loading screen function has a background timer with the key variable as timer, the timer here is set to 5 seconds and there are labels to show the user that the screen is loading. The first question page is one without exceptions. This means that no error will be shown no matter the input that is entered or sent. The other variety has exceptions for questions which require integer data type as an answer. If the user does not type an integer as a data type the question will keep on presenting the error until the timer runs out. Each question page contains a label as well as a variable known as “amount\_of\_points”. This variable changes throughout the program if the user gets the correct answer. Additionally, each question has a 15 second background timer, if the user does not get the answer correct in time, then the user would be led to a function called the wrong tab function. This is where a screen displaying “You’re wrong” is shown. However, if the user gets the correct answer a screen will be displayed showing “You’re correct”. The questions are taken in a list of dictionaries and such information that goes to the dictionaries will come from the question files. Answers will be checked at the function called “answer\_check”, at this part the answers from another text file will be taken as data into a list of dictionaries and this list will be used as a method to check the answer for each question. In the earlier stages, a list was also used to check if the user has inputted a proper username. The list included key alphabets to check whether the users name would be suitable. Adding to that a fake text is included in almost any entry box. This fake text would present itself to the user to show the location to write the answers. There are also buttons to move between each page or tabs. Another key function is the factorial function, here a user is allowed to type a number to do factorial calculations. Once typed the user will have to solve it by themselves and write the answer, the program will also solve it by itself and check the answer. The program will solve it using a recursive function. The final key function is the leader board function, here the number of points is accumulated and stored in a text file. The data will be taken from the text file and appended to a list and sorted out into ascending to descending order. The data will then be displayed in labels for the top 5 highest scores.

## Section 2.3

The program was tested and fixed with a method called debugging. This method takes a slow approach to the program however could fix issues in a big program. I would make functions for each tab and test it before going to the next function. If an error was presented within the functions, I would check which areas has errors and try to fix them. Sometimes I would need to google search on how to solve such problems. Since this was a very big code, many errors were presented in each of the functions. Therefore, it was time consuming to fix each bit of code. Each image also had to be resized so I would have to run the program each time to see If the images could fit the screen. Additionally, I had to fit each label and button in the right positioning to make it look like a nice, presented game, as well as the background colours and more. Timers also had bugs sometimes, so I had to fix each timer to work. Data is written to the files, therefore with each test I had to erase data in files as well as save the file. I also had to search for the suitable questions as well as photos to go along such questions. Each exception should also appear at the right questions, this made it tough and hard to apply in the program as each exception came at the end of the functions. Multiple files of program were made to test each code before applying it to the main code file to be presented. The results at the end were successful as all criteria were met as well as the program was presented like an actual quiz game.

## Section 2.4

The program is called Kahood. It is a type of quiz game where the user gets to choose what type of quiz, he/she would want and then try to answer the questions correctly. It is a single player game, and the user results will be stored if the user doesn’t close the program. The user can type their own username and it would appear at the top of the screen. The program has 2 categories called anime and factorial. In the factorial category the user would get to type his own input and the program will find the factorial of the input. The user will then have to input the answer to the factorial of their own input. If the user gets it right then a correct page will show up, or else a wrong page will appear stating they got it wrong. The user will then have a choice to restart the quiz or quit. The anime category has a similar approach except the program will provide questions and award points for each questions the user had gotten correct. The points are also stored and can be viewed at the top of the screen. It will also be displayed in the leader board after the quiz. There are a total of 5 questions in the anime section of the quiz, after 5 questions the leader board will show up. The leader board will show the users points in ascending order. I believe I had met the criteria to make the program and have made a good quiz program that works. It has an answering system with categories just like other quiz games. However, I believe it can be fixed by being better structured as well as having nice colours as a quiz. I also believe the program can be sorted in a better manner.

## Section 3.1

#Importing all the modules to access the features

from time import strftime

from flask import Flask, render\_template, url\_for, redirect, flash,request

from wtforms import Form, StringField, TextAreaField, validators, SubmitField, BooleanField,RadioField, EmailField

from wtforms.validators import InputRequired, Length, DataRequired, Email

from wtforms.widgets import TextArea

app = Flask(\_\_name\_\_)

app.config.from\_object(\_\_name\_\_)

app.config['SECRET\_KEY'] = 'JDaodfhiuowefndsfnsfnsdnfsidofnsdUHIfdhsdf' #Secret key for safety purposes

#first part of flask

class The\_first\_form(Form):#This part gets the name, email and GIC number given by the user. It validates each entrybox to see weather any data has been submitted. If none has an error will pop up.

name = StringField('Name:', validators=[validators.DataRequired()])#This is all possible by the work of Forms

GIC = StringField('GIC', validators=[validators.DataRequired(),Length(min=7, max=7, message="At least 7 characters")])#This part checks if the P number length is correct, and provides a error if the P number is wrong in terms of length

email = EmailField('email:', validators=[validators.DataRequired(),Email(message="Please enter a valid email addresss")])#This parrt validates if the email is correctly written using validation specifically for email

def current\_time():#This part gets the time when the user submits the data.

time\_get = strftime("%Y-%m-%dT%H:%M")

return time\_get

def write\_to\_file(name, GIC, email):#This part writes to file the name, time, GIC number and email in a certain format.

store\_data = open('GIC\_details.txt', 'a')#This part opens the file to write

time\_store = current\_time()

store\_data.write('DateStamp={}, Name={}, GIC number={}, Email={} \n'.format(time\_store, name, GIC, email))#format is over here

store\_data.close()#This part closes the file

#second part of flask

class Second\_form(Form):#This part validates that the user has ticked the checkbox, if the user doesnt a error will pop up. This is done by boolean validators.

accept = BooleanField('accept', validators=[validators.DataRequired()])

#third part of flask

class Third\_form(Form):#Radio fields are added here to have multiple choice questions. In total there are 4 questions with the choices of yes or no. As well as each yes data is connected to a yes data that will be written to the file.

question = RadioField('Label',validators=[InputRequired()],choices=[("yes", 'yes'), ("no", 'no')], validate\_choice=True)#Each choice will have to validate true, the following four lines of code was taken from https://stackoverflow.com/questions/53101631/wtforms-radiofield-preventing-form-validation

question1 = RadioField('Label',validators=[InputRequired()],choices=[("yes", 'yes'), ("no", 'no')], validate\_choice=True)

question2 = RadioField('Label',validators=[InputRequired()],choices=[("yes", 'yes'), ("no", 'no')], validate\_choice=True)

question3 = RadioField('Label',validators=[InputRequired()],choices=[("yes", 'yes'), ("no", 'no')], validate\_choice=True)

def write\_to\_file1(question, question1, question2, question3):#This part takes the answer from the radiofields(4 answers) and stores them in a file, formatted way just like the file above.

store\_data1 = open('ans1.txt', 'a')#This part opens the file

store\_data1.write('question={}, Q1={}, Q2={}, Q3={} \n'.format(question, question1, question2, question3))#This is the format

store\_data1.close()#This part closes the file

#final part of flask

class Fourth\_form(Form):#This area of the flask uses Textareafield. such a field enables us to write alot of words in a box. The box can be configurated by rows and columns. The maximum character and minimum will be set. If the user doesnt meet the max and min values, there will be a error.

question4 = TextAreaField('question4', validators=[DataRequired(), Length(min=15, max=250, message="At least 15 characters, maximum 250")], render\_kw={"rows": 10, "cols": 80}) #The following 2 lines of code was taken from https://stackoverflow.com/questions/7979548/how-to-render-my-textarea-with-wtforms

question5 = TextAreaField('question5', validators=[DataRequired(), Length(min=15, max=250, message="At least 15 characters, maximum 250")], render\_kw={"rows": 10, "cols": 80})

def write\_to\_disk2(question4, question5):#The answers again are stored in another file. Seperately just like above.

store\_data1 = open('ans2.txt', 'a')

store\_data1.write('Q4={}, Q5={} \n'.format(question4, question5))

store\_data1.close()

@app.route('/home', methods =['GET','POST']) #Set the route of the app and the method to access it is POST, this is to be safer.

def start1():

form = The\_first\_form(request.form) #the form here is assigned to the first form to help request form in shorter way

if request.method == 'POST': #if method is post then we would request a form for name, email and Gic in html.

name = request.form['name']

GIC = request.form['GIC']

email = request.form['email']

if form.validate(): #If the form validates using the validators we will write the information in the file and go to a different url

write\_to\_file(name, GIC, email)

return redirect(url\_for('check'))

else: #if the form doesnt validate there will be a error popping up

flash('Error: please enter a valid email adress')

return render\_template('index.html', form=form) #render template will take the html form from the template folder and use it for website building.

@app.route('/check', methods =['GET','POST']) #Set the route of the app and the method to access it is POST, this is to be safer.

def check():

form = Second\_form(request.form)

if request.method == 'POST':#if method is post then we would instantly go to the validation process. We dont need to store data for accepting terms and conditions.

if form.validate():#This would validate if the box has been ticked, if it has we would go to url question1

return redirect(url\_for('questions1'))

else: #or else a error will pop up

flash('Error: please tick the box')

return render\_template('index1.html', form=form)#This is another html form in the same template

@app.route('/questions1', methods =['GET','POST']) #Set the route of the app and the method to access it is POST, this is to be safer.

def questions1():

form = Third\_form()

if request.method == 'POST': #if method is post then we would request a form for the 4 questions assigned here.

question = request.form['question']

question1 = request.form['question1']

question2 = request.form['question2']

question3 = request.form['question3']

if form.validate(): #same thing as above, if one of the radiofields isnt ticked then an error will be flashed

flash('Error: A box must be ticked')

else: #or else the answers will be stored and the user will move forward to the next file

write\_to\_file1(question, question1, question2, question3)

return redirect(url\_for('questions2'))

return render\_template('index2.html', form=form) #This is a different html file in the same folder template

@app.route('/questions2', methods =['GET','POST']) #Set the route of the app and the method to access it is POST, this is to be safer.

def questions2():

form = Fourth\_form()

if request.method == 'POST':#if method is post then we would request a form for the 2 questions assigned here.

question4 = request.form['question4']

question5 = request.form['question5']

if form.validate(): # if the textfield has input but doesnt meet the validation requirement a error will be supplied

flash('Error: a input must be filled')

else:#or else answer will be stored and the user will go to the final page

write\_to\_disk2(question4, question5)

return redirect(url\_for('end'))

return render\_template('index3.html', form=form)#its a different html form but in the same template

@app.route('/end', methods =['GET','POST']) #Set the route of the app and the method to access it is POST, this is to be safer.

def end():#This is the final route where its a thank you page.

return render\_template('index4.html')#its a different html form but the same template

if \_\_name\_\_ == '\_\_main\_\_':

app.run() #This is for the flask to run.

## Section 3.2

#imports so we can use the features

import tkinter

from tkinter import \*

import tkinter as tk

from PIL import Image,ImageTk

import time

window = tk.Toplevel() #the whole window here is assigned to tkinter

#function for username tab

def tab1():

label1.destroy() #destroy() enables us to destroy labels or antyhing placed down in tkinter

btn.destroy()

btn2.destroy()

window.configure(bg="#A44CD3")

label2 = tk.Label(window,text="KAHOOD!",fg="white",bg="#A44CD3") # This is a label that is being configurated and placed down.

label2.config(font=('Comic Sans MS','100','underline',"bold"))

label2.pack() # pack() allows the program to choose where to place your label or any variable it is assigned to place down.

def fake\_text€: # The following 4 lines of code was taken from <https://stackoverflow.com/questions/27820178/how-to-add-placeholder-to-an-entry-in-tkinter>

entrybox.delete(0,”end”)

entrybox = Entry(window, bg=”white”, width=20,font=(‘Arial 50’), borderwidth=10) #This is a entrybox, where we must input data.

Entrybox.insert(0, “enter your username”)

entrybox.pack(pady=200)

entrybox.bind(“<FocusIn>”, fake\_text) # This is basically a fake text, once the entrybox is clicked the text will isappear

btn\_back = Button(window,text=”quit”,command=window.destroy,fg=”white”,bg=”black”,) # This is a button which has a command to end the whole program if clicked

btn\_back.config(font=(‘Comic Sans MS’,’60’))

btn\_back.pack(side=LEFT,pady=3)

correction = Label(window,text=’’,bg=”#A44CD3”,fg=”yellow”) # This label is for future references, we will need to use it in the next function.

Correction.pack(pady=20)

#function for checking the username tab

def tab2():

entrybox1 = entrybox.get() # This allows us to get the input from the entrybox, basically what username the user has entered

allowed\_characters=['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','1','2','3','4','5','6','7','8','9','0','(',')','$','%','\_','/']

if any(x not in allowed\_characters for x in entrybox1) or len(entrybox1)==0 or len(entrybox1)>16: # This will check what has inputted in the entrybox to see weather or not its valid, if the length of the data is 0, or more than 16 or a character is not in the list above then a error message is shown.

correction.config(text="Please enter a username with valid characters or a username which doesnt exceed 16 characters.") #The correction from before has been configurated from before to show a error message.

else: #this block is follows if the users input doesnt match the validations listed above, this then would make a whole new page to display for the user. The page displayed here is the category page.

window.configure(bg="white")

btn\_back.destroy()

label2.destroy()

btn\_next.destroy()

correction.destroy()

entrybox.destroy()

label\_check = Label(window,text=entrybox1,bg="#A44CD3",fg="white",width=50,height=5)

label\_check.place(x=0, y=0)#the place() block enables you to exactly place down where you would want the label to be

Label\_brand = Label(window,text=”KAHOOD”,bg=”#A44CD3”,fg=”white”,width=50,height=5)

Label\_brand.place(x=1400,y=0)

Label3 = Label(window,text=””,bg=”#A44CD3”,fg=”white”,width=160,height=5).place(x=300,y=0)

Label\_cat = Label(window,text=”Categories”,fg=”blue”,bg=”white”)

Label\_cat.config(font=(‘Comic Sans MS’,’100’,’underline’))

Label\_cat.pack(padx=90,pady=90)

global photo1 #global basically enables to use photo1 here in other functions or outside the else statement.

Global resized\_photo2

global new\_image2

global photomath

global resized\_photomath

global new\_imagemath

photo1 = (Image.open(“anime.jpg”)) # This block opens a image to use further in the program, the following 5 lines of code has been taken from <https://www.codegrepper.com/code-examples/python/how+to+resize+image+in+python+tkinter>

resized\_photo2 = photo1.resize((500,300),Image. ANTIALIAS) # This block is to resize the image

new\_image2= ImageTk.PhotoImage(resized\_photo2)

photomath = (Image.open(“factorial.jpg”))

resized\_photomath = photomath.resize((500,300),Image. ANTIALIAS)

new\_imagemath= ImageTk.PhotoImage(resized\_photomath)

#This is the factorial page, first comes the rule page. The rules will be explained here. Using labels

def factorial\_page():

window.config(bg = "white")

btn\_factorial.destroy()

btn\_math.destroy()

Label\_cat.destroy()

Label\_rule = Label(window,text="Rules",fg="Red",bg="white")

Label\_rule.config(font=('Comic Sans MS','100','underline'))

Label\_rule.pack(padx=90,pady=90)

label\_rule1 = Label(window,text="-Type in a factorial of your choice",bg="white")

label\_rule1.config(font=('Comic Sans MS','30'))

label\_rule1.pack(padx=0,pady=0)

label\_rule2=Label(window,text="-Calculate it and then place the answer in the box",bg="white")

label\_rule2.config(font=('Comic Sans MS','30'))

label\_rule2.pack(padx=0,pady=0)

label\_rule4=Label(window,text="-You will have unlimited time to answer.",bg="white")

label\_rule4.config(font=('Comic Sans MS','30'))

label\_rule4.pack(padx=0,pady=0)

#Next part is where the user gets to choose any number they would like to get factorial. Only numbers or else a exception will be raised.

def factorial\_page2():

label\_rule4.destroy()

Label\_rule.destroy()

label\_rule1.destroy()

label\_rule2.destroy()

start\_quiz\_btn.destroy()

Label\_question1= Label(window,text="Whats your factorial?",bg="white")

Label\_question1.config(font=('Comic Sans MS','40'))

Label\_question1.pack(padx=100,pady=100)

def fake\_text(e):

answer.delete(0,"end")

answer = Entry(window,width=30,fg="red",bg="yellow",font=('Arial 50'))

answer.pack(padx=100,pady=100)

answer.insert(0, "enter your number here!")

answer.bind("<FocusIn>", fake\_text)

Label\_error= Label(window,text="Sorry please enter a number, not a alphabet or don't leave it blank",bg="white",font=('Arial 20'))

# This part the calculation will be done to collect the results of the factorial, it will also as the user to answer the factorial of the number he inputted.

def check\_ansfact():

answer.destroy()

Label\_question1.destroy()

Label\_error.destroy()

btn\_checkfact.destroy()

#This part takes the input of the user and if its 1 then the result would be 1. However if its not 1, a recursive loop will happen. Where the input would multiply with its own self and minus 1 each loop.

def factorial(r):

if r == 1:

return r

else:

return r\*factorial(r-1)

#The results will then be collected here. The user here will input the answer to his input factorial. A exception will be raised if the user doesnt input a number.

global result

result = factorial(r)

Label\_question2= Label(window,text="Whats your answer?",bg="white")

Label\_question2.config(font=('Comic Sans MS','40'))

Label\_question2.pack(padx=100,pady=100)

def fake\_text(e):

answer1.delete(0,"end")

answer1 = Entry(window,width=30,fg="red",bg="yellow",font=('Arial 50'))

answer1.pack(padx=100,pady=100)

answer1.insert(0, "enter your answer here!")

answer1.bind("<FocusIn>", fake\_text)

Label\_error1= Label(window,text="Sorry please enter a number, not a alphabet or don't leave it blank",bg="white",font=('Arial 20'))

#This part of the function checks his result and displays the correct or wrong page.

def check\_ansfact1():

answer1.destroy()

Label\_question2.destroy()

Label\_error1.destroy()

btn\_checkfact1.destroy()

if x == result:

window.configure(bg="green")

Label\_sign\_fact = Label(window, text = "Congrats!you are correct! answer is:", fg="yellow",bg="green",font=('Arial 50'))

Label\_sign\_fact.place(x=450,y=300)

Label\_correct\_sign = Label(window, text = result, fg="yellow",bg="green",font=('Arial 50'))

Label\_correct\_sign.place(x=450,y=500)

else:

window.configure(bg="red")

Label\_sign\_fact = Label(window, text = "SORRY!you are wrong! answer is :", fg="black",bg="red",font=('Arial 50'))

Label\_sign\_fact.place(x=450,y=300)

Label\_correct\_sign = Label(window, text = result, fg="yellow",bg="red",font=('Arial 50'))

Label\_correct\_sign.place(x=450,y=500)

#This function basically goes back to the factorial function to retry the factorial if he gets it wrong.

def retry1():

Label\_sign\_fact.destroy()

Label\_correct\_sign.destroy()

btn\_retry.destroy()

btn\_return.destroy()

factorial\_page()

#This function basically goes back to username section if the user wants to try other categories.

def return1():

Label\_sign\_fact.destroy()

Label\_correct\_sign.destroy()

btn\_retry.destroy()

btn\_return.destroy()

label\_check.destroy()

Label\_brand.destroy()

tab1()

#This is the retry button which connects to the retry function

btn\_retry = Button(window,text="retry",command=retry1,fg="white",bg="black",width=40,height=5)

btn\_retry.place(x=1400,y=800)

#This is the return button which connects to the return function

btn\_return = Button(window,text="exit",command=return1,fg="white",bg="black",width=40,height=5)

btn\_return.place(x=1400,y=600)

#This is exception function to get the answer to the user own factorial. If the user inputs a value other than integer data type a error will be raised.

def exceptionqfact1():

try:

int(answer1.get())

global x

x = int(answer1.get())

check\_ansfact1()

except ValueError:

Label\_error1.place(x=300,y=800)

#This button is after the user types his answer, it will lead to do some exception checks at the exception function.

btn\_checkfact1 = Button(window,text="next",command=exceptionqfact1,fg="white",bg="black",width=40,height=5)

btn\_checkfact1.place(x=1400,y=800)

#This is exception function to get the input of the factorial. If the user inputs a value other than integer data type a error will be raised.

def exceptionqfact():

try:

int(answer.get())

global r

r = int(answer.get())

check\_ansfact()

except ValueError:

Label\_error.place(x=300,y=800)

#This button is after the user types his input, it will lead to do some exception checks at the exception function.

btn\_checkfact = Button(window,text="next",command=exceptionqfact,fg="white",bg="black",width=40,height=5)

btn\_checkfact.place(x=1400,y=800)

#This button is to start the factorial small quiz

start\_quiz\_btn= Button(window,text="START",bg="black",fg="white",command = factorial\_page2)

start\_quiz\_btn.config(font=('Comic Sans MS','60'))

start\_quiz\_btn.pack(side= RIGHT, padx =700,pady=50)

# This function is to display the rules in the anime quiz.

def tab4():

window.configure(bg="white")

btn\_factorial.destroy()

btn\_math.destroy()

Label\_cat.destroy()

Label\_rule = Label(window,text="Rules",fg="Red",bg="white")

Label\_rule.config(font=('Comic Sans MS','100','underline'))

Label\_rule.pack(padx=90,pady=90)

label\_rule1 = Label(window,text="-You have a 15 second time limit for each question.",bg="white")

label\_rule1.config(font=('Comic Sans MS','30'))

label\_rule1.pack(padx=0,pady=0)

label\_rule2=Label(window,text="-There are 5 questions, the moment you start the question the timer will also start.",bg="white")

label\_rule2.config(font=('Comic Sans MS','30'))

label\_rule2.pack(padx=0,pady=0)

label\_rule3=Label(window,text="-If the timer runs out and you have typed the answer in the box,",bg="white")

label\_rule3.config(font=('Comic Sans MS','30'))

label\_rule3.pack(padx=0,pady=0)

label\_rule4=Label(window,text="the answer will be checked",bg="white")

label\_rule4.config(font=('Comic Sans MS','30'))

label\_rule4.pack(padx=0,pady=0)

#This function is the loading screen, for 5 seconds. It also shows the question in the process. The question is picked from the txt.file and placed in a list dictionary.

def loading():

label\_rule4.destroy()

label\_rule3.destroy()

Label\_rule.destroy()

label\_rule1.destroy()

label\_rule2.destroy()

start\_quiz\_btn.destroy()

with open('animequestions.txt') as f: # This opens the file

a = dict(i.rstrip().split(None, 1) for i in f) # This takes each item in the file as if the item in the file has a space. For example if my item in txt.file is "Q1 BLA BLA", the dictionary key will be Q1 and values are "BLA BLA".

Label\_question= Label(window,text=a['Q1'],bg="white")

Label\_question.config(font=('Comic Sans MS','40'))

Label\_question.pack(padx=300,pady=300)

global amount\_of\_points2

amount\_of\_points2 = 0

point = Label(window,text=amount\_of\_points2,fg="white",bg="black",width=10,height=2)

point.place(x=800,y=30)

point\_label = Label(window,text="Points gained:",fg="white",bg="black",width=10,height=2).place(x=700,y=30)

sign\_label = Label(window,text = "Loading please wait....",bg="white")

sign\_label.config(font=('Comic Sans MS','20'))

sign\_label.pack(padx=100,pady=50)

# This is the first question. A photo is placed down and a entrybox witha fake text will appear. The user will be required to guess the correct answer for question 1. The question is taken from dictionary as a label.

def q1():

sign\_label.destroy()

Label\_question.destroy()

Label\_question1= Label(window,text=a['Q1'],bg="white")

Label\_question1.config(font=('Comic Sans MS','40'))

Label\_question1.pack(padx=100,pady=100)

global photo\_q1

global new\_image\_q1

global new\_image\_q1#the following 5 lines of code has been taken from <https://www.codegrepper.com/code-examples/python/how+to+resize+image+in+python+tkinter>

photo\_q1 = (Image.open(“manga anime.jpg”))

resized\_photo\_q1 = photo\_q1.resize((500,300),Image. ANTIALIAS)

new\_image\_q1= ImageTk.PhotoImage(resized\_photo\_q1)

image\_label\_q1= Label(window,image = new\_image\_q1)

image\_label\_q1.pack()

def fake\_textI:#The following 4 lines of code was taken from <https://stackoverflow.com/questions/27820178/how-to-add-placeholder-to-an-entry-in-tkinter>

answer\_q1.delete(0,”end”)

answer\_q1 = Entry(window,width=30,fg=”red”,bg=”yellow”,font=(‘Arial 50’))

answer\_q1.pack(padx=100,pady=100)

answer\_q1.insert(0, “enter your answer here!”)

answer\_q1.bind(“<FocusIn>”, fake\_text)

correction1 = Label(window,text=’’,fg=”black”)

correction2 = Label(window,text=’’,fg=”black”)

correction1.place(x=100,y=850)

correction2.place(x=800,y=850)

with open(‘answer.txt’) as f:

ans = dict(i.rstrip().split(None, 1) for i in f)

answer\_to\_1 = ans[‘Ans1’]

alternate\_1 = ans[‘Alt\_ans’] # There are 2 awnswers, therefore the dictionary contains 2 answers collected from the file.

def check1(): # here the answer to question1 will be checked. Two tabs are made, correctab and wrongtab. The correctab displays a window that answer is correct if the users answer is exactly the answer provided. or else it will go to a wrongtab. The answers are taken from the function above in another text file and placed as dictionary.

check\_answer\_1=answer\_q1.get()

global correctab

def correctab():

window.configure(bg="green")

global Label\_sign\_q1

Label\_sign\_q1 = Label(window, text = "Congrats!you are correct!", fg="yellow",bg="green",font=('Arial 50'))

Label\_sign\_q1.place(x=450,y=300)

global Label\_correct\_sign

Label\_correct\_sign = Label(window, text = "100 points have been awarded!", fg="yellow",bg="green",font=('Arial 50'))

Label\_correct\_sign.place(x=450,y=500)

global wrongtab

def wrongtab():

window.configure(bg="red")

global Label\_sign\_q1

Label\_sign\_q1 = Label(window, text = “SORRY!you are wrong”, fg=”black”,bg=”red”,font=(‘Arial 50’))

Label\_sign\_q1.place(x=450,y=300)

global Label\_correct\_sign

Label\_correct\_sign = Label(window, text = “0 points have been awarded…”, fg=”black”,bg=”red”,font=(‘Arial 50’))

Label\_correct\_sign.place(x=450,y=500)

if check\_answer\_1 == answer\_to\_1 or check\_answer\_1 == alternate\_1:

window.after\_cancel(time3)

answer\_q1.destroy()

image\_label\_q1.destroy()

correction1.destroy()

correction2.destroy()

Label\_question1.destroy()

btn\_next2.destroy()

point.destroy()

global amount\_of\_points

amount\_of\_points = amount\_of\_points2 + 100 # each time the user gets correct, the amount of points will increment by a 100, if wrong then it increments by none.

Global point\_after

point\_after = Label(window,text=amount\_of\_points,fg=”white”,bg=”black”,width=10,height=2)

point\_after.place(x=800,y=30)

correctab()

else:

window.after\_cancel(time3)

btn\_next2.destroy()

answer\_q1.destroy()

correction1.destroy()

correction2.destroy()

Label\_question1.destroy()

image\_label\_q1.destroy()

point.destroy()

amount\_of\_points = amount\_of\_points2 + 0

point\_after = Label(window,text=amount\_of\_points,fg="white",bg="black",width=10,height=2)

point\_after.place(x=800,y=30)

wrongtab()

#at the end of each tab there is always a button to go next. It has to be placed all the way below to go next page or else an error will occur.

# Once the button is pressed, the user will be led to the second loading page before question2.

def loading1():

window.config(bg="white")

btn\_nextq2.destroy()

Label\_correct\_sign.destroy()

Label\_sign\_q1.destroy()

Label\_question= Label(window,text=a['Q2'],bg="white")

Label\_question.config(font=('Comic Sans MS','40'))

Label\_question.pack(padx=100,pady=100)

sign\_label = Label(window,text = "Loading please wait....",bg="white")

sign\_label.config(font=('Comic Sans MS','20'))

sign\_label.pack(padx=100,pady=50)

# In question 2 a similar approach will occur, except this time there will be a error raised. That is if the user types nothing or a wrong data type which isnt integer. The exception is all the way below however the label error is assigned in here.

def q2():

sign\_label.destroy()

global photo\_q2

global new\_image\_q2

global new\_image\_q2#the following 5 lines of code has been taken from https://www.codegrepper.com/code-examples/python/how+to+resize+image+in+python+tkinter

photo\_q2 = (Image.open("naruto1.jpg"))

resized\_photo\_q2 = photo\_q2.resize((500,300),Image. ANTIALIAS)

new\_image\_q2= ImageTk.PhotoImage(resized\_photo\_q2)

image\_label\_q2= Label(window,image = new\_image\_q2)

image\_label\_q2.pack()

def fake\_text(e):#The following 4 lines of code was taken from https://stackoverflow.com/questions/27820178/how-to-add-placeholder-to-an-entry-in-tkinter

answer\_q2.delete(0,"end")

answer\_q2 = Entry(window,width=30,fg="red",bg="yellow",font=('Arial 50'))

answer\_q2.pack(padx=100,pady=100)

answer\_q2.insert(0, "enter your answer here!")

answer\_q2.bind("<FocusIn>", fake\_text)

answer\_to\_2 = ans['Ans2']

global Label\_error

Label\_error= Label(window,text="Sorry please enter a number, not a alphabet or don't leave it blank",bg="white",font=('Arial 20'))

def check\_ans2(): # here the answer inputted to question 2 will be checked.

ans\_q2 = answer\_q2.get()

if ans\_q2 == answer\_to\_2:

window.after\_cancel(time4)

Label\_error.destroy()

btn\_checkq2.destroy()

point\_after.destroy()

answer\_q2.destroy()

image\_label\_q2.destroy()

Label\_question.destroy()

amount\_of\_points3 = amount\_of\_points + 100

point\_afterq2 = Label(window,text=amount\_of\_points3,fg="white",bg="black",width=10,height=2)

point\_afterq2.place(x=800,y=30)

correctab()

else:

window.after\_cancel(time4)

Label\_error.destroy()

btn\_checkq2.destroy()

point\_after.destroy()

answer\_q2.destroy()

image\_label\_q2.destroy()

Label\_question.destroy()

amount\_of\_points3 = amount\_of\_points + 0

point\_afterq2 = Label(window,text=amount\_of\_points3,fg="white",bg="black",width=10,height=2)

point\_afterq2.place(x=800,y=30)

wrongtab()

# This is the loading tab before question 3.

def loading3():

window.config(bg="white")

btn\_nextq3.destroy()

Label\_correct\_sign.destroy()

Label\_sign\_q1.destroy()

Label\_question= Label(window,text=a['Q3'],bg="white")

Label\_question.config(font=('Comic Sans MS','40'))

Label\_question.pack(padx=100,pady=100)

sign\_label = Label(window,text = "Loading please wait....",bg="white")

sign\_label.config(font=('Comic Sans MS','20'))

sign\_label.pack(padx=100,pady=50)

# Question 3 here is placed, exactly like question 2 however this time there isn't any exceptions.

def q3():

sign\_label.destroy()

global photo\_q3

global new\_image\_q3

global new\_image\_q3

photo\_q3 = (Image.open("emiya.jpg"))#the following 5 lines of code has been taken from <https://www.codegrepper.com/code-examples/python/how+to+resize+image+in+python+tkinter>

resized\_photo\_q3 = photo\_q3.resize((500,300),Image. ANTIALIAS)

new\_image\_q3= ImageTk.PhotoImage(resized\_photo\_q3)

image\_label\_q3= Label(window,image = new\_image\_q3)

image\_label\_q3.pack()

def fake\_text(e):#The following 4 lines of code was taken from https://stackoverflow.com/questions/27820178/how-to-add-placeholder-to-an-entry-in-tkinter

answer\_q3.delete(0,"end")

answer\_q3 = Entry(window,width=30,fg="red",bg="yellow",font=('Arial 50'))

answer\_q3.pack(padx=100,pady=100)

answer\_q3.insert(0, "enter your answer here!")

answer\_q3.bind("<FocusIn>", fake\_text)

answer\_to\_3 = ans['Ans3']

alt\_ans\_3 = ans['Alt\_ans3']

# This is to check question3 just like all the other questions

def check\_q3():

ans3 = answer\_q3.get()

if ans3 == answer\_to\_3 or ans3 == alt\_ans\_3:

window.after\_cancel(time5)

Label\_question.destroy()

answer\_q3.destroy()

image\_label\_q3.destroy()

btn\_next\_q3.destroy()

amount\_of\_points4 = amount\_of\_points3 + 100

point\_afterq3 = Label(window,text=amount\_of\_points4,fg="white",bg="black",width=10,height=2)

point\_afterq3.place(x=800,y=30)

correctab()

else:

window.after\_cancel(time5)

Label\_question.destroy()

answer\_q3.destroy()

image\_label\_q3.destroy()

btn\_next\_q3.destroy()

amount\_of\_points4 = amount\_of\_points3 + 0

point\_afterq3 = Label(window,text=amount\_of\_points4,fg="white",bg="black",width=10,height=2)

point\_afterq3.place(x=800,y=30)

wrongtab()

# This is the loading screen before question4

def loading4():

window.config(bg="white")

btn\_nextq4.destroy()

Label\_correct\_sign.destroy()

Label\_sign\_q1.destroy()

Label\_question= Label(window,text=a['Q4'],bg="white")

Label\_question.config(font=('Comic Sans MS','40'))

Label\_question.pack(padx=100,pady=100)

sign\_label = Label(window,text = "Loading please wait....",bg="white")

sign\_label.config(font=('Comic Sans MS','20'))

sign\_label.pack(padx=100,pady=50)

# This is to make the window to answer question4, this time a exception is raised, just like in question2.

def q4():

sign\_label.destroy()

global photo\_q4

global new\_image\_q4

global new\_image\_q4#the following 5 lines of code has been taken from https://www.codegrepper.com/code-examples/python/how+to+resize+image+in+python+tkinter

photo\_q4 = (Image.open("senku3.jpg"))

resized\_photo\_q4 = photo\_q4.resize((500,300),Image. ANTIALIAS)

new\_image\_q4= ImageTk.PhotoImage(resized\_photo\_q4)

image\_label\_q4= Label(window,image = new\_image\_q4)

image\_label\_q4.pack()

def fake\_text(e):#The following 4 lines of code was taken from https://stackoverflow.com/questions/27820178/how-to-add-placeholder-to-an-entry-in-tkinter

answer\_q4.delete(0,"end")

answer\_q4 = Entry(window,width=30,fg="red",bg="yellow",font=('Arial 50'))

answer\_q4.pack(padx=100,pady=100)

answer\_q4.insert(0, "enter your answer here!")

answer\_q4.bind("<FocusIn>", fake\_text)

answer\_to\_4 = ans['Ans4']

global Label\_error4

Label\_error4= Label(window,text="Sorry please enter a number, not a alphabet or try not to leave it blank",bg="white",font=('Arial 20'))

#This is to check question 4.

def check\_ans4():

ans\_q4 = answer\_q4.get()

if ans\_q4 == answer\_to\_4:

window.after\_cancel(time6)

Label\_error4.destroy()

btn\_checkq4.destroy()

point\_afterq3.destroy()

answer\_q4.destroy()

image\_label\_q4.destroy()

Label\_question.destroy()

amount\_of\_points5 = amount\_of\_points4 + 100

point\_afterq4 = Label(window,text=amount\_of\_points5,fg="white",bg="black",width=10,height=2)

point\_afterq4.place(x=800,y=30)

correctab()

else:

window.after\_cancel(time6)

Label\_error4.destroy()

btn\_checkq4.destroy()

point\_afterq3.destroy()

answer\_q4.destroy()

image\_label\_q4.destroy()

Label\_question.destroy()

amount\_of\_points5 = amount\_of\_points4 + 0

point\_afterq4 = Label(window,text=amount\_of\_points5,fg="white",bg="black",width=10,height=2)

point\_afterq4.place(x=800,y=30)

wrongtab()

# This is the loading screen before question5

def loading5():

window.config(bg="white")

btn\_nextq5.destroy()

Label\_correct\_sign.destroy()

Label\_sign\_q1.destroy()

Label\_question= Label(window,text=a['Q5'],bg="white")

Label\_question.config(font=('Comic Sans MS','40'))

Label\_question.pack(padx=100,pady=100)

sign\_label = Label(window,text = "Loading please wait....",bg="white")

sign\_label.config(font=('Comic Sans MS','20'))

sign\_label.pack(padx=100,pady=50)

#This is to display question5 in a window, however no exception is made here.

def q5():

sign\_label.destroy()

global photo\_q5

global new\_image\_q5

global new\_image\_q5#the following 5 lines of code has been taken from <https://www.codegrepper.com/code-examples/python/how+to+resize+image+in+python+tkinter>

photo\_q5 = (Image.open(“hunter.jpg”))

resized\_photo\_q5 = photo\_q5.resize((500,300),Image. ANTIALIAS)

new\_image\_q5= ImageTk.PhotoImage(resized\_photo\_q5)

image\_label\_q5= Label(window,image = new\_image\_q5)

image\_label\_q5.pack()

def fake\_text€:#The following 4 lines of code was taken from <https://stackoverflow.com/questions/27820178/how-to-add-placeholder-to-an-entry-in-tkinter>

answer\_q5.delete(0,”end”)

answer\_q5 = Entry(window,width=30,fg=”red”,bg=”yellow”,font=(‘Arial 50’))

answer\_q5.pack(padx=100,pady=100)

answer\_q5.insert(0, “enter your answer here!”)

answer\_q5.bind(“<FocusIn>”, fake\_text)

answer\_to\_5 = ans[‘Ans5’]

# This is to check question 5 which is the final question. The button will then lead to the leaderboard page.

def check\_ans5():

ans\_q5 = answer\_q5.get()

if ans\_q5 == answer\_to\_5:

window.after\_cancel(time7)

btn\_checkq5.destroy()

point\_afterq4.destroy()

answer\_q5.destroy()

image\_label\_q5.destroy()

Label\_question.destroy()

amount\_of\_points6 = amount\_of\_points5 + 100

point\_afterq5 = Label(window,text=amount\_of\_points6,fg="white",bg="black",width=10,height=2)

point\_afterq5.place(x=800,y=30)

correctab()

else:

window.after\_cancel(time7)

btn\_checkq5.destroy()

point\_afterq4.destroy()

answer\_q5.destroy()

image\_label\_q5.destroy()

Label\_question.destroy()

amount\_of\_points6 = amount\_of\_points5 + 0

point\_afterq5 = Label(window,text=amount\_of\_points6,fg="white",bg="black",width=10,height=2)

point\_afterq5.place(x=800,y=30)

wrongtab()

#The leaderboard page where the leaderboard is displayed.

def leaderboard():

window.config(bg="black")

btn\_leaderboard.destroy()

Label\_correct\_sign.destroy()

Label\_sign\_q1.destroy()

def points(): # The points are taken and stored in a text file as string.

with open("amount of point.txt", "a") as a\_file:

a\_file.write("\n")

a\_file.write(str(amount\_of\_points6))

points()

f = open("amount of point.txt","r")

# The file is then open again and the program will take the content and append them into a list. The list will then be sorted out. The labels will then display top 5 items in the list.

with open('amount of point.txt') as f:

my\_list = [ int(x) for x in f.read().split()]

my\_list1 = [0,0,0,0,0]

final\_list = my\_list1 + my\_list

final\_list.sort(reverse=True)

Label\_leaderboard = Label(window,text="leaderboard",fg='red',bg="black")

Label\_leaderboard.config(font = ('Arial','60','underline'))

Label\_leaderboard.pack(padx =100,pady=100)

Label\_name1= Label(window,text="username:",bg="black", fg="red",font = 'Arial 50')

Label\_name1.place(x =0,y=200)

Label\_name= Label(window,text=entrybox1,bg="black",fg='red',font = 'Arial 50')

Label\_name.place(x =500,y=200)

Label\_first1= Label(window,text="1st:",bg="black",fg="red",font = 'Arial 50')

Label\_first1.place(x =50,y=400)

Label\_first= Label(window,text=final\_list[0],bg="black",fg="red",font = 'Arial 50')

Label\_first.place(x =500,y=400)

Label\_2nd= Label(window,text="2nd:",bg="black",fg="red",font = 'Arial 50')

Label\_2nd.place(x =50,y=500)

Label\_second= Label(window,text=final\_list[1],bg="black",fg="red",font = 'Arial 50')

Label\_second.place(x =500,y=500)

Label\_3rd= Label(window,text="3rd:",bg="black",fg="red",font = 'Arial 50')

Label\_3rd.place(x =50,y=600)

Label\_third= Label(window,text=final\_list[2],bg="black",fg="red",font = 'Arial 50')

Label\_third.place(x =500,y=600)

Label\_4th= Label(window,text="4th:",bg="black",fg="red",font = 'Arial 50')

Label\_4th.place(x =50,y=700)

Label\_fourth= Label(window,text=final\_list[3],bg="black",fg="red",font = 'Arial 50')

Label\_fourth.place(x =500,y=700)

Label\_5th= Label(window,text="5th:",bg="black",fg="red",font = 'Arial 50')

Label\_5th.place(x =50,y=800)

Label\_fifth= Label(window,text=final\_list[4],bg="black",fg="red",font = 'Arial 50')

Label\_fifth.place(x =500,y=800)

#This is a retry function. If the retry button is pressed, the retry function appears and will reset the tab to tab4(). The rule page.

def retry():

Label\_leaderboard.destroy()

Label\_name1.destroy()

Label\_name.destroy()

Label\_first1.destroy()

Label\_first.destroy()

Label\_2nd.destroy()

Label\_second.destroy()

Label\_3rd.destroy()

Label\_third.destroy()

Label\_4th.destroy()

Label\_fourth.destroy()

Label\_5th.destroy()

Label\_fifth.destroy()

btn\_retry.destroy()

btn\_end.destroy()

tab4()

# However if user decides to quit, the text file will be erased of all data and the window will fully close.

def theend():

file = open("amount of point.txt","r+")

file.truncate(0)

file.close()

window.destroy()

# The button which leads to retry function

btn\_retry = Button(window,text="retry",command=retry,fg="black",bg="white",width=40,height=5)

btn\_retry.place(x=1400,y=800)

# The button which leads to thend function where everything is erased and destroyed

btn\_end = Button(window,text="exit game",command=theend,fg="black",bg="white",width=40,height=5)

btn\_end.place(x=1400,y=300)

# The button which leads to leaderboard function

btn\_leaderboard = Button(window,text="next",command=leaderboard,fg="white",bg="black",width=40,height=5)

btn\_leaderboard.place(x=1400,y=800)

# This is a time block, where the page will only be displayed for 15 seconds and then it will go to checkans5() function

time7=window.after(15000,check\_ans5)

# The button which leads to checking of answer for question5 function

btn\_checkq5 = Button(window,text="next",command=check\_ans5,fg="white",bg="black",width=40,height=5)

btn\_checkq5.place(x=1400,y=800)

# The timer is for 5 seconds before question 5, this is loading screen.

Window.after(5000,q5)

#This button is to follow up to the loading screen before question5

btn\_nextq5 = Button(window,text=”continue”,command=loading5,fg=”white”,bg=”black”,width=40,height=5)

btn\_nextq5.place(x=1400,y=800)

# This is timer for question 4. After timer is done it will go to check\_ans4() function.

Time6=window.after(15000,check\_ans4)

#exception raise block for question4

def exceptionq4():

try:

int(answer\_q4.get())

check\_ans4()

except ValueError:

Label\_error4.place(x=300,y=800)

# This is button to go to the exception to check answer in question4

btn\_checkq4 = Button(window,text="next",command=exceptionq4,fg="white",bg="black",width=40,height=5)

btn\_checkq4.place(x=1400,y=800)

# The timer is for 5 seconds before question 4, this is loading screen.

window.after(5000,q4)

#This button is to follow up to the loading screen before question4

btn\_nextq4 = Button(window,text=”continue”,command=loading4,fg=”white”,bg=”black”,width=40,height=5)

btn\_nextq4.place(x=1400,y=800)

# This is timer for question 3. After timer is done it will go to check\_ans3() function.

Time5=window.after(15000,check\_q3)

# This is button to go to check answer in question3

btn\_next\_q3 = Button(window,text=”continue”,command=check\_q3,fg=”white”,bg=”black”,width=40,height=5)

btn\_next\_q3.place(x=1400,y=800)

# The timer is for 5 seconds before question 3, this is loading screen.

Window.after(5000,q3)

# This is button to go to loading before question3

btn\_nextq3 = Button(window,text=”continue”,command=loading3,fg=”white”,bg=”black”,width=40,height=5)

btn\_nextq3.place(x=1400,y=800)

# This is timer for question 2. After timer is done it will go to check\_ans2() function.

Time4=window.after(15000,check\_ans2)

#This is a exception block to check answer 2, if the answer is other than a nteger data type an error will be raised. Or else it will move forward to the next function

def exceptionq2():

try:

int(answer\_q2.get())

check\_ans2()

except ValueError:

Label\_error.place(x=300,y=800)

#This button is to take the users nswer and go to the exception function to check weather it is integer.

Btn\_checkq2 = Button(window,text=”next”,command=exceptionq2,fg=”white”,bg=”black”,width=40,height=5)

btn\_checkq2.place(x=1400,y=800)

# The timer is for 5 seconds before question 2, this is loading screen.

Window.after(5000,q2)

# This is button to go to loading before question2

btn\_nextq2 = Button(window,text=”next”,command=loading1,fg=”white”,bg=”black”,width=40,height=5)

btn\_nextq2.place(x=1400,y=800)

# This is button to go to check answer in question1

btn\_next2 = Button(window,text=”next”,command=check1,fg=”white”,bg=”black”,width=50,height=5)

btn\_next2.place(x=1400,y=800)

# This is timer for question 1. After timer is done it will go to check1() function.

Time3=window.after(15000,check1)

#This is timer for the first loading screen, before question 1

window.after(5000,q1)

#This is the start quiz button to start quiz from rule pagee, it leads to the loading screen function

start\_quiz\_btn= Button(window,text="START",bg="black",fg="white",command = loading)

start\_quiz\_btn.config(font=('Comic Sans MS','60'))

start\_quiz\_btn.pack(side= RIGHT, padx =700,pady=50)

#This button takes a factorial image and acts as a button, when pressed will lead to the factorial\_page function(Do note this is from the category page)

btn\_factorial = Button(window, image=new\_imagemath,command=factorial\_page,bg=”black”)

btn\_factorial.place(x=100,y=450)

#This button takes a anime image and acts as a button, when pressed will lead to the tab4 function(Do note this is from the category page)

btn\_math = Button(window, image=new\_image2,command=tab4,bg=”black”)

btn\_math.place(x=800,y=450)

#btn next to go after username for specifically tab 1

btn\_next = Button(window,text="next",command=tab2,fg="white",bg="black",)

btn\_next.config(font=('Comic Sans MS','50'))

btn\_next.pack(side=RIGHT,pady=5)

#main menu screen

window.configure(bg="black")

label1 = tk.Label(window,text="KAHOOD!",fg="white",bg="black")

label1.config(font=('Comic Sans MS','200','underline'))

label1.pack()#the following 7 lines of code has been taken from https://www.codegrepper.com/code-examples/python/how+to+resize+image+in+python+tkinter

photo = (Image.open("startbutton2.png"))

resized\_photo = photo.resize((600,300),Image. ANTIALIAS)

new\_image= ImageTk.PhotoImage(resized\_photo)

photo2 = (Image.open("exit2.jpg"))

resized\_photo1 = photo2.resize((800,300),Image. ANTIALIAS)

new\_image1= ImageTk.PhotoImage(resized\_photo1)

btn = Button(window, image =new\_image,command=tab1,bg="black")#When startbutton pressed it will lead to function tab1()

btn.pack(side = LEFT, padx =25,pady=20)

btn2 = Button(window, image=new\_image1,command = window.destroy,bg=”black”)#When exit button pressed(or picture) the whole window will be destroyed

btn2.pack(side = RIGHT, padx =25,pady=20)

window.mainloop() # The mainloop waits for events from the user and executes the program until the user exits the window.

## References

1. Stonkus(2020). How to resize image in python tkinter. Retrieved from https://www.codegrepper.com/code-examples/python/how+to+resize+image+in+python+tkinter
2. Nae(2017). How to add placeholder to an Entry in Tkinter?. Retrieved from <https://stackoverflow.com/questions/27820178/how-to-add-placeholder-to-an-entry-in-tkinter>
3. Niklas.R(2011). How to render my TextArea with WTForms?. Retrieved from <https://stackoverflow.com/questions/7979548/how-to-render-my-textarea-with-wtforms>
4. Bartnyk(2021). WTForms TextAreaField Length validate. Retrieved from <https://stackoverflow.com/questions/67619362/wtforms-textareafield-length-validate>
5. Red.C(2020). WTForms RadioField preventing from validation. Retrieved from <https://stackoverflow.com/questions/53101631/wtforms-radiofield-preventing-form-validation>