

(P.O Box 26845, Al Quoz,Dubai, UAE)

Tel No: 04- 3391188 Fax: 04- 3391188 Email: info\_ois@gemsedu.com

Website: www.gemsoo-alquoz.com

**A COMPUTER PROJECT REPORT ON**

**PYTHON GUI GAMES**

**Session: 2021-2022**

**GEMS OUR OWN INDIAN SCHOOL, DUBAI**

Submitted to: submitted by:

mr. DEVANAND serah kurian

CLASS: XII b REG NO:



CERTIFICATE

This is to certify that the Project Titled PYTHON GUI GAMES is a bonafide work done by \_\_\_\_\_\_\_\_\_Serah Grace Kurian \_\_\_\_\_\_\_\_\_\_ of grade \_\_\_\_12 B\_\_\_\_ under Computer Science in partial fulfilment of the requirements for All India Senior Certificate Examination as prescribed by the Central Board of Secondary Education (CBSE) during the academic year 2021-2022.

Date: \_\_\_\_\_\_\_\_\_\_\_

Teacher In charge: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Internal Examiner External Examiner Principal

**ACKNOWLEDGMENT**

It is with pleasure that i acknowledge my sincere gratitude to our teacher, Mr. ‘Devanand .A. V. ’ who taught and undertook the responsibility of teaching the subject computer science. I have been greatly benefited from his classes.

I am especially indebted to our principal Ms. Lalitha Suresh who has always been a source of encouragement and support & has always been a source of encouragement & support and without whose inspiration this project would not have been a successful I would like to place on record heartfelt thanks to her.

Finally, I would like to express my sincere appreciation for all the other students from my batch their friendship & the fine times that we all shared together.

**TABLE OF CONTENTS**

* **Objective pageno 5**
* **Package/module required**  **pageno 6**

* **Requirements pageno 7**
* **Coding pageno 10**
* **Conclusion pageno 32**
* **Bibliography pageno 33**

**OBJECTIVE**

**The aim of this project to create three basic games employing tkinter and pygame module. The games are as follows:**

**.Speed typing test**

**. Tic-Tac-Toe**

**. Colour game**

**A brief about the mentioned games are explained below.**

**Speed typing test –**

Typing speed is measured by the number of words you can type correctly in a set amount of time. A “word” is equivalent to five keystrokes. During a test, both speed and accuracy are measured. You will receive a number that indicates your average words per minute (WPM) and a percentage that indicates your accuracy. It will also provide the time taken to type in seconds.

**Tic-Tac-Toe**

Tic-tac-toe, noughts and crosses, or Xs and Os is a paper-and-pencil game for two players who take turns marking the spaces in a three-by-three grid with X or O. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner.

**Colour game**

In this game player must enter color of the word that appears on the screen and hence the score increases by one, the total time to play this game is 30 seconds. Colors used in this game are Red, Blue, Green, Pink, Black, Yellow, Orange, White, Purple and Brown. Interface will display name of different colors in different colors. Player must identify the color and enter the correct color name to win the game.

**IMPORTED FILES USED AND THEIR PURPOSE**

1. **math** – for mathematical calculations
2. **string** – for string handling
3. **Time module:** This module provides various time

-related functions.

**4. Random module**: Python Random module is an in-built module of Python which is used to generate random numbers**.**

**5.pygame**

**6.sys-** The sys module in Python provides various functions and variables that are used to manipulate different parts of the Python runtime environment.

**7.Tkinter**

**8.** Functools module is for higher-order functions that work on other functions. It provides functions for working with other functions and callable objects to use or extend them without completely rewriting them.

**Requirements:**

Speed Typing Test requirements:

* **Background.jpg –** A background image we will use in our program
* **Icon.png –** An icon image that we will use as a reset button.
* **Sentences.txt –** This text file will contain a list of sentences separated by a new line.
* **Speed typing.py –** The main program file that contains all the code
* **Typing-speed-open.png –** The image to display when starting game

This project in Python requires you to have basic knowledge of python programming and the pygame library.

For this project based on Python, we are using the pygame library.

**PYGAME**

Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language.

A picture containing clipart

Description automatically generated

TIC TAC TOE requirements:

**TKINTER**:

Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit and is Python's de facto standard GUI. Tkinter is included with standard GNU/Linux, Microsoft Windows and macOS installs of Python. The name Tkinter comes from Tk interface.

A picture containing logo

Description automatically generated

**Description of other functions:**

* gameboard\_pc() and gameboard\_pl() will create the another geometry to play the game. It will add 9 buttons on 3×3 board of the game (Three rows of buttons containing three buttons each).
* get\_text\_pc() and get\_text() functions will put the text on buttons as it pressed.
* pc() function will decide the next move of the system.
* winner() function will check whether the player won the match or not.
* isfree() function will check whether the player can put it’s a coin or not.
* isfull() function will check the board is full or not.

Colour game requirements:

**Description of other functions:**

* startGame() will begin the game and the timer
* nextColour(): choose next random colour accordingly.
* Countdown(): displays and counts the number of seconds left.
* **Hardware Required**
* **Printer, to print the required documents of the project**
* **Compact Drive**
* **Processor : Intel Core i3**
* **Ram : 64 MB**
* **Harddisk : 20 Gb.**
* **software Required**
* **Operating system : Windows 10**
* **Python IDLE, for execution of program and**
* **Ms word, for presentation of output.**

**CODING**

#menu page

#Serah Kurian

print('''1. Speed typing test

2.Tic tac toe

3.Colour game

0.Exit''')

choice=int(input('Enter choice'))

import tkinter

import pygame

while True:

if choice==1:

from speed\_typing import \*

elif choice==2:

from colourgame import \*

elif choice==3:

from x\_o import \*

elif choice==0:

print('THANK YOU')

else:

print('Invalid choice')

break

import pygame

from pygame.locals import \*

import sys

import time

import random

# 750 x 500

class Game: #game class

def \_\_init\_\_(self):

self.w=750

self.h=500

self.reset=True

self.active = False

self.input\_text=''

self.word = ''

self.time\_start = 0

self.total\_time = 0

self.accuracy = '0%'

self.results = 'Time:0 Accuracy:0 % Wpm:0 '

self.wpm = 0

self.end = False

self.HEAD\_C = (255,213,102) #R G B values

self.TEXT\_C = (240,240,240)

self.RESULT\_C = (255,70,70)

pygame.init()

self.open\_img = pygame.image.load('type-speed-open.png')

self.open\_img = pygame.transform.scale(self.open\_img, (self.w,self.h))

self.bg = pygame.image.load('background.jpg')

self.bg = pygame.transform.scale(self.bg, (500,750))

self.screen = pygame.display.set\_mode((self.w,self.h))

pygame.display.set\_caption('Type Speed test')

def draw\_text(self, screen, msg, y ,fsize, color):

font = pygame.font.Font(None, fsize)

text = font.render(msg, 1,color) #doesnt allow the direct way to write text on the output screen

text\_rect = text.get\_rect(center=(self.w/2, y))

screen.blit(text, text\_rect) #used to paste one image over another

pygame.display.update() #after drawing anything we must update

def get\_sentence(self): #used to get the sentences from the file

f = open('sentences.txt').read() #using text file

sentences = f.split('\n')

sentence = random.choice(sentences)

return sentence

def show\_results(self, screen):

if(not self.end):

#Calculate time

self.total\_time = time.time() - self.time\_start

#Calculate accuracy

count = 0

for i,c in enumerate(self.word): # gives the count of the word along with the text

try:

if self.input\_text[i] == c:

count += 1

except:

pass

self.accuracy = count/len(self.word)\*100

#Calculate words per minute

self.wpm = len(self.input\_text)\*60/(5\*self.total\_time) # a word contains around 5 letters so to calculate the wpm we divide by 5 and total time and convert to mins

self.end = True

print(self.total\_time)

self.results = 'Time:'+str(round(self.total\_time)) +" secs Accuracy:"+ str(round(self.accuracy)) + "%" + ' Wpm: ' + str(round(self.wpm))

# draw icon image

self.time\_img = pygame.image.load('icon.png')

self.time\_img = pygame.transform.scale(self.time\_img, (150,150))

screen.blit(self.time\_img, (self.w/2-75,self.h-140))

self.draw\_text(screen,"Reset", self.h - 70, 26, (100,100,100))

print(self.results)

pygame.display.update()

def run(self):

self.reset\_game

self.running=True

while(self.running):

clock = pygame.time.Clock()

self.screen.fill((0,0,0), (50,250,650,50))

pygame.draw.rect(self.screen,self.HEAD\_C, (50,250,650,50), 2)

# update the text of user input

self.draw\_text(self.screen, self.input\_text, 274, 26,(250,250,250))

pygame.display.update()

for event in pygame.event.get():

if event.type == QUIT:

self.running = False

sys.exit()

elif event.type == pygame.MOUSEBUTTONUP:

x,y = pygame.mouse.get\_pos()

# position of input box

if(x>=50 and x<=650 and y>=250 and y<=300):

self.active = True

self.input\_text = ''

self.time\_start = time.time()

# position of reset box

if(x>=310 and x<=510 and y>=390 and self.end):

self.reset\_game()

x,y = pygame.mouse.get\_pos()

elif event.type == pygame.KEYDOWN:

if self.active and not self.end:

if event.key == pygame.K\_RETURN:

print(self.input\_text)

self.show\_results(self.screen)

print(self.results)

self.draw\_text(self.screen, self.results,350, 28, self.RESULT\_C)

self.end = True

elif event.key == pygame.K\_BACKSPACE: #used to preform delete

self.input\_text = self.input\_text[:-1]

else:

try:

self.input\_text += event.unicode #unicode?

except:

pass

pygame.display.update()

clock.tick(60) #doesnot run for more than 60 frames per second

def reset\_game(self): #The reset\_game() method resets all variables so that we can start testing our typing speed again.

self.screen.blit(self.open\_img, (0,0))

pygame.display.update()

time.sleep(1)

self.reset=False

self.end = False

self.input\_text=''

self.word = ''

self.time\_start = 0

self.total\_time = 0

self.wpm = 0

# Get random sentence

self.word = self.get\_sentence()

if (not self.word): self.reset\_game()

#drawing heading

self.screen.fill((0,0,0))

self.screen.blit(self.bg,(0,0))

msg = "Typing Speed Test"

self.draw\_text(self.screen, msg,80, 80,self.HEAD\_C)

# draw the rectangle for input box

pygame.draw.rect(self.screen,(255,192,25), (50,250,650,50), 2)

# draw the sentence string

self.draw\_text(self.screen, self.word,200, 28,self.TEXT\_C)

pygame.display.update()

Game().run()

**Tic-tac-toe**

# Tic Tac Toe game with GUI

# using tkinter

# importing all necessary libraries

import random

import tkinter

from tkinter import \*

from functools import partial

from tkinter import messagebox

from copy import deepcopy

# sign variable to decide the turn of which player

sign = 0

# Creates an empty board

global board

board = [[" " for x in range(3)] for y in range(3)]

# Check l(O/X) won the match or not

# according to the rules of the game

def winner(b, l):

return ((b[0][0] == l and b[0][1] == l and b[0][2] == l) or

(b[1][0] == l and b[1][1] == l and b[1][2] == l) or

(b[2][0] == l and b[2][1] == l and b[2][2] == l) or

(b[0][0] == l and b[1][0] == l and b[2][0] == l) or

(b[0][1] == l and b[1][1] == l and b[2][1] == l) or

(b[0][2] == l and b[1][2] == l and b[2][2] == l) or

(b[0][0] == l and b[1][1] == l and b[2][2] == l) or

(b[0][2] == l and b[1][1] == l and b[2][0] == l))

# Configure text on button while playing with another player

def get\_text(i, j, gb, l1, l2):

global sign

if board[i][j] == ' ':

if sign % 2 == 0:

l1.config(state=DISABLED)

l2.config(state=ACTIVE)

board[i][j] = "X"

else:

l2.config(state=DISABLED)

l1.config(state=ACTIVE)

board[i][j] = "O"

sign += 1

button[i][j].config(text=board[i][j])

if winner(board, "X"):

gb.destroy()

box = messagebox.showinfo("Winner", "Player 1 won the match")

elif winner(board, "O"):

gb.destroy()

box = messagebox.showinfo("Winner", "Player 2 won the match")

elif(isfull()):

gb.destroy()

box = messagebox.showinfo("Tie Game", "Tie Game")

# Check if the player can push the button or not

def isfree(i, j):

return board[i][j] == " "

# Check the board is full or not

def isfull():

flag = True

for i in board:

if(i.count(' ') > 0):

flag = False

return flag

# Create the GUI of game board for play along with another player

def gameboard\_pl(game\_board, l1, l2):

global button

button = []

for i in range(3):

m = 3+i

button.append(i)

button[i] = []

for j in range(3):

n = j

button[i].append(j)

get\_t = partial(get\_text, i, j, game\_board, l1, l2)

button[i][j] = Button(

game\_board, bd=5, command=get\_t, height=4, width=8)

button[i][j].grid(row=m, column=n)

game\_board.mainloop()

# Decide the next move of system

def pc():

possiblemove = []

for i in range(len(board)):

for j in range(len(board[i])):

if board[i][j] == ' ':

possiblemove.append([i, j])

move = []

if possiblemove == []:

return

else:

for let in ['O', 'X']:

for i in possiblemove:

boardcopy = deepcopy(board) #create copy

boardcopy[i[0]][i[1]] = let

if winner(boardcopy, let):

return i

corner = []

for i in possiblemove:

if i in [[0, 0], [0, 2], [2, 0], [2, 2]]:

corner.append(i)

if len(corner) > 0:

move = random.randint(0, len(corner)-1)

return corner[move]

edge = []

for i in possiblemove:

if i in [[0, 1], [1, 0], [1, 2], [2, 1]]:

edge.append(i)

if len(edge) > 0:

move = random.randint(0, len(edge)-1)

return edge[move]

# Configure text on button while playing with system

def get\_text\_pc(i, j, gb, l1, l2):

global sign

if board[i][j] == ' ':

if sign % 2 == 0:

l1.config(state=DISABLED)

l2.config(state=ACTIVE)

board[i][j] = "X"

else:

button[i][j].config(state=ACTIVE)

l2.config(state=DISABLED)

l1.config(state=ACTIVE)

board[i][j] = "O"

sign += 1

button[i][j].config(text=board[i][j])

x = True

if winner(board, "X"):

gb.destroy()

x = False

box = messagebox.showinfo("Winner", "Player won the match")

elif winner(board, "O"):

gb.destroy()

x = False

box = messagebox.showinfo("Winner", "Computer won the match")

elif(isfull()):

gb.destroy()

x = False

box = messagebox.showinfo("Tie Game", "Tie Game")

if(x):

if sign % 2 != 0:

move = pc()

button[move[0]][move[1]].config(state=DISABLED)

get\_text\_pc(move[0], move[1], gb, l1, l2)

# Create the GUI of game board for play along with system

def gameboard\_pc(game\_board, l1, l2):

global button

button = []

for i in range(3):

m = 3+i

button.append(i)

button[i] = []

for j in range(3):

n = j

button[i].append(j)

get\_t = partial(get\_text\_pc, i, j, game\_board, l1, l2)

button[i][j] = Button(

game\_board, bd=5, command=get\_t, height=4, width=8)

button[i][j].grid(row=m, column=n)

game\_board.mainloop()

# Initialize the game board to play with system

def withpc(game\_board):

game\_board.destroy()

game\_board = Tk()

game\_board.title("Tic Tac Toe")

l1 = Button(game\_board, text="Player : X", width=10)

l1.grid(row=1, column=1)

l2 = Button(game\_board, text = "Computer : O",

width = 10, state = DISABLED)

l2.grid(row = 2, column = 1)

gameboard\_pc(game\_board, l1, l2)

# Initialize the game board to play with another player

def withplayer(game\_board):

game\_board.destroy()

game\_board = Tk()

game\_board.title("Tic Tac Toe")

l1 = Button(game\_board, text = "Player 1 : X", width = 10)

l1.grid(row = 1, column = 1)

l2 = Button(game\_board, text = "Player 2 : O",

width = 10, state = DISABLED)

l2.grid(row = 2, column = 1)

gameboard\_pl(game\_board, l1, l2)

# main function

def play():

menu = Tk()

menu.geometry("250x250")

menu.title("Tic Tac Toe")

wpc = partial(withpc, menu)

wpl = partial(withplayer, menu)

head = Button(menu, text = "---Welcome to tic-tac-toe---",

activeforeground = 'red',

activebackground = "yellow", bg = "red",

fg = "yellow", width = 500, font = 'summer', bd = 5)

B1 = Button(menu, text = "Single Player", command = wpc,

activeforeground = 'red',

activebackground = "yellow", bg = "red",

fg = "yellow", width = 500, font = 'summer', bd = 5)

B2 = Button(menu, text = "Multi Player", command = wpl, activeforeground = 'red',

activebackground = "yellow", bg = "red", fg = "yellow",

width = 500, font = 'summer', bd = 5)

B3 = Button(menu, text = "Exit", command = menu.quit, activeforeground = 'red',

activebackground = "yellow", bg = "red", fg = "yellow",

width = 500, font = 'summer', bd = 5)

head.pack(side = 'top')

B1.pack(side = 'top')

B2.pack(side = 'top')

B3.pack(side = 'top')

menu.mainloop()

# Call main function

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

play()

**COLOUR GAME**

# import the modules

import tkinter

import random

# list of possible colour.

colours = ['Red','Blue','Green','Pink','Black','Yellow','Orange','White','Purple','Brown']

score = 0

# the game time left, initially 30 seconds.

timeleft = 30

# function that will start the game.

def startGame(event):

if timeleft == 30:

# start the countdown timer.

countdown()

# run the function to

# choose the next colour.

nextColour()

# Function to choose and

# display the next colour.

def nextColour():

# use the globally declared 'score'

# and 'play' variables above.

global score

global timeleft

# if a game is currently in play

if timeleft > 0:

# make the text entry box active.

e.focus\_set()

# if the colour typed is equal

# to the colour of the text

if e.get().lower() == colours[1].lower():

score += 1

# clear the text entry box.

e.delete(0, tkinter.END)

random.shuffle(colours)

# change the colour to type, by changing the

# text \_and\_ the colour to a random colour value

label.config(fg = str(colours[1]), text = str(colours[0]))

# update the score.

scoreLabel.config(text = "Score: " + str(score))

# Countdown timer function

def countdown():

global timeleft

# if a game is in play

if timeleft > 0:

# decrement the timer.

timeleft -= 1

# update the time left label

timeLabel.config(text = "Time left: "

+ str(timeleft))

# run the function again after 1 second.

timeLabel.after(1000, countdown)

# Driver Code

# create a GUI window

root = tkinter.Tk()

# set the title

root.title("COLORGAME")

# set the size

root.geometry("375x200")

# add an instructions label

instructions = tkinter.Label(root, text = "Type in the colour of the words, and not the word text!", font = ('Helvetica', 12))

instructions.pack()

# add a score label

scoreLabel = tkinter.Label(root, text = "Press enter to start", font = ('Helvetica', 12))

scoreLabel.pack()

# add a time left label

timeLabel = tkinter.Label(root, text = "Time left: " +

str(timeleft), font = ('Helvetica', 12))

timeLabel.pack()

# add a label for displaying the colours

label = tkinter.Label(root, font = ('Helvetica', 60))

label.pack()

# add a text entry box for

# typing in colours

e = tkinter.Entry(root)

# run the 'startGame' function

# when the enter key is pressed

root.bind('<Return>', startGame)

e.pack()

# set focus on the entry box

e.focus\_set()

# start the GUI

root.mainloop()

**OUTPUTS**

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, website

Description automatically generated

Graphical user interface, website

Description automatically generated

TIC TAC TOE

Output:

Chart

Description automatically generated with medium confidence

Single player

A picture containing text, indoor, keyboard

Description automatically generated Graphical user interface, text, application, chat or text message

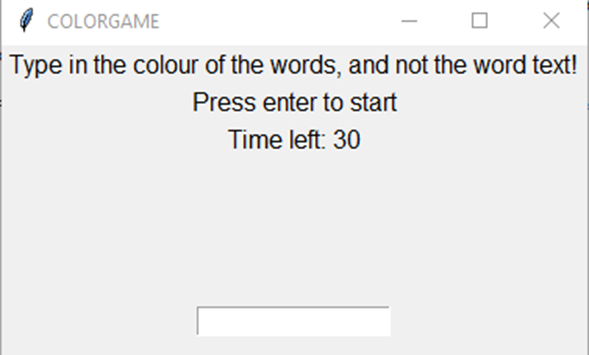
Description automatically generated

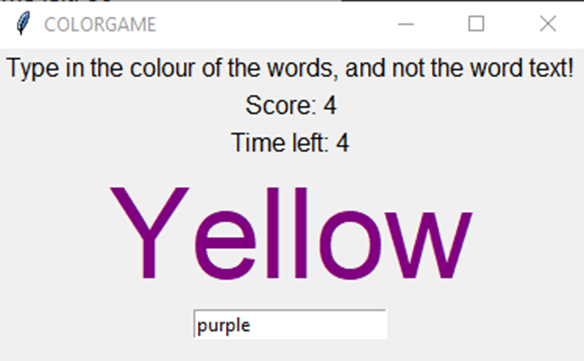
Multiplayer

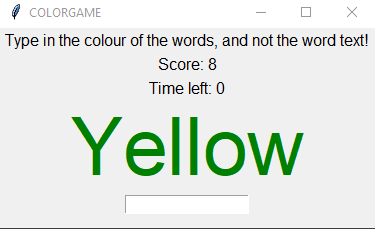
A picture containing text, indoor, keyboard

Description automatically generated Graphical user interface, text, application, Teams

Description automatically generated







**Conclusion**

**THE PROGRAM CODE FOR THE APPLICATION “****PYTHON GUI GAMES” HAS BEEN WRITTEN IN THE PYHTON PROGRAMMING LANGUAGE.**

**THIS SOFTWARE: “PYTHON GUI GAMES” FOUND TO BE SUCCESSFULLY WORKING AND HAS BEEN TESTED & APPROVED.**

**THE FRONT END OF THIS APPLICATION IS BUILT WITH THE HELP OF PYTHON LANGUAGE AND BINARY FILE IS USED TO STORE THE DATA**

**BIBLIOGRAPHY**

1. **PYTHON - SUMITA ARORA**
2. **https://www.geeksforgeeks.org/**
3. **www.w3schools.com**