



2020 - 2021



ARSHA VIDYA MANDIR

114, Velachery Road, Guindy, Chennai, Tamil Nadu 600032

|  |  |
| --- | --- |
| **Name of the Student :** | **RETHIK MANOHARAN** |
| **CBSE Roll No :** |  |
| **Class :** | **XII - ‘A’** |
| **Project Title :** | **Typing Speed Calculator** |
| **Academic Year :** | **2020 - 2021** |

‘

**ARSHA VIDYA MANDIR**

**COMPUTER SCIENCE PROJECT**

**This is certified to be the Bonafide project work done by**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**of Class XII A in the Computer Science laboratory of**

**Arsha Vidya Mandir, Chennai during the year 2020-2021.**

**Teacher in Charge**

**Register No. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Submitted for the practical examination held on**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ at Arsha Vidya Mandir.**

**Chief Superintendent Internal Examiner External Examiner**

# ACKNOWLEDGEMENT

I thank the almighty god for his grace and blessings that he has showered at every stage of this project.

I thank my School Management, Deputy Correspondent, Principal for giving me this wonderful opportunity and for extending all possible supports for completing this project.

I thank my Computer Science Teacher, for providing me required guidance and support. It is my humble pleasure to acknowledge my deep sense of gratitude, for his valuable support, constant help and guidance at each and every stage, without which it wouldn’t have been possible to complete this project.

I express my sincere thanks to all my family members for the moral support and encouragement, which motivated me towards the successful completion of this project.

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Topic** | **Page Number** |
| 1. | Abstract | 6 |
| 2. | Introduction to Python | 7 - 8 |
| 3. | Software Description | 9 |
| 4. | Modules and Packages Used | 10 - 12 |
| 5. | Coding/Project Explanation | 13-30 |
| 6. | Screenshots of Output | 31 -41 |
| 7. | Conclusion | 42 |
| 8. | Bibliography | 43 |

# ABSTRACT

The project is aimed at developing a tool for one to test their typing speed using Python which provides various modules for the development and also with a SQL database to store the players speed scores data. The project is created with the sole purpose of helping one learn touch typing and/or improve their speed. Most of the existing tools are online and don't have database and analysis features. The functionality is similar in both cases with additional database and analysis features.

We also aim at creating a user friendly and smooth gaming experience for the end user. And also, minimization of codes at necessary areas is to be done cautiously to take this game to its maximum efficiency. A GUI has been created with additional tools for the users to view their previous attempts and their achievements.

Since the data in plain text can't help the player to study or make a thing out of it, we have also designed GUI to visualize the data in the form of interactive graphs. The player can plot the data in graphs with parameters and customize to their needs.This helps the users to know their past scores and areas of improvement.

# INTRODUCTION TO PYTHON

**What is Python?**

Python is a high level, general-purpose, interpreted dynamic programming language which is increasingly becoming popular in the world of coding in recent times. The uniqueness of this programming language lies in the fact that its design philosophy gives importance to code readability, and the syntax is designed in such a way that concepts can be expressed in a fewer number of lines than in the case of other programming languages like C++ or Java. Python is capable of supporting multiple programming paradigms like “object-oriented programming”, “imperative programming” and “functional programming”. Python also has automatic memory management along with a vast and comprehensive standard library.

**History**

Python was conceptualized by a man named Guido van Rossum in the late 1980s, and the implementation of this programming began in December 1989 at Centrum Wiskunde & Informatica in the Netherlands.

**Versions**

Python 2.0 was released on 16th October, 2000 and showcased many innovative features that weren’t seen before, including a cycle-detecting garbage collector and a support for Unicode. The development process there hence, became far more transparent and community supported.

Python 3.0, which was a major backwards-incompatible release, was released on 3rd December, 2008 after an extended period of testing.

**Syntax and Semantics**

Python is a programming language which was meant to be a highly readable and user friendly with a clean, uncluttered visual layout. While other languages use punctuation and have a large number of syntactic expressions, Python uses primarily English keywords and has fewer syntactic expressions.

Indentation:

Whitespace indentation is used in Python as opposed to keywords or curly brackets as seen in other languages.

Statements:

Some of the key statements used in Python include, ‘def’, ‘if’, ‘for’, ‘while’, ‘class’, ‘import’, ‘print’, etc.

Methods: Methods are functions which are attached to an object’s class. *instance.method(argument)* is for normal methods and functions. Python methods have an explicit *self* parameter to access ‘instance data’, whereas, implicit *self* or *this* are used in other OOP languages.

**Applications**

Python excels at integration tasks, and therefore, it is an efficient tool for gluing things together. For example, 3D software like ‘Maya’ utilizes Python to automate small user tasks or even for more complex integration like talking to databases and asset management systems.

Due to the combination of easy extensibility, good iteration time and good integration with the database and other web standards, Python is popularly used in web development, although primarily on the back end. Quora is a website which utilizes a lot of this Python code.

**Advantages**

1. The code is easy to read and understand.
2. The standard library allows the execution of a lot of complex functionalities.
3. Supports multiple systems and platforms.
4. Python has a plethora of frameworks that make web programming flexible.
5. Gives rise to quick development using less code.
6. Built in testing framework enables fast workflows.

**Disadvantages**

1. Python is slow compared to other languages like C++.
2. It is not preferred for mobile development.
3. It is not a good choice for memory intensive tasks 4. There are limitations with database access.

5. IT is not good for multi-processor/multi-core work.

# SOFTWARE DESCRIPTION

|  |  |
| --- | --- |
| **Machine Name** | AVM01 |
| **Operating System** | Windows 7 Professional 64-bit (6.1,  Build 7601) Service Pack 1  (7601.win7sp1\_gdr.150427-0707) |
| **Language** | English (Regional Setting: English) |
| **System Manufacturer** | Dell Inc. |
| **System Model** | OptiPlex990 |
| **BIOS** | Default System BIOS |
| **Processor** | IntelR Core™ i5-2400 CPU @  3.10GHz (4 CPUs), ~3.1GHz |
| **Memory** | 4096MB RAM |
| **Available OS Memory** | 3978MB RAM |
| **Page File** | 2179MB used, 5772MB available |
| **Windows Dir** | C:\Windows |
| **DirectX Version** | DirectX 11 |
| **DX Setup Parameters** | Not Found |
| **User DPI Setting** | Using System DPI |
| **System DPI Setting** | 96 DPI (100 percent) |
| **DWM DPI Scaling** | Disabled |
| **DxDiag Version** | 6.01.7601.17514 32bit UNICODE |

# Modules and Packages Used

1. **Random** – to generate random values
   1. random.randint()
   2. random.choice()
2. **Time** - module provides various time-related functions 
   1. time.sleep()
3. **Tkinter -** standard Python interface to the Tk GUI toolkit
   1. tkinter.Tk()
   2. tkinter.Button()
   3. tkinter.Label()
   4. tkinter.Frame()
   5. tkinter.StringVar()
   6. tkinter.IntVar()
   7. tkinter.DoubleVar()
   8. tkinter.Entry()
   9. tkinter.Canvas()
   10. tkinter.Scrollbar()
   11. tkinter.Listbox()
   12. tkinter.PhotoImage()
   13. tkinter.Toplevel()
   14. window.destroy()
   15. window.iconbitmap()
   16. window.overrideredirect()
   17. gadget.config()
   18. var.get()
   19. var.set()
   20. tkinter.after()
   21. tkinter.OptionMenu()
   22. tkinter.ttk.Style()
   23. tkinter.mainloop()
4. **Sqlite3 -** used to create a database, define tables, insert and change rows, run queries and manage an SQLite database file.
   1. sqlite3.connect()
   2. connection.cursor()
   3. cursor.execute()
   4. cursor.fetchall()
   5. connection.row\_factory()
   6. connection.commit()
   7. connection.close()
5. **PIL -** support for opening, manipulating, and saving many different image file formats
   1. PIL.Image()
   2. PIL.Image.open()
   3. PIL.ImageSequence
   4. PIL.ImageTk.PhotoImage()
6. **Os** - provides a portable way of using operating system dependent functionality
   1. os.system()
   2. os.getcwd()
   3. os.chdir()
   4. os.mkdir()
   5. os.path.exists()
7. **urllib.request**- module for opening urls.
   1. urllib.request.urlretreive()
8. **string**- module that manages strings
   1. string.ascii\_lowercase

1. **bs4**- allows us to webscrape
   1. bs4.BeautifulSoup()
2. **requests**- used to interact with urls
   1. requests.get()
3. **matplotlib.pyplot**- helps plot graphs and visualize data
   1. matplotlib.pyplot.plot()
   2. matplotlib.pyplot.xticks()
   3. matplotlib.pyplot.yticks()
   4. matplotlib.pyplot.xlabel()
   5. matplotlib.pyplot.ylabel()
   6. matplotlib.pyplot.title()
   7. matplotlib.pyplot.show()
4. ttkthemes- allows us to add a theme to tkinter windows

12.1 ttkthemes.themedTk()

# Source Code

Files in the Project:

* Project.py

Folders in the Project:

* TypeIt

Other files:

(\*)-Downloaded/Created during execution

(#)-Pre existing

* bgimg.png(\*)
* iconimg.ico(#)
* SpeedScores.db(\*)
* tenor.gif(\*)
* test\_info.txt(#)
* tut\_info.txt(#)
* words.txt(\*)
* touchtyping.png(\*)

## Project.py

import tkinter as tk

import sqlite3

import urllib.request

import time

import random

import string

from PIL import ImageTk, Image, ImageSequence

import bs4

import requests

import os

import matplotlib.pyplot as plt

from tkinter import ttk

from ttkthemes import themed\_tk as tt

if (os.getcwd != 'TypeIt'):

try:

os.mkdir("TypeIt")

except:

os.chdir(os.getcwd() + "/TypeIt")

conn = sqlite3.connect("SpeedScores.db")

cur = conn.cursor()

cur.execute("""CREATE TABLE IF NOT EXISTS Users(

UserID INTEGER PRIMARY KEY AUTOINCREMENT,

UserName VARCHAR(20),

Score INTEGER

Highscore INTEGER)""")

cur.execute("""SELECT DISTINCT(UserName) FROM Users""")

alluserlist = cur.fetchall()

cur.execute(

"""SELECT MAX(Score) FROM Users GROUP BY UserName HAVING UserName <>'Select User' ORDER BY Score DESC """)

highscorelist = cur.fetchall()

conn.commit()

def img\_downloader(a):

if(a==1):

urllib.request.urlretrieve(

'https://i.ibb.co/N3yQkNB/Screenshot-2021-01-28-084757.png',get cwd() + '//bgimg.png')

if(a==2):

urllib.request.urlretrieve( 'https://sites.google.com/site/introductiontokeyboardingcmpld/\_/rsrc/1330540565295/home/touch-typing/i\_fingering\_guide.gif', os.getcwd() + '//touchtyping.png')

if(a==3):

urllib.request.urlretrieve( 'https://media.tenor.com/images/8d0e83860667fe684da6a4564f0fe3bd/tenor.gif',os.getcwd() + '//tenor.gif')

if not (os.path.exists(os.getcwd() + 'bgimg.png')):

img\_downloader(1)

if not (os.path.exists(os.getcwd() + 'touchtype.png')):

img\_downloader(2)

if not (os.path.exists(os.getcwd() + 'tenor.gif')):

img\_downloader(3)

with open("words.txt", 'w+') as file:

if (len(file.readlines()) == 0):

res=requests.get(

'https://www.ef.com/wwen/english-resources/english-vocabulary/top-1000-words/')

soup = bs4.BeautifulSoup(res.text, 'lxml')

word = soup.select('p')

x = word[11].getText()

wordlist = x.split("\r\n\t")

for i in wordlist:

file.write(i + "\n")

with open("test\_info.txt","r") as file2:

testinfo=file2.read()

with open("tut\_info.txt","r") as file3:

tutinfo=file3.read()

def splash\_screen():

splash=tk.Tk()

ws = splash.winfo\_screenwidth()

hs = splash.winfo\_screenheight()

width = (ws / 2) - (300 / 2)

height = (hs / 2) - (100 / 2)

splash.geometry('%dx%d+%d+%d' % (220, 100, width, height))

splash.overrideredirect(True)

def animate(counter):

canvas.itemconfig(image, image=sequence[counter])

splash.after(20, lambda: animate((counter + 1) % len(sequence)))

canvas = tk.Canvas(splash, width=600, height=200)

canvas.pack()

sequence = [ImageTk.PhotoImage(img) for img in

ImageSequence.Iterator(Image.open('tenor.gif'))]

image = canvas.create\_image(110, 75, image=sequence[0])

animate(1)

splash.update()

splash.after(2175,splash.destroy)

splash.mainloop()

splash\_screen()

home = tt.ThemedTk()

home.get\_themes()

home.set\_theme('clearlooks')

home.title("type test")

ws = home.winfo\_screenwidth()

hs = home.winfo\_screenheight()

home.iconbitmap("iconimg.ico")

global height,width

width = (ws / 2) - (700 / 2)

height = (hs / 2) - (500 / 2)

home.geometry('%dx%d+%d+%d' % (700, 500, width,height))

bg = ImageTk.PhotoImage(file="bgimg.png")

global bg\_label

bg\_label = ttk.Label(home, image=bg)

bg\_label.place(relheight=1, relwidth=1)

def resize(a):

global bg, new\_bg, final\_bg

bg = Image.open("bgimg.png")

new\_bg = bg.resize((a.width, a.height), Image.ANTIALIAS)

final\_bg = ImageTk.PhotoImage(new\_bg)

bg\_label.config(image=final\_bg)

home.bind('<Configure>', resize)

def clean\_word(word):

word2 = str()

word = str(word)

for i in word:

if (65 <= ord(i) <= 90 or 97 <= ord(i) <= 122):

word2 += i

return word2

with open("words.txt", "r") as f:

wls = f.readlines()

def meaningful\_word\_giver(no, lis):

sel = random.randint(0, 999)

length = len(wls[sel])

if (no == length):

return wls[sel]

else:

return meaningful\_word\_giver(no, wls)

cur.execute("SELECT UserName from Users")

global user\_list

user\_list = list(set(cur.fetchall()))

for i in range(len(user\_list)):

user\_list[i] = clean\_word(user\_list[i])

if("Select User" not in user\_list):

cur.execute("INSERT INTO Users(UserName) VALUES('Select User')")

def page1():

def get\_random\_string(length):

letters = string.ascii\_lowercase

result\_str = ''.join(random.choice(letters) for i in range(length))

return (result\_str)

def avg\_size(l):

x = random.randint(0, 1)

if (x == 0):

return (l - random.randint(0, l // 2))

else:

return (l + random.randint(0, l // 2))

def word\_display(t, l, c, n):

if (t == 1 or t == 3):

c.set(meaningful\_word\_giver(avg\_size(l), wls).strip("\n"))

n.set(meaningful\_word\_giver(avg\_size(l), wls).strip("\n"))

if (t == 2):

c.set(get\_random\_string(avg\_size(l)))

n.set(get\_random\_string(avg\_size(l)))

global c

c = 0

def word\_check(event):

global c

if (text\_enter.get().strip() == display\_text\_cur.get()):

c += 1

text\_display\_label\_2.config(bg="green")

time.sleep(4)

text\_display\_label\_2.config(bg="blue")

display\_text\_prev.set(display\_text\_cur.get().strip("\n"))

display\_text\_cur.set(display\_text\_next.get().strip("\n"))

if (word\_type\_choice.get() == "Random words"): display\_text\_next.set(

get\_random\_string(word\_length\_choice.get()))

else:

display\_text\_next.set(

meaningful\_word\_giver(word\_length\_choice.get(), wls).strip("\n"))

else:

text\_display\_label\_2.config(bg="red")

text\_enter.delete(0, 'end')

tim\_var = tk.StringVar()

def inserter():

conn = sqlite3.connect("SpeedScores.db")

cur = conn.cursor()

cur.execute("INSERT INTO Users(UserName,Score) VALUES(?,?)", (clean\_word(user\_choice.get()), c))

conn.commit()

def countdown(t, time\_var=tim\_var):

while t:

mins, secs = divmod(t, 60)

timer = '{:02d}:{:02d}'.format(mins, secs)

tim\_var.set(timer)

time.sleep(1)

t -= 1

st\_page.update()

else:

tim\_var.set("times up")

res\_label = tk.Label(st\_page, text="Score is " + str(c) + " Words Per Minute", bg="yellow", fg="black")

res\_label.place(relx=0.5, rely=0.8, relwidth=0.3, relheight=0.1)

text\_enter.config(state="disabled")

if (user\_choice.get() != "Select User"):

inserter()

def begin\_time(var, index, mode):

if enter\_var.get():

countdown(time\_limit\_choice.get() \* 60)

def give\_val(d, v):

for i in d.keys():

if (d[i] == v):

return (i)

def ok():

tim\_var.set("time left")

time\_canvas = tk.Canvas(st\_page, bg='black')

time\_canvas.place(relx=0.1, rely=.28, relwidth=0.9, relheight=0.12)

time\_label=tk.Label(time\_canvas,bg="yellow",fg="blue",

textvariable=tim\_var)

time\_label.pack()

ok\_button.config(state="disabled")

text\_enter.config(state='normal')

if (word\_type\_choice.get() == "choose word type"):

word\_type\_choice.set("meaningful words")

word\_display(give\_val(option\_list\_1,

word\_type\_choice.get()),word\_length\_choice.get(), display\_text\_cur,display\_text\_next)

st\_page = tk.Toplevel(home, bg="black")

st\_page.iconbitmap("iconimg.ico")

st\_page.geometry("700x500")

st\_page.title("test your speed!")

frame = tk.Frame(st\_page, bg="#44bcd8")

frame.place(relx=0.1, rely=0.05, relwidth=0.8, relheight=0.2)

back\_button=ttk.Button(st\_page,text="goback",command=

st\_page.destroy)

back\_button.place(anchor="n", relx=0.96)

word\_type\_choice = tk.StringVar(st\_page)

word\_type\_choice.set("choose word type")

option\_list\_1 = {1: "meaningful words", 3: "meaningful words", 2: "Random words"}

drop\_1 = ttk.OptionMenu(st\_page, word\_type\_choice, \*option\_list\_1.values())

drop\_1.place(relx=0.1, rely=0.28, relheight=0.1, relwidth=0.25)

word\_length\_choice = tk.IntVar()

word\_length\_choice.set(6)

option\_list\_2 = [4, 4, 5, 6, 7, 8, 9]

drop2 = ttk.OptionMenu(st\_page, word\_length\_choice, \*option\_list\_2)

word\_length\_info\_label = ttk.Label(st\_page, text="choose average \nword length")

word\_length\_info\_label.place(relx=0.38, rely=0.28, relwidth=0.15, relheight=0.1)

drop2.place(relx=0.53, rely=0.28, relheight=0.1, relwidth=0.1)

time\_limit\_choice = tk.IntVar()

time\_limit\_choice.set(1)

time\_info\_label = ttk.Label(st\_page, text="chose time limit\n(minutes)")

time\_info\_label.place(relx=0.65, rely=0.28, relwidth=0.2, relheight=0.1)

option\_list\_3 = [1, 1, 2, 3]

drop3 = ttk.OptionMenu(st\_page, time\_limit\_choice, \*option\_list\_3)

drop3.place(relx=0.85, rely=0.28, relheight=0.1, relwidth=0.1)

display\_text\_prev = tk.StringVar()

text\_display\_label\_1 = tk.Label(frame, textvariable=display\_text\_prev, bg="white", font="times 28 bold")

text\_display\_label\_1.place(relx=0.2, rely=0.5, relheight=0.41, relwidth=0.3, anchor="c")

display\_text\_cur = tk.StringVar()

text\_display\_label\_2 = tk.Label(frame, textvariable=display\_text\_cur, font="times 28 bold", bg="blue", fg="yellow")

text\_display\_label\_2.place(relx=0.5, rely=0.5, relheight=0.41, relwidth=0.3, anchor="c")

display\_text\_next = tk.StringVar()

text\_display\_label\_3 = tk.Label(frame, textvariable=display\_text\_next, bg="white", font="times 28 bold")

text\_display\_label\_3.place(relx=0.8, rely=0.5, relheight=0.41, relwidth=0.3, anchor="c")

enter\_var = tk.StringVar()

enter\_var.trace\_add('write', begin\_time)

text\_enter = ttk.Entry(st\_page, textvariable=enter\_var, font="times 28 bold", state="disabled")

text\_enter.place(relx=0.5, rely=0.5, anchor="c", relwidth=0.35, relheight=0.15)

text\_enter.bind("<space>", word\_check)

ok\_button = ttk.Button(st\_page, text="Start", command=ok)

ok\_button.place(relx=0.7, rely=0.5, relwidth=0.2)

st\_page.update()

def test\_info():

info\_box=tk.Toplevel(st\_page)

info\_box.iconbitmap("iconimg.ico")

info\_box.geometry('%dx%d+%d+%d' % (450, 150, int(width\*(2/3)), int(height\*(2/3))))

info\_box.overrideredirect(True) info\_label=tk.Label(info\_box,bg="black",fg="yellow"

,text=testinfo) info\_label.place(relx=0,rely=0,relheight=1,relwidth=1) close\_button=tk.Button(info\_box,text="close",

command=info\_box.destroy)

close\_button.place(relx=0.9,rely=0.05)

info\_button=tk.Button(st\_page,command=test\_info,text="info") info\_button.place(relx=0.9,rely=0.187,relwidth=0.05,

relheight=0.05)

new\_user\_var = tk.StringVar()

def createcommand():

if (new\_user\_var.get()):

conn = sqlite3.connect("SpeedScores.db")

cur = conn.cursor()

cur.execute("INSERT INTO Users(UserName) VALUES(?)", (new\_user\_var.get(),))

conn.commit()

new\_user\_entry.delete(0, 'end')

frame\_1 = ttk.Frame(home, height=100, width=310)

frame\_1.place(relx=0.05, rely=0.05)

hello\_label = tk.Label(frame\_1, text="Hello", bg="#e9e8e1")

hello\_label.place(relx=0.08, rely=0.1, height=24, anchor="n", relwidth=0.2)

user\_choice = tk.StringVar()

user\_choice.set('Select User')

user\_drop = ttk.OptionMenu(frame\_1, user\_choice, \*user\_list)

user\_drop.place(anchor='n', relx=0.39, rely=0.1, relwidth=0.4)

open\_test\_button = ttk.Button(frame\_1, text="test your speed!", command=page1)

open\_test\_button.place(anchor="n", relx=0.8, rely=0.1, relwidth=0.4)

def plotter(a):

if (clean\_word(a) != "SelectUser"):

conn = sqlite3.connect("SpeedScores.db")

conn.row\_factory = lambda cursor, row: row[0]

cur = conn.cursor()

cur.execute(" SELECT Score FROM Users WHERE Username=(?)", (clean\_word(a),))

conn.commit()

x = cur.fetchall()

y=list(range(len(x)))

plt.plot(x,y, 'b--')

plt.xlabel("nth try")

plt.ylabel = ("WPM")

plt.xticks(list(range(0, len(x) + 1)))

plt.yticks(list(range(5, 100, 5)))

plt.title("User Progress")

plt.show()

def stats\_page(a):

statspage = tk.Toplevel(home)

statspage.overrideredirect(True)

close\_button = tk.Button(statspage, text="close", command=statspage.destroy)

close\_button.place(relx=0.88, rely=0.05)

statspage.geometry('%dx%d+%d+%d' % (350, 300, int(width\*(2/3)), int(height\*(2/3))))

user\_result = tk.StringVar()

user\_result.set("Choose User To See Performance")

choose\_menu = ttk.OptionMenu(statspage, user\_result, \*user\_list)

choose\_menu.place(relx=0.5, rely=0.1, anchor='n')

ok\_button = ttk.Button(statspage, text="View Progress", command=lambda: plotter(user\_result.get()))

ok\_button.place(relx=0.5,rely=0.8,anchor='s')

def deleteuser():

def deleter(page, name):

x = clean\_word(name)

if (x != "Select User"):

conn = sqlite3.connect("SpeedScores.db")

cur = conn.cursor()

cur.execute("DELETE FROM Users WHERE Username=(?)", (x,))

conn.commit()

page.destroy()

conn = sqlite3.connect("SpeedScores.db")

new\_page = tk.Toplevel(home)

new\_page.overrideredirect(True)

close\_button = tk.Button(new\_page, text="close", command=new\_page.destroy)

close\_button.place(relx=0.88, rely=0.05)

new\_page.geometry('%dx%d+%d+%d' % (350, 300, int(width\*(2/3)), int(height\*(2/3))))

new\_page.title("Delete User")

del\_var = tk.StringVar()

del\_var.set("Choose")

del\_user\_menu = ttk.OptionMenu(new\_page, del\_var, \*user\_list)

del\_user\_menu.place(relx=0.5, rely=0.2, relwidth=0.8, anchor='c')

ok\_button = tk.Button(new\_page, text="Delete!", command=lambda: deleter(new\_page, del\_var.get()))

ok\_button.place(relx=0.5, rely=0.3, anchor='c')

conn.commit()

def leaderboard(a, b):

table\_page = tk.Toplevel(home)

table\_page.overrideredirect(True)

close\_button = tk.Button(table\_page, text="close", command=table\_page.destroy)

close\_button.place(relx=0.88, rely=0.88)

table\_page.geometry('%dx%d+%d+%d' % (450, 300, int(width\*(2/3)), int(height\*(2/3))))

table = ttk.Treeview(table\_page, columns=(1, 2), show='headings')

table.pack()

table.heading(1, text="Name")

table.heading(2, text="Highscore")

def grapher():

conn = sqlite3.connect("SpeedScores.db")

cur = conn.cursor()

cur.execute("""SELECT UserName, MAX(Score) FROM Users GROUP BY UserName ORDER BY MAX(Score) DESC""")

x = cur.fetchall()

bar\_vals\_1 = []

bar\_vals\_2 = []

for i in x:

bar\_vals\_1.append(i[0])

bar\_vals\_2.append(i[1])

plt.bar(bar\_vals\_1, bar\_vals\_2)

print(bar\_vals\_1)

print(bar\_vals\_1)

plt.show()

conn.commit()

conn.close()

view\_graph\_button = ttk.Button(table\_page, text="view graph!", command=grapher)

view\_graph\_button.place(relx=0.5, rely=0.9, anchor="s")

try:

for i in range(len(a)):

table.insert('', 'end', values=(a[i], b[i]))

except Exception:

pass

frame\_2 = ttk.Frame(home, height=200, width=310)

frame\_2.place(relx=0.75, rely=0.05, anchor="n")

new\_user\_label = ttk.Label(frame\_2, text="Enter Name To Create User")

new\_user\_label.place(relx=0, rely=0.05)

new\_user\_entry = ttk.Entry(frame\_2, textvariable=new\_user\_var)

new\_user\_entry.place(relx=0.475, rely=0.05, relwidth=0.4)

create\_new\_user\_button = ttk.Button(frame\_2, text="Ok", command=createcommand)

create\_new\_user\_button.place(relx=0.88, rely=0.05, height=22)

view\_results\_button = ttk.Button(frame\_2, text="Track Progress", command=lambda: stats\_page(user\_list))

view\_results\_button.place(relx=0.01, rely=0.6, relwidth=0.4)

delete\_user\_button = ttk.Button(frame\_2, text="Delete User", command=deleteuser)

delete\_user\_button.place(relx=0.01, rely=0.2, relwidth=0.4)

leaderboard\_button = ttk.Button(frame\_2, text="Leaderboard", command=lambda: leaderboard(alluserlist, highscorelist))

leaderboard\_button.place(relx=0.01, rely=0.4, relwidth=0.4)

def page2():

tut\_page = tk.Toplevel(home, bg='black')

tut\_page.iconbitmap("iconimg.ico")

tut\_page.geometry("700x600")

tut\_page.title("Learn To Type")

close\_button = tk.Button(tut\_page, text="Go Back", command=tut\_page.destroy)

close\_button.place(relx=0.9, rely=0.05)

sample\_img=Image.open("touchtyping.png")

sample\_img=sample\_img.resize((500,200))

im = ImageTk.PhotoImage(sample\_img)

canvas = tk.Canvas(tut\_page, width=300, height=300)

canvas.place(anchor="c",relx=0.5,rely=0.4,relheight=0.3,relwidth=0.8)

canvas.create\_image(20, 0, image=im, anchor="nw")

def tut\_info():

info\_box=tk.Toplevel(tut\_page)

info\_box.iconbitmap("iconimg.ico")

info\_box.geometry('%dx%d+%d+%d' % (450, 150, int(width\*(2/3)), int(height\*(2/3))))

info\_box.overrideredirect(True)

info\_label=tk.Label(info\_box,bg="black",fg="yellow",text=tutinfo)

info\_label.place(relx=0,rely=0,relheight=1,relwidth=1) close\_button=tk.Button(info\_box,text="close"

,command=info\_box.destroy)

close\_button.place(relx=0.9,rely=0.05)

info\_button=tk.Button(tut\_page,command=tut\_info,text="info")

info\_button.place(relx=0.9,rely=0.1,relwidth=0.05,relheight=0.05)

def new\_word(self):

entry\_var.set('')

display\_text.set(meaningful\_word\_giver(5, wls).strip())

global display\_text

display\_text = tk.StringVar()

display\_text.set(meaningful\_word\_giver(5, wls).strip())

text\_display\_label\_ = tk.Label(tut\_page, textvariable=display\_text, font="times 28 bold", bg="blue", fg="yellow")

text\_display\_label\_.place(relx=0.5, rely=0.05, relheight=0.1, relwidth=0.25, anchor="n")

global keyboard\_layout

keyboard\_layout = tk.Label(tut\_page)

keyboard\_layout.place(relx=0.5,rely=0.7,anchor='c')

global labels

labels = ['q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p', 'a', 's', 'd', 'f', 'g', 'h', 'j', 'k', 'l',

'z', 'x', 'c', 'v', 'b', 'n', 'm']

global text\_entry

global entry\_var

entry\_var = tk.StringVar()

text\_entry = ttk.Entry(tut\_page, textvariable=entry\_var)

text\_entry.place(relx=0.5, rely=0.85, anchor='s')

text\_entry.bind("<space>", new\_word)

def key\_display():

global y

y = {}

varRow = 2

varColumn = 0

for i, label in enumerate(labels):

x = 'letter' + str(i)

y[x] = tk.Label(keyboard\_layout, text=label, width=4, bg="#3c4987", fg="#ffffff",activebackground="#ffffff",

activeforeground="#3c4987", relief='raised', padx=1, pady=1, bd=1)

y[x].grid(row=varRow, column=varColumn)

varColumn += 1

if (varColumn > 9 and varRow == 2):

varColumn = 0

varRow += 1

if (varColumn > 8 and varRow == 3):

varColumn = 0

varRow += 1

if (varColumn > 8 and varRow == 4):

varColumn = 0

varRow += 1

global p

p = 0

def overall():

def timed\_labeller(key, keyn, dec):

if dec == "ok":

t = 1

y[keyn].config(bg="green", fg="white")

while t:

time.sleep(2)

t -= 1

else:

y[key].config(bg="#3c4987", fg="#ffffff")

y[keyn].config(bg="green", fg="white")

if len(str(entry\_var.get())) == len(display\_text.get()):

y[key].config(bg="#3c4987", fg="#ffffff")

else:

if (len(entry\_var.get()) > 0):

y[key].config(bg="red")

global p

if (p == 0):

r = display\_text.get()[0]

q = labels.index(r)

y["letter" + str(q)].config(bg="green", fg="white")

else:

y["letter" + str(q)].config(bg="#3c4987", fg="#ffffff")

p += 1

tut\_page.update()

def colour\_scheme(var, indx, mode):

if len(entry\_var.get()) > len(display\_text.get()):

entry\_var.set(

entry\_var.get()[:len(display\_text.get())])

if len(str(entry\_var.get())) == 0:

a = display\_text.get()[len(str(entry\_var.get()))]

an = display\_text.get()

[len(str(entry\_var.get())) + 1]

elif len(str(entry\_var.get())) == len(display\_text.get()):

a = display\_text.get()

[len(str(entry\_var.get())) - 1]

an = display\_text.get()

[len(str(entry\_var.get())) - 1]

else:

a = display\_text.get()

[len(str(entry\_var.get())) - 1]

an = display\_text.get()[len(str(entry\_var.get()))]

b = labels.index(a)

bn = labels.index(an)

global key, keyn

key = 'letter' + str(b)

keyn = 'letter' + str(bn)

try:

if (entry\_var.get()[-1:] == display\_text.get()[len(str(entry\_var.get())) - 1]):

timed\_labeller(key, keyn, "ok")

else:

if (len(entry\_var.get()) > 0):

timed\_labeller(key, keyn, "no")

except Exception:

pass

tut\_page.update()

entry\_var.trace\_add('write', colour\_scheme)

overall()

key\_display()

tut\_page.mainloop()

tutorial\_button = ttk.Button(frame\_1, text="Learn To Type!", command=page2)

tutorial\_button.place(anchor="s", relx=0.5, rely=0.9, relwidth=0.5)

quit\_button = ttk.Button(home, text="Quit!", command=lambda: home.destroy())

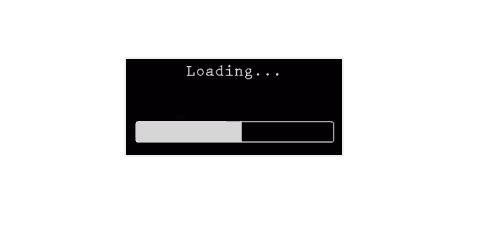
quit\_button.place(relx=0.9, rely=0.9)

conn.commit()

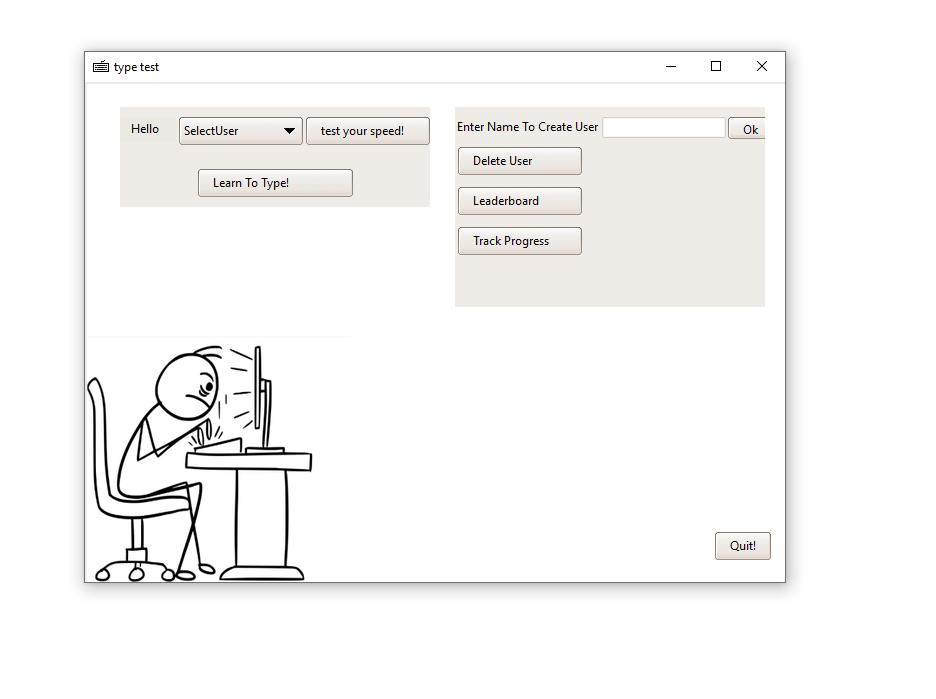
conn.close()

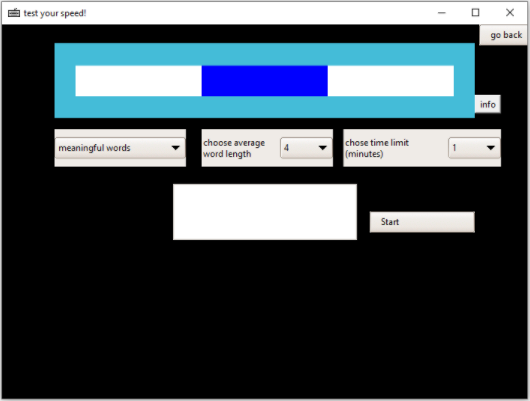
home.mainloop()

# Output (Screenshots)

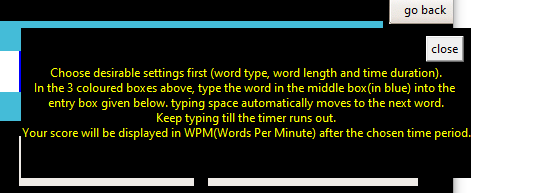


Loading Splash Screen of the Tool

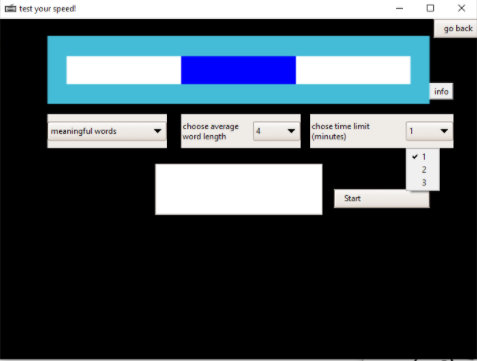
Main WIndow of the Tool



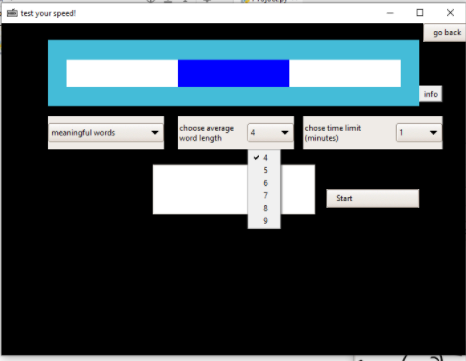
Typing Speed Calculator Page(inactive)



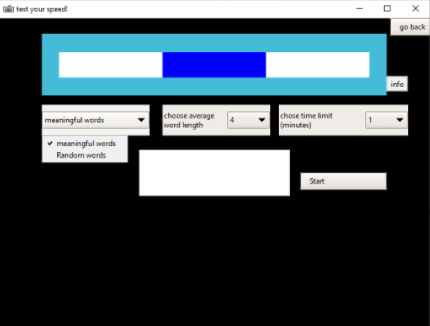
Typing Speed Calculator Info Box



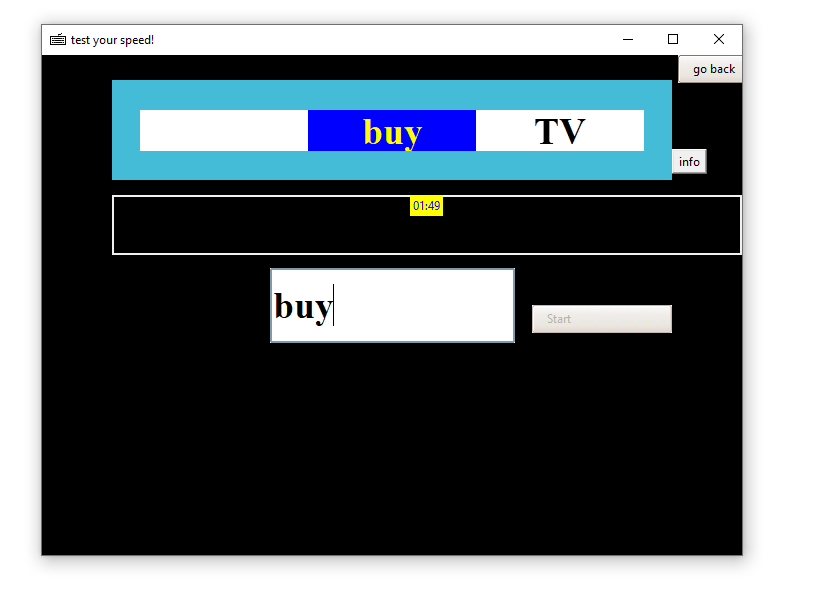
Time Changing Option

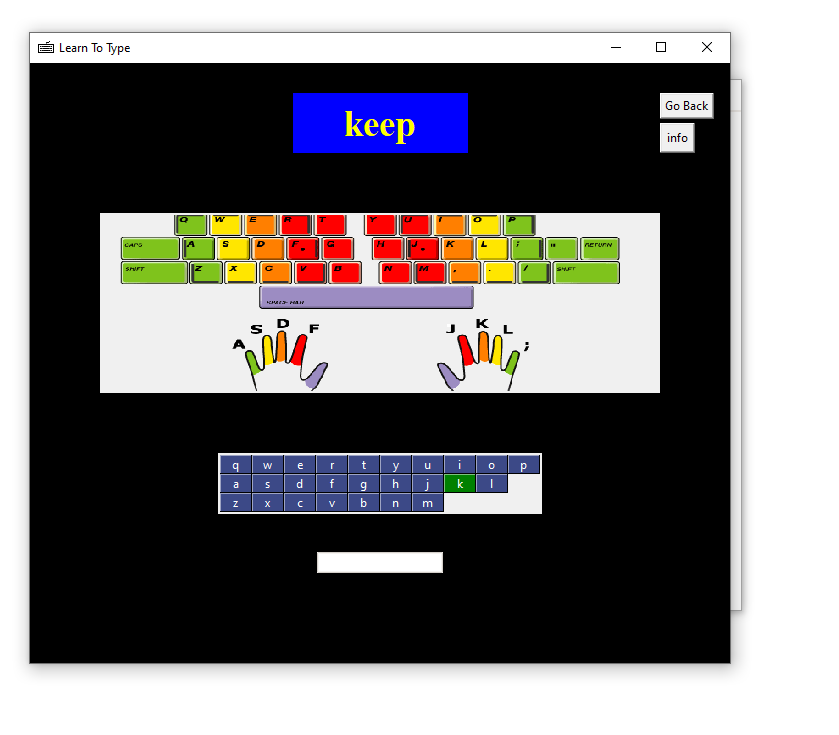


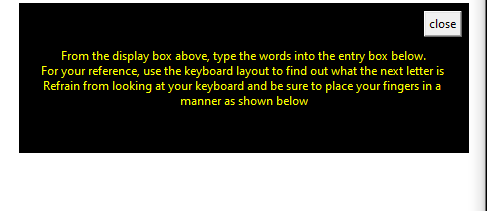
Word Length Changing Options



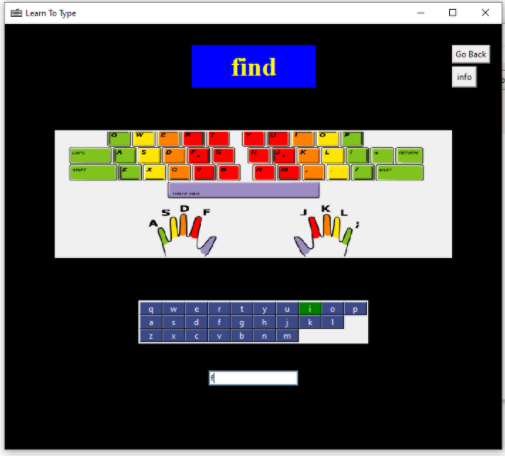
Word Type Changing Options

Typing Speed Calculator(active)

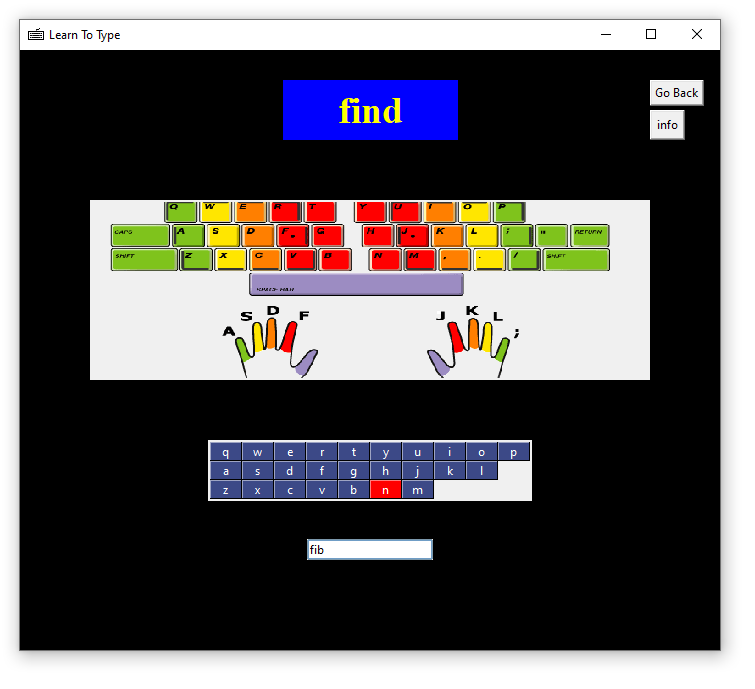
Typing Tutorial Window



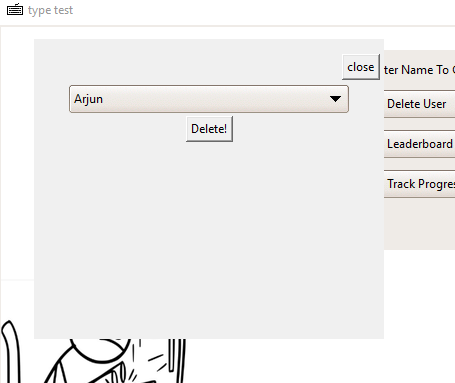
Typing Tutorial Helpbox



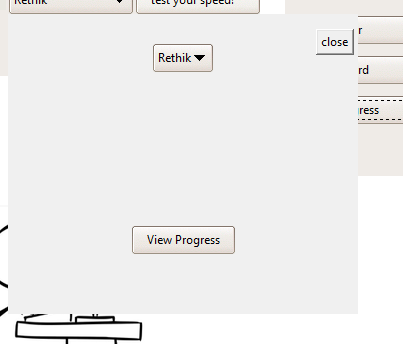
Typing Tutorial(active)(i)

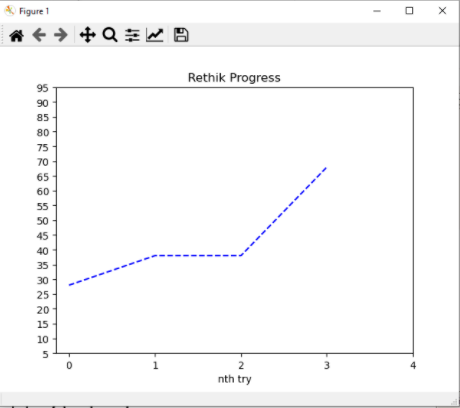


Typing Tutorial(active)(ii)

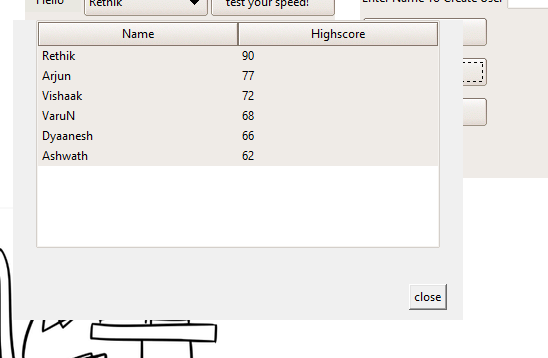


Delete User Box

View Progress Box



Progress Graph



HighScores

# 

# Leaderboard Graph

# Conclusion

The Typing Speed Calculator has been created successfully using Python programming which is integrated with SQL which acts as the database for the backend. Various modules and packages for both python and SQL have been used in this project for a smooth gaming experience for the end user which is the motto of this project.

Vigorous coding and debugging of the program had made it a user-friendly tool. This base code can be further developed to have competitive typing tests with real time players and even computers, which is beyond the scope of this game.

# Bibliography

Books:

Class XI NCERT textbook

Class XII NCERT textbook

Sumitha Arora textbook

Preeti Arora textbook

Websites:

[www.python.org](https://www.python.org/)

[www.stackoverflow.com](https://stackoverflow.com/)

[www.geeksforgeeks.org](http://www.geeksforgeeks.org)

[www.w3schools.com](http://www.w3schools.com)

[www.tutorialspoint.com](https://www.tutorialspoint.com/)

[www.udemy.com](https://www.udemy.com/)

[www.youtube.com](http://www.youtube.com)