```
from tkinter import *
import tkinter as tk
from tkinter import ttk
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import poisson,norm,uniform
import os
import sys



class MyWindow:
    def __init__(self, window):
        # window = Tk()
        self.lbl = Label(window, text="Probability Distribution Visualizer", fg='Black',
font=("Helvetica", 16))
        self.lbl.place(x=70, y=10)

        # ********************************************
        ttk.Label(window, text="Select Distibution :",
                font=("Times New Roman", 10)).grid(column=0,
                                            row=15, padx=50, pady=70)

        self.n = tk.StringVar()
        self.DistChoosen = ttk.Combobox(window, width=27,
                                textvariable=self.n)
        self.DistChoosen.place(x=50, y=30)

        # Adding combobox drop down list
        self.DistChoosen['values'] = ('Poisson Distribution',
                            'Normal Distribution',
                            'Uniform Distribution')

        self.inputStr = self.n.get()
        # performCalculations(inputStr)

        self.DistChoosen.grid(column=1, row=15)

        self.DistChoosen.current(0)
        # ********************************************
        self.controlBut =  ttk.Button(window, text="Select", command=self.select)
```

```
        self.controlBut.place(x=100, y=150)

        self.restartBut = ttk.Button(window, text="restart code", command=self.restart)
        self.restartBut.place(x=300, y=150)



        self.b1 = Button(window, text='Compute and Find Graph', command =
self.performCalculations)
        self.b1.place(x=150, y=350)

        self.inputval = StringVar()
        self.errorLabel = Label()
        self.inputentry = Entry(window, textvariable=self.inputval , bd = 5 , width =
40)
        # inputentry.place(x = 80 , y = 100)
        # self.inputentry.place(x = 200 , y = 400)
        # self.errLabel = Label(window, text='Error (if any)')




    def restart(self):
        python = sys.executable
        os.execl(python, python, *sys.argv)
        # window.mainloop()

    def assign_lamda(self , v):
        self.lbda = v
    def select(self):
        if (self.n.get() == "Poisson Distribution"):

            self.sc = tk.Scale(window, label='Slide for value of lambda', from_=0, to=50,
orient=tk.HORIZONTAL, length=200, showvalue = True,
                       tickinterval=0, resolution=0.01, command=self.assign_lamda)

            self.sc.set(10)
            self.sc.place(x = 100 , y = 250)
```

```
    elif (self.n.get() == "Uniform Distribution"):

        self.uppperLim = Entry(window, bd=5)
        self.lowerLim = Entry(window, bd=5)
        self.uppperLim.place(x=150, y=200)
        self.lowerLim.place(x=150, y=240)
        self.lbl1 = Label(window, text='Enter upper Limit')
        self.lbl2 = Label(window, text='Enter lower Limit')

        self.lbl1.place(x=20, y=200)
        self.lbl2.place(x=20, y=240)
        # self.plot_normal()

    elif (self.n.get() == "Normal Distribution"):
        self.mean = Entry(window, bd=5)
        self.std_deviation = Entry(window, bd=5)
        self.x_cordinte = Entry(window, bd=5)
        self.mean.place(x=50, y=250)
        self.std_deviation.place(x=230, y=250)
        self.x_cordinte.place(x=100, y=300)
        self.lbl3 = Label(window, text='μ')
        self.lbl4 = Label(window, text='σ')
        self.lbl5 = Label(window, text='x')

        self.lbl3.place(x=20, y=250)
        self.lbl4.place(x=210, y=250)
        self.lbl5.place(x=80, y=300)

def plot_p(self):
    lam = float(self.lbda)
    if (lam < 0 ):
        message = "Entered value is less than 0"
        error = Label(window, text=message)
        error.place(x=80, y=400)
    else:
        x = np.linspace(0, 40, 100)
        y = poisson(lam).pmf(x)
        plt.plot(x, y)
        plt.title('Poisson Distribution')
        plt.xlabel('x')
        plt.ylabel('P(x)')
```

```
        plt.show()

    def plot_normal(self):
        m = float(self.mean.get())
        s = float(self.std_deviation.get())
        x_coor = float(self.x_cordinte.get())
        x = np.linspace(m - 10, m + 10, 100)
        y = norm.pdf(x, m, s)
        plt.fill_between(x, y, where=x < x_coor)
        plt.plot(x, y)
        plt.title('Normal Probability Distribution')
        plt.xlabel('x')
        plt.ylabel('P(x)')
        plt.show()

    def plot_uniform(self):
        # lwr =
        upper_limit = float(self.uppperLim.get())
        lower_limit = float(self.lowerLim.get())
        if(upper_limit < lower_limit):

            message = "Error : Upper limit less than Lower Limit"
            error = Label(window, text=message)
            error.place(x=120, y=400)
            # error.config(text="" + message)

        else:

            x = np.arange(0,10,0.1)
            print(x)
            y = uniform.cdf(x,lower_limit, upper_limit)
            plt.plot(x, y)
            plt.title('Uniform Probability Distribution')
            plt.xlabel('x')
            plt.ylabel('P(x)')
            plt.show()


    def performCalculations(self):
        if (self.n.get() == "Poisson Distribution"):
            self.plot_p()
```

```
        elif (self.n.get() == "Normal Distribution"):
            self.plot_normal()

        elif (self.n.get() == "Uniform Distribution"):
            self.plot_uniform()

        else:
            print("Not a valid input")




window=Tk()
# bgimg= tk.PhotoImage(file = "finalmf.gif")
mywin=MyWindow(window)
window.title('Ashitosh Phadatare GUI')
window.geometry("500x500+10+10")
window['bg']='#6699CC'
window.mainloop()
```