PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100 Feet Ring Road, BSK III Stage, Bengaluru-560 085

Department of Computer Science & Engineering

Title: VISUALIZE DATA

Team member details:

PES1UG21CS494: Rohan J

PES1UG21CS542 – Sathvik Malgikar

PES1UG21CS607 – Sonupatel A L

Abstract

VISUALIZE DATA is a simple tool that allows you to enter the input statistical data and view it in terms of graphics and analyze it in a more understandable manner. We can input multiple inputs as and their labels and the program generates a pie chart and a bar graph for the given statistical data. This makes understanding data statistics more intuitive and simple for the users.

Statistics is always best understood when we can actually see what the terms mean and for this purpose we can use this program. It is self explanatory and requires no prerequisite knowledge from the users' side.

This program has been made in a very flexible way and hence we can always add other functionality like other forms for graphs, etc., whenever required. The program is built in python and uses [1] Tkinter for the GUI part of it. Modules imported are [2] matplotlib for the calculations part, [3] Pygame for music. [4] PIL for background images and GIFs.

Table of Contents

Introduction

The main inspiration for making this program was the lack of any viable solution for anyone looking for any easy way for converting any statistical data into an easy to understand and simple model based on data . Hence this program solves such problems for anybody even with just the very basic understanding of how to operate a computer as it uses GUI anybody without any technical knowledge can still use this tool to get his job done. This program is graphical in nature and hence it very intuitive in nature.

Design / Implementation

The program has been implemented keeping in mind the limited memory resources during execution, hence care has been taken that the program runs with as minimum as possible load on the computer, hence this program has been designed in a decentralized way.

The main program file VISUALIZEDATA.py contains the core processing codes that displays the welcome window providing the instruction to the users regarding the buttons and the usage of labels and entities. Then after receiving the data from the user and clicking the pie chart or bar graph button the respective actions are performed through two other modules called barcreator.py and piecreator.py which pass the data to matplotlib[2] for generating graphs.

PIL[4] module has been utilized to render a background image to the interface page and relevant GIFs have been added in order the make the project look more appealing to the users.

Care has been taken considering that an average person will be using this program and might make some invalid inputs. Hence there are systems to Grey out the buttons if the provided inputs are not valid. This ensures that the program is reliable and does not crash easily.

The main code :

```
from distutils import command

import imp
from tkinter import *
from turtle import bgcolor, color
from matplotlib import image

from matplotlib.pyplot import close, text
from numpy import var
from pyparsing import And, col
import piecreator
from PIL import ImageTk, Image
import pygame
import barcreator


def piechart():
    global vallist
    vallist=list(map(lambda n : n.get(),list_e))
    piecreator.pialg(vallist,list_l)
pygame.mixer.init()
pygame.mixer.music.load('Funky.mp3')
pygame.mixer.music.play()
def bargraph():
    global vallist
    vallist=list(map(lambda n : n.get(),list_e))
    barcreator.baralg(vallist)


no_of_items=0
```

```
def getter(from_return=False):
    global no_of_items
    if from_return :
        if input_in_e1.get()=='':
            pass
        else :
            no_of_items=int(input_in_e1.get())
            nextone()
    else:
        no_of_items=int(input_in_e1.get())
        nextone()
welcome=Tk()
welcome.state('zoomed')
welcome.geometry('800x600')
welcome.title('VISUALISE DATA')
textstring=open('textstring.txt','r')
Label(welcome,text=textstring.read(),font=('Helvetica','16')).pack()
Button(welcome,text='PROCEED',font=50,command=lambda :
welcome.destroy(),height=4,width=17,bg='green').pack(pady=38)
gfl1=Label(welcome,image='')
gfl1.pack(side=LEFT)
gfl2=Label(welcome,image='')
gfl2.pack(side=RIGHT)

piefile='pie.gif'
pieimage=Image.open(piefile)
piegiffframes=pieimage.n_frames
pie_im=[PhotoImage(file=piefile,format=f'gif -index {i}') for i in range(piegiffframes)]
count=0
anim=None
```

```
barfile='bar.gif'
barimage=Image.open(barfile)
bargiffframes=barimage.n_frames
bar_im=[PhotoImage(file=barfile,format=f'gif -index {i}') for i in range(bargiffframes)]
# print(bargiffframes,piegiffframes)
def animation(count=0):
    global anim
    pim=pie_im[count]
    bim=bar_im[count]
    gfl1.configure(image=pim)
    gfl2.configure(image=bim)
    count+=1
    if count==piegiffframes:
        count=0
    anim=welcome.after(40,lambda : animation(count))
animation()
welcome.mainloop()
textstring.close()
root=Tk()


root.geometry("1200x950")
root.title('VISUALISE DATA')
img = ImageTk.PhotoImage(Image.open("back.webp"))


label1 = Label( root, image = img)
label1.place(x = 0, y = 0)
```

```
l1=Label(root,text='Number of items : ',font=100)
l1.grid(row=1,column=1,padx=60,pady=20)


input_in_e1 = StringVar(root)


e1=Entry(root, textvariable=input_in_e1)
e1.grid(row=1,column=2)
b1=Button(root,text='GO',command=getter)
b1.grid(row=1,column=3,padx=30)
def to_check():
    if input_in_e1.get().isdigit() :
        if len(input_in_e1.get()) > 0:
            b1.configure(state=NORMAL)
    else :
        b1.configure(state=DISABLED)



input_in_e1.trace('w', lambda *args : to_check())
to_check()

def nextone(save=None):
    def checking_last_field():
        if list_sv[-1].get()=='' :
            bargraph_button.configure(state=DISABLED)
            piechart_button.configure(state=DISABLED)
        else:
            bargraph_button.configure(state=NORMAL)
            piechart_button.configure(state=NORMAL)
```

```
    global list_l,list_e
    list_l=[]
    list_e=[]
    list_sv=[StringVar(root) for i in range(no_of_items)]
    for i in range(no_of_items):
        if save and i+1<no_of_items:
            list_sv[i].set(save[i].get())
        else:
            list_sv[i].set(0)

        list_l.append(Entry(root,font=50))
        list_l[i].insert(0, string=f'item {i+1} :')
        list_e.append(Entry(root,textvariable=list_sv[i]))

        list_l[i].grid(row=i+2,column=1,pady=10)
        list_e[i].grid(row=i+2,column=2,padx=7,pady=10)

    piechart_button=Button(root,text='CREATE PIECHART',command=piechart)
    piechart_button.grid(row=1,column=4,padx=30)
    bargraph_button=Button(root,text='CREATE BARGRAPH',command=bargraph)
    bargraph_button.grid(row=2,column=4,padx=30)
    checking_last_field()
    Label(root,font=50,text='Click enter to reset all the
values',bg='green').grid(pady=10,row=70,column=1)
    list_sv[-1].trace('w', lambda * args: checking_last_field())
    def addone():
        global no_of_items
        no_of_items+=1
        input_in_e1.set(int(input_in_e1.get())+1)
        nextone(list_e)
```

```
    add_entry=Button(root,text='Add another entry',command=addone)
    add_entry.grid(column=1,row=30)

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

root.mainloop()
```

Code responsible for bar graph:

```
from matplotlib import pyplot
def baralg(vl):
    itemlist=list(range(1,len(vl)+1))
    vl=list(map(float,vl))
    pyplot.bar(itemlist,vl)
    return pyplot.show()
```

Code responsible for pie chart:

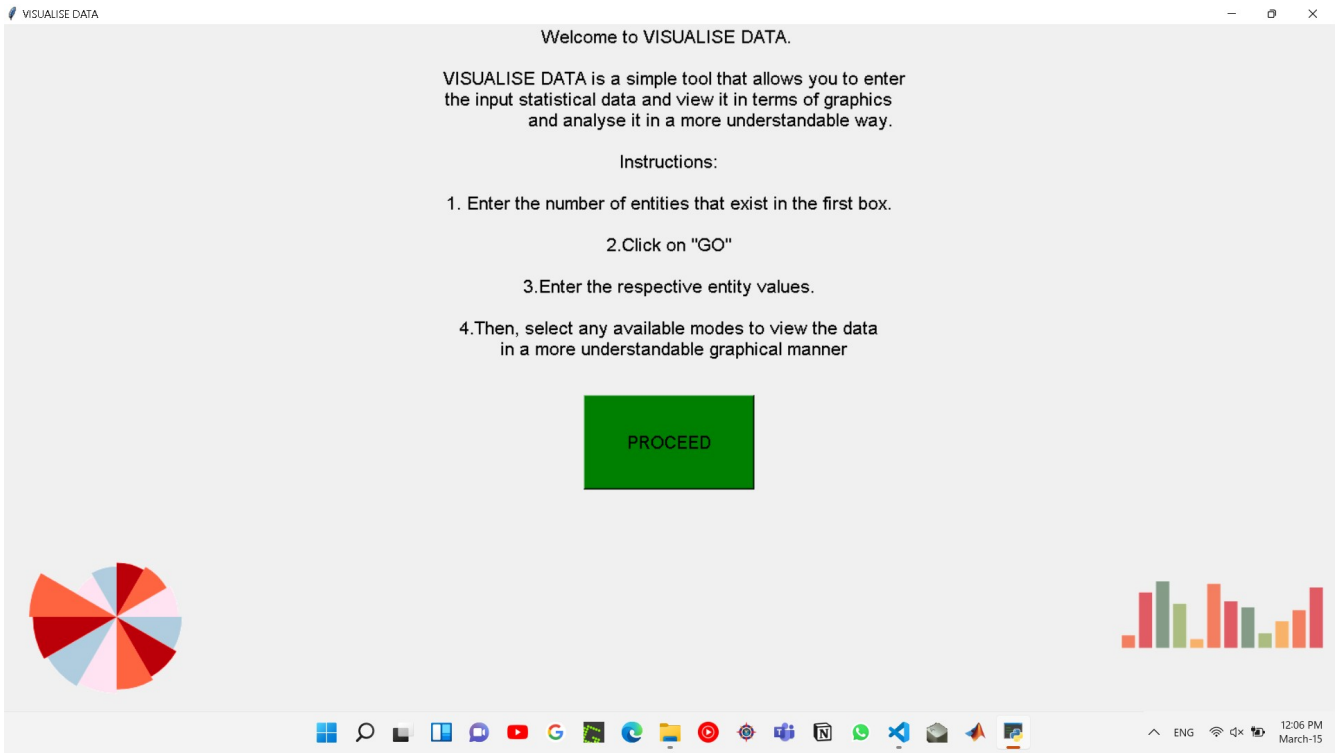
```
from functools import reduce
from matplotlib import pyplot
def pialg(vl,list_l):
    itemlist=[]
    for i in range(len(vl)):
        itemlist.append(list_l[i].get())
    pyplot.pie(vl,labels=itemlist)
    return pyplot.show()
```

Testing

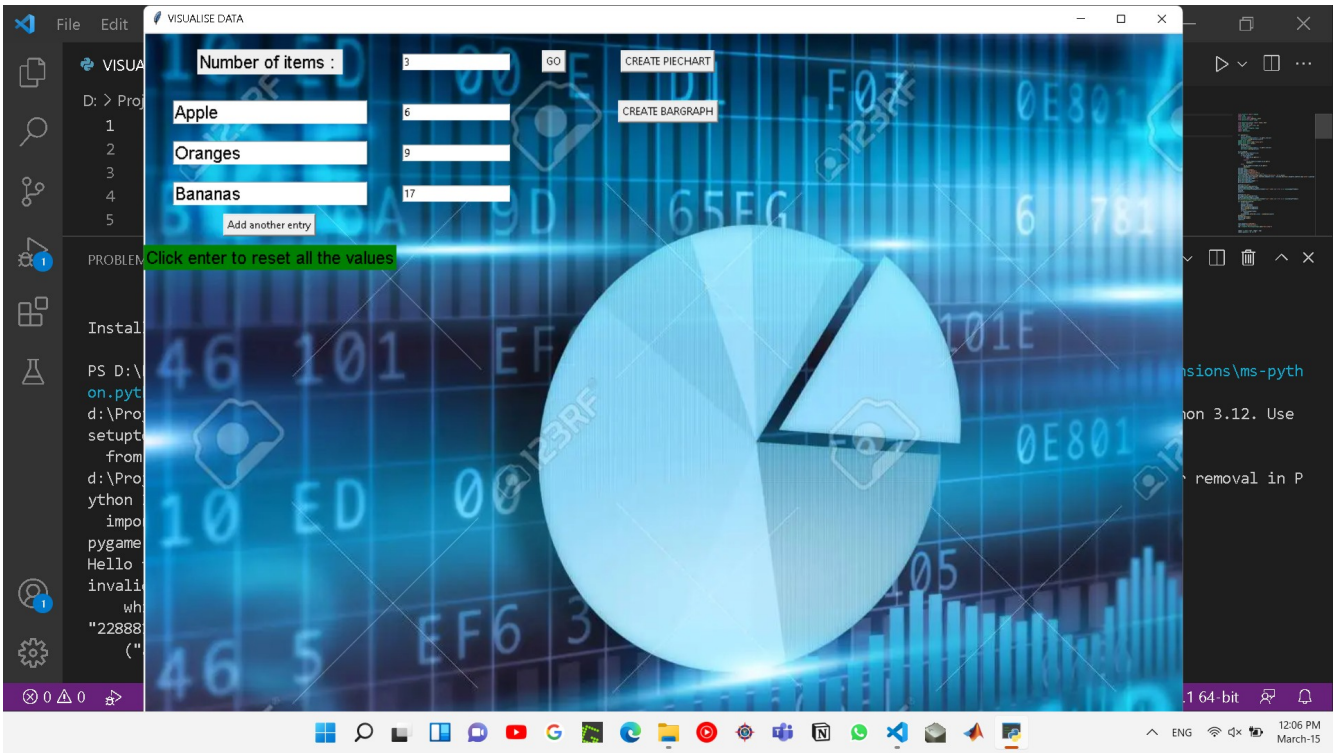
For the testing part since every program has its inbuilt problems for the debugging part we had to go through a lot of problems the program had errors whenever the inputs were invalid so in order to fix this issue we had to make procedures that rejected invalid inputs this was executed by having the go button greyed out or disabled whenever the inputs are not valid this was implemented with the help of trace function whenever the inputs are not valid like for example irregular entries like strings in case of integers due to users mistakes, the math part would crash and this needed to be fixed .

The bugs were mostly fixed by trial and error method there was a bug in which the bar graph showed irrelevant and improper representation of data then the issue was found out to be a problem in passing of arguments the there was a mistake in the logic and the function would take default arguments even when the user had passed actual arguments because of the default arguments the output should always stay the same and this was fixed by looking into the program.

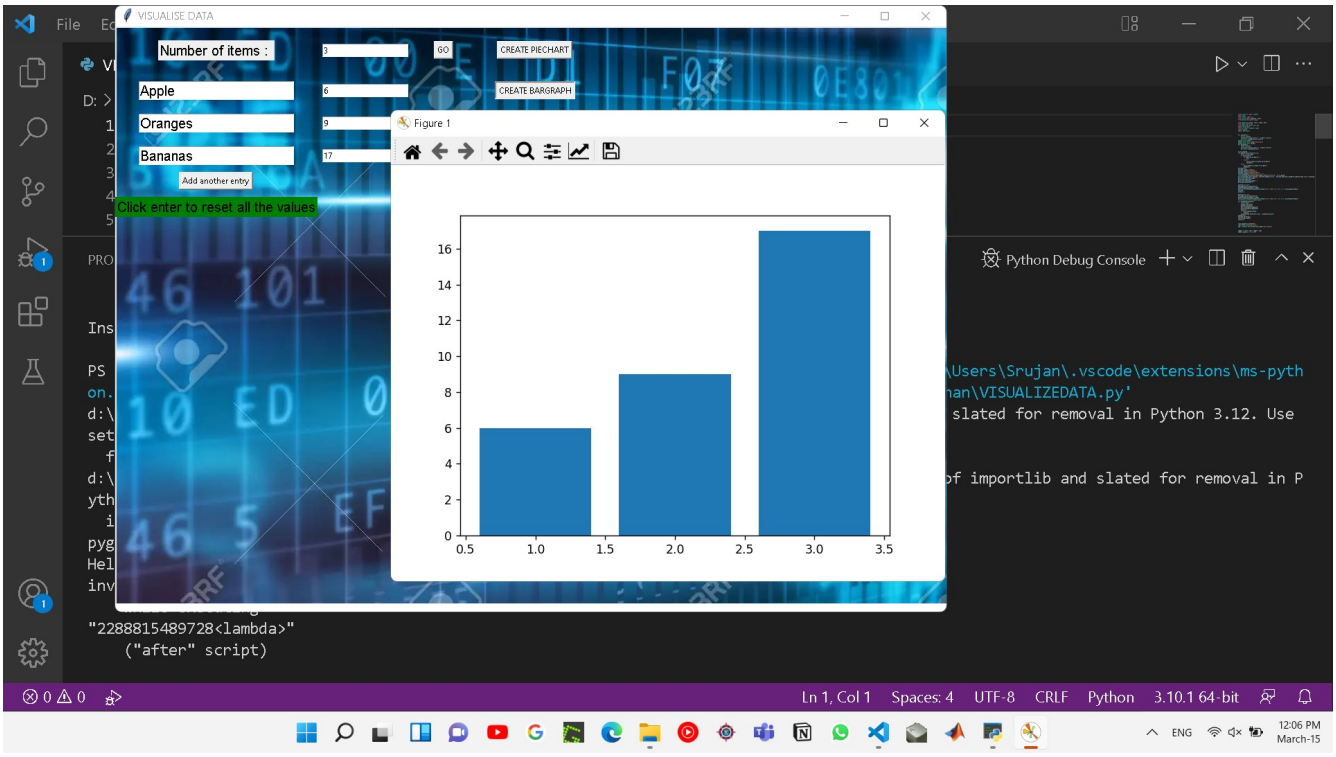
Result and Analysis



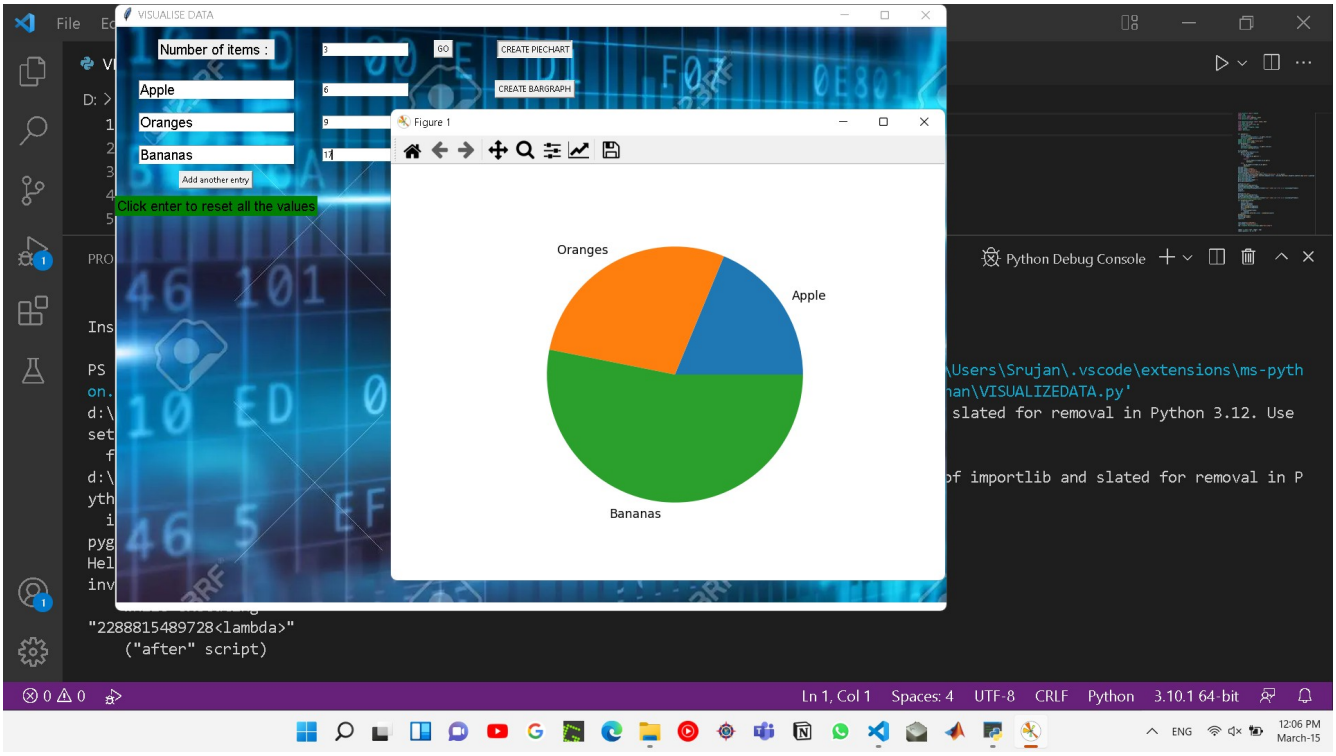
This is the main welcome screen which gives instructions to the user on how to use the program.



This Window is responsible for taking the inputs from the user and labels and entries. It allows the user for two options Either to make a bar graph or a pie chart



This is what shows up when the user creates a bar graph.



This is what Shows up when the user creates a pie chart also it has the respective labels given by the user to the corresponding data .

Conclusions and further enhancements

In conclusion, we would like to say that out project did meet up to our expectations and produced satisfactory results.

Regarding future enhancements, since our project is limited to just 2 formats of graphs i.e., pie charts and bar graphs, we think of including various other formats of visual representations into our project. Also we can try implementing it in such a way that it can display more than one graphs simultaneously. As of the design part, we can make the interface even more appealing to the users than it is now.

Modules used:

1. Matplotlib[2]

2. Tkinter[1]

3. Pygame[3]

4. PIL [4]

References :

1. Stack Overflow - Where Developers Learn, Share, & Build Careers [1] [2] [3]

2. https://www.youtube.com/c/CodeWithHarry [3] [4]

3 . https://www.youtube.com/c/Telusko [2]

4. https://www.youtube.com/c/Codemycom [1] [4]

5. https://github.com/ [2] [3]

6. https://www.geeksforgeeks.org/ [1] [4]

7. https://docs.python.org/3/reference/ [2]