1.1 SIMPLE CALCULATOR

Create a basic calculator that can perform basic arithmetic operations such as addition, subtraction, multiplication, and division.using functions

In [ ]:
```
def sum():
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))
    return num1 + num2


def sub():
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))
    return num1 - num2


def multi():
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))
    return num1 * num2


def div():
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))
    return num1 / num2


if __name__ == "__main__":
    while True:
        choice = input(
            " 1. Addition\n 2. Subtract\n 3. Multiplication\n 4. Divide\n 5. Exit\
        )

        if choice == "1":
            print("The Addition is:", sum())

        elif choice == "2":
            print("The subtract is:", sub())

        elif choice == "3":
            print("The Multiplication is:", multi())

        elif choice == "4":
            print("The divide is:", div())

        elif choice == "5":
            break

        else:
            print("Please Enter correct choice!")
```

1.2 SIMPLE CALCULATOR WITH GUI

In [ ]:
```
from tkinter import *
```

```
def button_press(num):

    global equation_text

    equation_text = equation_text + str(num)

    equation_label.set(equation_text)

def equals():

    global equation_text

    try:

        total = str(eval(equation_text))

        equation_label.set(total)

        equation_text = total

    except SyntaxError:

        equation_label.set("syntax error")

        equation_text = ""

    except ZeroDivisionError:

        equation_label.set("arithmetic error")

        equation_text = ""

def clear():

    global equation_text

    equation_label.set("")

    equation_text = ""


window = Tk()
window.title("Calculator program")
window.geometry("500x500")

equation_text = ""

equation_label = StringVar()

label = Label(window, textvariable=equation_label, font=('consolas',20), bg="white
label.pack()

frame = Frame(window)
frame.pack()

button1 = Button(frame, text=1, height=4, width=9, font=35,
                 command=lambda: button_press(1))
button1.grid(row=0, column=0)

button2 = Button(frame, text=2, height=4, width=9, font=35,
                 command=lambda: button_press(2))
button2.grid(row=0, column=1)

button3 = Button(frame, text=3, height=4, width=9, font=35,
```

```
                  command=lambda: button_press(3))
button3.grid(row=0, column=2)

button4 = Button(frame, text=4, height=4, width=9, font=35,
                  command=lambda: button_press(4))
button4.grid(row=1, column=0)

button5 = Button(frame, text=5, height=4, width=9, font=35,
                  command=lambda: button_press(5))
button5.grid(row=1, column=1)

button6 = Button(frame, text=6, height=4, width=9, font=35,
                  command=lambda: button_press(6))
button6.grid(row=1, column=2)

button7 = Button(frame, text=7, height=4, width=9, font=35,
                  command=lambda: button_press(7))
button7.grid(row=2, column=0)

button8 = Button(frame, text=8, height=4, width=9, font=35,
                  command=lambda: button_press(8))
button8.grid(row=2, column=1)

button9 = Button(frame, text=9, height=4, width=9, font=35,
                  command=lambda: button_press(9))
button9.grid(row=2, column=2)

button0 = Button(frame, text=0, height=4, width=9, font=35,
                  command=lambda: button_press(0))
button0.grid(row=3, column=0)

plus = Button(frame, text='+', height=4, width=9, font=35,
                  command=lambda: button_press('+'))
plus.grid(row=0, column=3)

minus = Button(frame, text='-', height=4, width=9, font=35,
                  command=lambda: button_press('-'))
minus.grid(row=1, column=3)

multiply = Button(frame, text='*', height=4, width=9, font=35,
                  command=lambda: button_press('*'))
multiply.grid(row=2, column=3)

divide = Button(frame, text='/', height=4, width=9, font=35,
                  command=lambda: button_press('/'))
divide.grid(row=3, column=3)

equal = Button(frame, text='=', height=4, width=9, font=35,
                  command=equals)
equal.grid(row=3, column=2)

decimal = Button(frame, text='.', height=4, width=9, font=35,
                  command=lambda: button_press('.'))
decimal.grid(row=3, column=1)

clear = Button(window, text='clear', height=4, width=12, font=35,
                  command=clear)
clear.pack()

window.mainloop()
```

2.1 CURRENCY CONVERTER

Build a program that can convert currency from one form to another using the latest exchange rates

In [ ]:
```
!pip install CurrencyConverter
from currency_converter import CurrencyConverter
a = CurrencyConverter()
print(a.convert(1,"USD","INR"))
```

2.2 CURRENCY CONVERTER WITH GUI

In [ ]:
```
!pip install CurrencyConverter

from currency_converter import CurrencyConverter
import tkinter as tk

a = CurrencyConverter()

window = tk.Tk()
window.geometry("500x360")


def clicked():
    amount = int(e1.get())
    curr1 = e2.get()
    curr2 = e3.get()
    data = a.convert(amount,curr1,curr2)
    l5 = tk.Label(window,text=data,font = "Times 25 bold").place(x=100,y=290 )

l1 = tk.Label(window,text="Currency Converter",font = "Times 25 bold").place(x = 1

l2 = tk.Label(window,text="Enter Amount Here: ",font = "Times 18 ").place(x = 50,y
e1 = tk.Entry(window)

l3 = tk.Label(window,text="Enter Currency",font = "Times 18 ").place(x = 50, y = 1
e2 = tk.Entry(window)

l4 = tk.Label(window,text="Enter Req. Currency",font = "Times 18 ").place(x = 50,
e3 = tk.Entry(window)

b1 = tk.Button(window,text="Convert",font = "bold",command = clicked).place(x = 23

e1.place(x=300,y=90)
e2.place(x=300,y=140)
e3.place(x=300,y=190)

window.mainloop()
```

*2.2 CURRENCY CONVERTER WITH GUI (ANOTHER)

In [ ]:
```
import tkinter as tk
from tkinter import *
import tkinter.messagebox

root = tk.Tk()
```

```
root.title("GUI : Currency Conversion")

Tops = Frame(root,bg = '#663300',pady = 2, width =1850, height = 100, relief = "ri
Tops.grid(row=0,column=0)

headlabel = tk.Label(Tops,font=('lato black', 19,'bold'), text = '      Pypower Pr
headlabel.grid(row=1, column=0,sticky=W)

variable1 = tk.StringVar(root)
variable2 = tk.StringVar(root)

variable1.set("currency")
variable2.set("currency")

def RealTimeCurrencyConversion():
    from forex_python.converter import CurrencyRates
    c=CurrencyRates()

    from_currency = variable1.get()
    to_currency = variable2.get()

    if (Amount1_field.get()==""):
        tkinter.messagebox.showinfo("Error !!","Amount Not Entered.\n Please a val

    elif (from_currency=="currency" or to_currency=="currency"):
        tkinter.messagebox.showinfo("Error !!","Currency Not Selected.\n Please se

    else:
        new_amt = c.convert(from_currency,to_currency,float(Amount1_field.get()))
        new_amount = float("{:.4f}".format(new_amt))
        Amount2_field.insert(0, str(new_amount))

def clear_all() :
        Amount1_field.delete(0, tk.END)
        Amount2_field.delete(0, tk.END)


CurrenyCode_list = ["INR", "USD", "CAD", "CNY", "DKK", "EUR"]


root.configure(background = '#e6e5e5')
root.geometry("700x400")

Label_1 =Label(root, font=('lato black', 27,'bold'), text="",padx=2,pady=2, bg="#e
Label_1.grid(row=1, column=0,sticky=W)


label1 = tk.Label(root,font=('lato black', 15,'bold'), text = "\t    Amount  :  ",
label1.grid(row=2, column=0,sticky=W)

label1 = tk.Label(root,font=('lato black', 15,'bold'), text = "\t    From Currency
label1.grid(row=3, column=0,sticky=W)

label1 = tk.Label(root,font=('lato black', 15,'bold'), text = "\t    To Currency
label1.grid(row=4, column=0,sticky=W)

label1 = tk.Label(root,font=('lato black', 15,'bold'), text = "\t    Converted Amo
label1.grid(row=8, column=0,sticky=W)


Label_1 =Label(root, font=('lato black', 7,'bold'), text="",padx=2,pady=2, bg="#e6
Label_1.grid(row=5, column=0,sticky=W)

Label_1 =Label(root, font=('lato black', 7,'bold'), text="",padx=2,pady=2, bg="#e6
```

```
Label_1.grid(row=7, column=0,sticky=W)



FromCurrency_option = tk.OptionMenu(root, variable1, *CurrenyCode_list)
ToCurrency_option = tk.OptionMenu(root, variable2, *CurrenyCode_list)

FromCurrency_option.grid(row = 3, column = 0, ipadx = 45,sticky=E)
ToCurrency_option.grid(row = 4, column = 0, ipadx = 45,sticky=E)




Amount1_field = tk.Entry(root)
Amount1_field.grid(row=2,column=0,ipadx =28,sticky=E)

Amount2_field = tk.Entry(root)
Amount2_field.grid(row=8,column=0,ipadx =31,sticky=E)


Label_9 =Button(root, font=('arial', 15,'bold'), text="   Convert  ",padx=2,pady=2
Label_9.grid(row=6, column=0)

Label_1 =Label(root, font=('lato black', 7,'bold'), text="",padx=2,pady=2, bg="#e6
Label_1.grid(row=9, column=0,sticky=W)

Label_9 =Button(root, font=('arial', 15,'bold'), text="   Clear All  ",padx=2,pady
Label_9.grid(row=10, column=0)


root.mainloop()
```

3.1 ROCK , PAPER & SCISSOR GAME

Create a program that allows the user to play the classic game of rock, paper, scissors against the computer.

In [ ]:

```
import random

print("     ~~~~~~ Welcome To Rock Paper Scissor ~~~~~    ")

User_Score = 0
Comp_Score = 0
Tie = 0
result=""

name = input("""Enter Your Name: """)

print("""
    * WINNING RULE *
1. Paper VS Rock -> Paper
2. Rock VS Scissor -> Rock
3. Scissor Vs Paper -> Scissor""")

while True:
    print()
    print("""   Choices Are:-
    1. Rock
    2. Paper
    3. Scissor"""
    )
```

```
choice = int(input("Enter Your Choice From 1-3: "))
print()


while choice > 3 or choice < 1:
    choice = int(input("Please Enter Valid Choice!"))

if choice == 1:
    user_choice = " Rock"
elif choice == 2:
    user_choice = "Paper"
elif choice == 3:
    user_choice = "Scissor"

print("User Selected:- ", user_choice)

print()
print("Now Computer Turn:- ")
computer = random.randint(1, 3)

if computer == 1:
    comp_choice = "Rock"
elif computer == 2:
    comp_choice = "Paper"
elif computer == 3:
    comp_choice = "Scissor"

print("Computer Selected:- ", comp_choice)

if ((user_choice == "Paper" and comp_choice == "Rock") or (user_choice == "Roc
    print()
    print("""       USER WIN!    """)
    result = "USER WIN!"

elif ((comp_choice == "Paper" and user_choice == "Rock") or (comp_choice == "R
    print()
    print("""       COMPUTER WIN!   """)
    result = "COMPUTER WIN!"

elif ((user_choice == "Paper" and comp_choice == "Paper") or (user_choice == "
    print()
    print("""       MATCH TIED!   """)
    result = "MATCH TIED!"

if result == "USER WIN!":
    User_Score +=1

elif result == "COMPUTER WIN!":
    Comp_Score +=1

elif result == "MATCH TIED!":
    Tie +=1

print()
print(name, "Score Is:- ", User_Score)
print("Computer Score Is:- ", Comp_Score)
print("Ties Matches Are:- ", Tie)

repeat = input("Do You Want To Play Again? yes/no ")
if repeat == "No" or repeat =="no":
    break
```

```
print("Game Over!")
print("Thanking You For Playing.")
```

3.2 ROCK , PAPER & SCISSOR GAME WITH GUI

In [ ]:

4.1 TIC TAC GAME

Create a simple Tic Tac Toe game that can be played between two players or against the computer.

In [ ]:
```
instructions = """
       This Will Be Our Tic Tac Board
               --------------
                 1 | 2 | 3
               --------------
                 4 | 5 | 6
               --------------
                 7 | 8 | 9
               --------------
 Key Points:-
      1. Insert the Spot Number(1-9) To Put Your Sign
      2. You Must Fill All 9 Spots To get the Results
      3. Player Fisrt Will Go First """

sign_dic = []
for i in range(9):
    sign_dic.append("")


def print_board():
    board = f"""
                --------------
                 {sign_dic[0]} | {sign_dic[1]} | {sign_dic[2]}
                --------------
                 {sign_dic[3]} | {sign_dic[4]} | {sign_dic[5]}
                --------------
                 {sign_dic[6]} | {sign_dic[7]} | {sign_dic[8]}
                --------------
               """
    print(board)


index_list = []


def take_input(player_name):
    while True:
        x = int(input(player_name))
        x -= 1
        if 0 <= x < 10:
            if x in index_list:
                print("This Spot Is Blocked ")
                continue
            index_list.append(x)
```

```
            return x
        print("Please Enter Number Between 1-9:- ")


def calculate_result(player_one, player_two):
    if (
        sign_dic[0] == sign_dic[1] == sign_dic[2] == "x"
        or sign_dic[1] == sign_dic[4] == sign_dic[7] == "x"
        or sign_dic[0] == sign_dic[4] == sign_dic[8] == "x"
        or sign_dic[2] == sign_dic[5] == sign_dic[8] == "x"
        or sign_dic[3] == sign_dic[4] == sign_dic[5] == "x"
        or sign_dic[2] == sign_dic[4] == sign_dic[6] == "x"
        or sign_dic[6] == sign_dic[7] == sign_dic[8] == "x"
        or sign_dic[0] == sign_dic[3] == sign_dic[6] == "x"
    ):
        print("Congratulation", player_one, "!!! . YOU WON.!")
        quit("thank you Both for Joining")

    elif (
        sign_dic[0] == sign_dic[1] == sign_dic[2] == "0"
        or sign_dic[1] == sign_dic[4] == sign_dic[7] == "0"
        or sign_dic[0] == sign_dic[4] == sign_dic[8] == "0"
        or sign_dic[2] == sign_dic[5] == sign_dic[8] == "0"
        or sign_dic[3] == sign_dic[4] == sign_dic[5] == "0"
        or sign_dic[2] == sign_dic[4] == sign_dic[6] == "0"
        or sign_dic[6] == sign_dic[7] == sign_dic[8] == "0"
        or sign_dic[0] == sign_dic[3] == sign_dic[6] == "0"
    ):
        print("Congratulation", player_two, "!!! . YOU WON.!")
        quit("thank you Both for Joining")


def main():
    print("         ***WELCOME TO THE TIC TAC TOE GAME***   ")
    print()
    player_one = input("Enter First Player Name:- ")
    player_two = input("Enter Second Player Name:- ")
    print()
    print("WELCOME TO THE TIC TAC TOE GAME", player_one, "AND", player_two)
    print()

    print(instructions)

    print()
    print(player_one, "`s Sign Will Be - X")
    print(player_two, "`s Sign Will Be - 0")
    print()
    input("Enter Any Key to Start the game:- ")

    print_board()

    for i in range(9):
        if i % 2 == 0:
            index = take_input(player_one)
            sign_dic[index] = "x"
        else:
            index = take_input(player_two)
            sign_dic[index] = "0"

        print_board()
        calculate_result(player_one, player_two)

    print("Match  TIE!")
```

```
main()
```

4.2 TIC TAC GAME WITH GUI

In [ ]:

```
from tkinter import *
import random

def next_turn(row, column):

    global player

    if buttons[row][column]['text'] == "" and check_winner() is False:

        if player == players[0]:

            buttons[row][column]['text'] = player

            if check_winner() is False:
                player = players[1]
                label.config(text=(players[1]+" turn"))

            elif check_winner() is True:
                label.config(text=(players[0]+" wins"))

            elif check_winner() == "Tie":
                label.config(text="Tie!")

        else:

            buttons[row][column]['text'] = player

            if check_winner() is False:
                player = players[0]
                label.config(text=(players[0]+" turn"))

            elif check_winner() is True:
                label.config(text=(players[1]+" wins"))

            elif check_winner() == "Tie":
                label.config(text="Tie!")

def check_winner():

    for row in range(3):
        if buttons[row][0]['text'] == buttons[row][1]['text'] == buttons[row][2]['
            buttons[row][0].config(bg="green")
            buttons[row][1].config(bg="green")
            buttons[row][2].config(bg="green")
            return True

    for column in range(3):
        if buttons[0][column]['text'] == buttons[1][column]['text'] == buttons[2][
            buttons[0][column].config(bg="green")
            buttons[1][column].config(bg="green")
            buttons[2][column].config(bg="green")
            return True

    if buttons[0][0]['text'] == buttons[1][1]['text'] == buttons[2][2]['text'] !=
        buttons[0][0].config(bg="green")
        buttons[1][1].config(bg="green")
```

```
        buttons[2][2].config(bg="green")
        return True

    elif buttons[0][2]['text'] == buttons[1][1]['text'] == buttons[2][0]['text'] !
        buttons[0][2].config(bg="green")
        buttons[1][1].config(bg="green")
        buttons[2][0].config(bg="green")
        return True

    elif empty_spaces() is False:

        for row in range(3):
            for column in range(3):
                buttons[row][column].config(bg="yellow")
        return "Tie"

    else:
        return False


def empty_spaces():

    spaces = 9

    for row in range(3):
        for column in range(3):
            if buttons[row][column]['text'] != "":
                spaces -= 1

    if spaces == 0:
        return False
    else:
        return True

def new_game():

    global player

    player = random.choice(players)

    label.config(text=player+" turn")

    for row in range(3):
        for column in range(3):
            buttons[row][column].config(text="",bg="#F0F0F0")


window = Tk()
window.title("Tic-Tac-Toe")
players = ["x","o"]
player = random.choice(players)
buttons = [[0,0,0],
           [0,0,0],
           [0,0,0]]

label = Label(text=player + " turn", font=('consolas',40))
label.pack(side="top")

reset_button = Button(text="restart", font=('consolas',20), command=new_game)
reset_button.pack(side="top")

frame = Frame(window)
frame.pack()
```

```
for row in range(3):
    for column in range(3):
        buttons[row][column] = Button(frame, text="",font=('consolas',40), width=5
                                      command= lambda row=row, column=column: next_
        buttons[row][column].grid(row=row,column=column)

window.mainloop()
```

5.1 TEMPERATURE CONVERTER

Create a program that allows the user to convert temperatures between Fahrenheit and Celsius

In [ ]:
```
unit = input("Is this temperature in Celsius or Fahrenheit (C/F): ")
temp = float(input("Enter the temperature: "))

if unit == "C":
    temp = round((9 * temp) / 5 + 32, 1)
    print(f"The temperature in Fahrenheit is: {temp}°F")
elif unit == "F":
    temp = round((temp - 32) * 5 / 9, 1)
    print(f"The temperature in Celsius is: {temp}°C")
else:
    print(f"{unit} is an invalid unit of measurement")
```

5.2 TEMPERATURE CONVERTER WITH GUI

In [ ]:
```
from tkinter import*
root=Tk()
var1=DoubleVar()
var2=DoubleVar()

label=Label(root,text='TEMPERATURE CONVERTER',font=('Arial',30))
label.pack(side=TOP)

label1=Label(root,text='Temperture in Celcius =',font=('Arial',25))
label1.place(x=200,y=200)

label2=Label(root,text='Temperture in Fahrenheit =',font=('Arial',25))
label2.place(x=200,y=300)

entry1=Entry(root,font=('Arial',25),textvariable=var1)
entry1.place(x=600,y=200)

label3=Label(root,font=('Arial',25))
label3.place(x=600,y=300)

def click1():
        label3.config(text='' +str(var1.get() * 1.8 +    32)+ ' °F')

button1=Button(root,text='Convert',font=('Arial',25),command=click1)
button1.place(x=920,y=200)

label4=Label(root,text='Temperture in Fahrenheit =',font=('Arial',25))
label4.place(x=200,y=400)
label5=Label(root,text='Temperture in Celcius =',font=('Arial',25))
label5.place(x=200,y=500)
```

```
entry2=Entry(root,font=('Arial',25),textvariable=var2)
entry2.place(x=600,y=400)
label6=Label(root,font=('Arial',25))
label6.place(x=600,y=500)

def click2():
        label6.config(text='' +str((var2.get()-32)/1.8)+ ' °C')
button2=Button(root,text='Convert',font=('Arial',25),command=click2)
button2.place(x=920,y=400)
root.mainloop()
```

6.1 ATM SIMULATOR

Create a program that simulates the all atm functionalities and operations (Check balance, Deposit, Withdraw)

In [ ]:

```
import time

print("Please Wait Some Time Your CARD Is Inserting. ")

time.sleep(5)

password = 12345

pin = int(input("Enter Your ATM PIN:-  "))

balance = 50000

if pin == password:
    while True:
        print(
            """
            1. BALANACE CHEACK
            2. WITHDRAW AMOUNT
            3. DEPOSIT AMOUNT
            4. EXIT
                        """
        )

        try:
            option = int(input("Please Enter Your Option:- "))
        except:
            print("Please Enter Valid Option! ")
            print()

        if option == 1:
            print(f"Your Current AMOUNT is {balance}")
            print()

        elif option == 2:
            withdraw_amount = int(input("Please Enter Withdraw Amount:- "))

            if withdraw_amount <= balance:
                balance = balance - withdraw_amount
                print(f"{withdraw_amount} Is Debited From Your Account. ")
                print()
                print(f"Your Current Balance Is {balance}")
                print()

            else:
                print()
```

```
                print("Please Enter Valid Amount. ")

        elif option == 3:
            deposit_amount = int(input("Please Enter Deposit Amount:- "))

            balance = balance + deposit_amount

            print(f"{deposit_amount} Is Credited To Your Account. ")
            print()

            print(f"Your Current Balance Is {balance}")
            print()

        elif option == 4:
            break


else:
    print("Please! Enter Valid Pin!! ")
```

6.2 ATM SIMULATOR WITH GUI

In [ ]:

```
from tkinter import *
import tkinter.font as font
import random


win = Tk()
win.title('ATM')
win.geometry('460x390')

tim40 = font.Font(family='Times', size=40, weight='bold', slant='italic', underlin



cour20 = font.Font(family='Courier', size=20, weight='bold')
cour15 = font.Font(family='Courier', size=15, weight='bold')

glob_count = 0

def display_func():

    question_func.question_win.withdraw()
    display_win = Toplevel(win)
    display_win.geometry('460x390')
    message = Message(display_win, text='\n\nYour transaction has been successful\
    message.pack()
    text = Label(display_win, text='ATM', font=tim40, fg='red')
    text.pack()

    exit_button = Button(display_win, text='EXIT', font=cour15, fg='red', command=
    exit_button.pack(side=BOTTOM, pady=10)


# window asking whether to show balance or not
def question_func():

    global glob_count
    glob_count+=1

    withdrawal_func.withdrawal_win.withdraw()
```

```
    question_func.question_win = Toplevel(win)
    question_func.question_win.geometry('460x390')

    bf = Frame(question_func.question_win)
    bf.pack(side=BOTTOM)

    msg_box = Message(question_func.question_win, text='\nYour transaction has beer
    msg_box.pack()

    yes_btn = Button(bf, text='YES', font=cour15, fg='green', command=balance_func
    yes_btn.pack(side=LEFT, pady=10)

    no_btn = Button(bf, text=' NO ', font=cour15, fg='red', command=display_func)
    no_btn.pack(pady=10, padx=10)


# withdrawing money window
def withdrawal_func():

    option_func.option_win.withdraw()
    withdrawal_func.withdrawal_win = Toplevel(win)
    withdrawal_func.withdrawal_win.geometry('460x390')

    enter_lbl = Label(withdrawal_func.withdrawal_win, text='\nPlease enter amount\r
    enter_lbl.pack()
    money_entry = Entry(withdrawal_func.withdrawal_win, font=cour15, justify='cente
    money_entry.pack()

    bf = Frame(withdrawal_func.withdrawal_win)
    bf.pack(side=BOTTOM)

    bf4 = Frame(withdrawal_func.withdrawal_win)
    bf4.pack(side=BOTTOM)

    bf3 = Frame(withdrawal_func.withdrawal_win)
    bf3.pack(side=BOTTOM)

    bf3 = Frame(withdrawal_func.withdrawal_win)
    bf3.pack(side=BOTTOM)

    bf2 = Frame(withdrawal_func.withdrawal_win)
    bf2.pack(side=BOTTOM)

    bf1 = Frame(withdrawal_func.withdrawal_win)
    bf1.pack(side=BOTTOM)

    b1 = Button(bf1, text='1', font=cour15, command=lambda: money_entry.insert('end
    b1.pack(side=LEFT, pady=10)

    b2 = Button(bf1, text='2', font=cour15, command=lambda: money_entry.insert('end
    b2.pack(side=LEFT, padx=10)

    b3 = Button(bf1, text='3', font=cour15, command=lambda: money_entry.insert('end
    b3.pack(side=LEFT)

    b4 = Button(bf2, text='4', font=cour15, command=lambda: money_entry.insert('end
    b4.pack(side=LEFT)

    b5 = Button(bf2, text='5', font=cour15, command=lambda: money_entry.insert('end
    b5.pack(side=LEFT, padx=10)

    b6 = Button(bf2, text='6', font=cour15, command=lambda: money_entry.insert('end
    b6.pack(side=LEFT)
```

```
    b7 = Button(bf3, text='7', font=cour15, command=lambda: money_entry.insert('en
    b7.pack(side=LEFT, pady=10)

    b8 = Button(bf3, text='8', font=cour15, command=lambda: money_entry.insert('en
    b8.pack(side=LEFT, padx=10)

    b9 = Button(bf3, text='9', font=cour15, command=lambda: money_entry.insert('en
    b9.pack(side=LEFT)

    btn = Button(bf4, text=' ', font=cour15)
    btn.pack(side=LEFT)

    b0 = Button(bf4, text='0', font=cour15, command=lambda: money_entry.insert('en
    b0.pack(side=LEFT, padx=10)

    btn_ = Button(bf4, text=' ', font=cour15)
    btn_.pack(side=LEFT)

    enter_btn = Button(bf, text='ENTER', font=cour15, fg='green', command=question_
    enter_btn.pack(side=LEFT, pady=10)

    clear_btn = Button(bf, text='CLEAR', font=cour15, fg='orange', command=lambda:
    clear_btn.pack(side=LEFT, padx=10)


def balance_func():

    global glob_count

    if glob_count == 1:
        question_func.question_win.withdraw()

    option_func.option_win.withdraw()
    balance_win = Toplevel(win)
    balance_win.geometry('460x390')

    balance = random.randrange(1000,1000000)
    message = Message(balance_win,text='\nYour transaction is successful\n\nAvailal
    message.pack()
    text = Label(balance_win, text='ATM', font=tim40, fg='red')
    text.pack()

    exit_button = Button(balance_win, text='EXIT', font=cour15, fg='red', command=
    exit_button.pack(side=BOTTOM, pady=10)


def message_func():
    change_pin_func.change_pin_win.withdraw()
    win2 = Toplevel(win)
    win2.geometry('460x390')
    message = Message(win2, text='\nYour transaction is successful\n\nYour PIN has
    message.pack()
    text = Label(win2, text='ATM', font=tim40, fg='red')
    text.pack()

    exit_button = Button(win2, text='EXIT', font=cour15, fg='red', command=lambda:
    exit_button.pack(side=BOTTOM, pady=10)


def change_pin_func():
    option_func.option_win.withdraw()
    change_pin_func.change_pin_win = Toplevel(win)
    change_pin_func.change_pin_win.geometry('460x420')
```

```
    pin_lbl = Label(change_pin_func.change_pin_win, text='\nEnter new-PIN', font=c
    pin_lbl.pack()
    pin_entry = Entry(change_pin_func.change_pin_win, font=cour15,justify='center'
    pin_entry.pack()

    re_entry_lbl = Label(change_pin_func.change_pin_win, text='\nRe-enter new-PIN'
    re_entry_lbl.pack()
    re_entry = Entry(change_pin_func.change_pin_win, font=cour15, justify='center'
    re_entry.pack()

    bf = Frame(change_pin_func.change_pin_win)
    bf.pack(side=BOTTOM)

    bf4 = Frame(change_pin_func.change_pin_win)
    bf4.pack(side=BOTTOM)

    bf3 = Frame(change_pin_func.change_pin_win)
    bf3.pack(side=BOTTOM)

    bf3 = Frame(change_pin_func.change_pin_win)
    bf3.pack(side=BOTTOM)

    bf2 = Frame(change_pin_func.change_pin_win)
    bf2.pack(side=BOTTOM)

    bf1 = Frame(change_pin_func.change_pin_win)
    bf1.pack(side=BOTTOM)

    b1 = Button(bf1, text='1', font=cour15, command=lambda: [pin_entry.insert('end
    b1.pack(side=LEFT,pady=10)

    b2 = Button(bf1, text='2', font=cour15, command=lambda: [pin_entry.insert('end
    b2.pack(side=LEFT, padx=10)

    b3 = Button(bf1, text='3', font=cour15, command=lambda: [pin_entry.insert('end
    b3.pack(side=LEFT)

    b4 = Button(bf2, text='4', font=cour15, command=lambda: [pin_entry.insert('end
    b4.pack(side=LEFT)

    b5 = Button(bf2, text='5', font=cour15, command=lambda: [pin_entry.insert('end
    b5.pack(side=LEFT, padx=10)

    b6 = Button(bf2, text='6', font=cour15, command=lambda: [pin_entry.insert('end
    b6.pack(side=LEFT)

    b7 = Button(bf3, text='7', font=cour15, command=lambda: [pin_entry.insert('end
    b7.pack(side=LEFT,pady=10)

    b8 = Button(bf3, text='8', font=cour15, command=lambda: [pin_entry.insert('end
    b8.pack(side=LEFT, padx=10)

    b9 = Button(bf3, text='9', font=cour15, command=lambda: [pin_entry.insert('end
    b9.pack(side=LEFT)

    btn = Button(bf4, text=' ', font=cour15)
    btn.pack(side=LEFT)

    b0 = Button(bf4, text='0', font=cour15, command=lambda: [pin_entry.insert('end
    b0.pack(side=LEFT, padx=10)                          # with help of list we can

    btn_ = Button(bf4, text=' ', font=cour15)
    btn_.pack(side=LEFT)
```

```
    enter_btn = Button(bf, text='ENTER', font=cour15, fg='green', command=message_
    enter_btn.pack(side=LEFT, pady=10)

    clear_btn = Button(bf, text='CLEAR', font=cour15, fg='orange', command=lambda:
    clear_btn.pack(side=LEFT, padx=10)


def option_func():
    enter_pin.new_win.withdraw()
    option_func.option_win = Toplevel(win)
    option_func.option_win.geometry('460x390')


    text_title = Label(option_func.option_win, text='\nATM', font=tim40)
    text_title.pack()

    rf = Frame(option_func.option_win)
    rf.pack(side=RIGHT)

    lf = Frame(option_func.option_win)
    lf.pack(side=LEFT)

    withdrawal_btn = Button(rf, text=' WITHDRAWAL ', font=cour15, fg='blue', comma
    withdrawal_btn.pack(padx=40, pady=10)

    balance_btn = Button(rf, text='BALANCE INQ', font=cour15, command=balance_func
    balance_btn.pack(padx=40, pady=10)

    change_pin_btn = Button(lf, text='CHANGE PIN', font=cour15, command=change_pin
    change_pin_btn.pack(padx=40, pady=10)

    exit_btn = Button(lf, text='   EXIT   ', font=cour15, fg='red', command=lambda
    exit_btn.pack(padx=40, pady=10)


def enter_pin():

    win.withdraw()

    enter_pin.new_win = Toplevel(win)
    enter_pin.new_win.geometry('460x390')



    def setInputText(text):
        entry_box.insert('end',text)
    def text_delete():
        entry_box.delete(0)
    lbl = Label(enter_pin.new_win, text='Enter your PIN',font=cour20,fg='red')
    lbl.pack(pady=20)

    entry_box = Entry(enter_pin.new_win, font=cour15, show='*', justify='center')
    entry_box.pack()

    bf = Frame(enter_pin.new_win)
    bf.pack(side=BOTTOM)

    bf0 = Frame(enter_pin.new_win)
    bf0.pack(side=BOTTOM)

    bf1 = Frame(enter_pin.new_win)
    bf1.pack(side=BOTTOM)

    bf2 = Frame(enter_pin.new_win)
```

```
    bf2.pack(side=BOTTOM)

    bf3 = Frame(enter_pin.new_win)
    bf3.pack(side=BOTTOM)

    bf4 = Frame(enter_pin.new_win)
    bf4.pack(side=BOTTOM)

    rf = Frame(enter_pin.new_win)
    rf.pack(side=RIGHT)

    btn1 = Button(bf4,text='1',font=cour15, command=lambda:setInputText('1'))
    btn1.pack(side=LEFT, pady=10)

    btn2 = Button(bf4, text='2', font=cour15, command=lambda:setInputText('2'))
    btn2.pack(side=LEFT,padx=10)

    btn3 = Button(bf4, text='3', font=cour15, command=lambda:setInputText('3'))
    btn3.pack(side=LEFT)

    btn4 = Button(bf3, text='4', font=cour15, command=lambda:setInputText('4'))
    btn4.pack(side=LEFT)

    btn5 = Button(bf3, text='5', font=cour15, command=lambda:setInputText('5'))
    btn5.pack(side=LEFT,padx=10)

    btn6 = Button(bf3, text='6', font=cour15, command=lambda:setInputText('6'))
    btn6.pack(side=LEFT)

    btn7 = Button(bf2, text='7', font=cour15, command=lambda:setInputText('7'))
    btn7.pack(side=LEFT,pady=10)

    btn8 = Button(bf2, text='8', font=cour15, command=lambda:setInputText('8'))
    btn8.pack(side=LEFT, padx=10)

    btn9 = Button(bf2, text='9', font=cour15, command=lambda:setInputText('9'))
    btn9.pack(side=LEFT)

    btn = Button(bf1, text=' ', font=cour15)
    btn.pack(side=LEFT)

    btn0 = Button(bf1, text='0', font=cour15, command=lambda:setInputText('0'))
    btn0.pack(side=LEFT, padx=10)

    btn_ = Button(bf1, text=' ', font=cour15)
    btn_.pack(side=LEFT)

    enter_btn = Button(bf0, text='ENTER', font=cour15,fg='green', command=option_f
    enter_btn.pack(side= LEFT, pady=10,padx=10)

    exit_btn = Button(bf0, text='EXIT', font=cour15, fg='red', command=lambda:[ent
    exit_btn.pack(side=RIGHT, padx=10)

    clear_btn = Button(bf0,text='CLEAR', font=cour15, fg='orange', command=text_de
    clear_btn.pack(side=LEFT)

    note = Label(bf, text='Note:Enter pin either from keyboard or keypad', fg='red
    note.pack()



title_label = Label(win, text='ATM', font=tim40, fg='red')
title_label.pack(pady=10)
user_id = random.randrange(1000,10000)
```

```
intro = Label(win, text='\nWelcome User '+str(user_id), font=cour20, fg='green')
intro.pack()
option_label = Label(win, text='\nSelect your account type', font=cour15, fg='grey
option_label.pack()

bottom_frame = Frame(win)
bottom_frame.pack(side=BOTTOM)
right_frame = Frame(win)
right_frame.pack(side=RIGHT)

note = Label(bottom_frame, text='NOTE:Use only EXIT button to exit', font=cour15,
note.pack(pady=10)
saving = Button(right_frame, text='Savings', font=cour15, bg='sky blue', fg='red',
saving.pack(padx=30, pady=10)
current = Button(right_frame, text="Current", font=cour15, bg='sky blue', fg='red'
current.pack(padx=30, pady=10)

win.mainloop()
```

7.1 TO DO LIST

Create a program that allows the user to create and manage a to-do list.

In [ ]:

7.2 TO DO LIST WITH GUI

In [ ]:
```
import tkinter
import threading
from tkinter import messagebox
import sys

tasks = []
timer = threading
real_timer = threading
ok_thread = True


def get_entry(event=""):
    text = todo.get()
    hour = int(time.get())
    todo.delete(0, tkinter.END)
    time.delete(0, tkinter.END)
    todo.focus_set()
    add_list(text, hour)
    if 0 < hour < 999:
        update_list()


def add_list(text, hour):
    tasks.append([text, hour])
    timer = threading.Timer(hour, time_passed, [text])
    timer.start()


def update_list():
    if todolist.size() > 0:
        todolist.delete(0, "end")
```

```
    for task in tasks:
        todolist.insert("end", "[" + task[0] + "] Time left: " + str(task[1]) + "

def time_passed(task):
    tkinter.messagebox.showinfo("Notification", "Time for : " + task)


def real_time():
    if ok_thread:
        real_timer = threading.Timer(1.0, real_time)
        real_timer.start()
    for task in tasks:
        if task[1] == 0:
            tasks.remove(task)
        task[1] -= 1
    update_list()


if __name__ == '__main__':

    app = tkinter.Tk()
    app.geometry("480x680")
    app.title("Todolist Remainder")
    app.rowconfigure(0, weight=1)


    frame = tkinter.Frame(app)
    frame.pack()

    label = tkinter.Label(app, text="Enter work to do:",
                          wraplength = 200,
                          justify = tkinter.LEFT)
    label_hour = tkinter.Label(app, text="Enter time (secondes)",
                               wraplength = 200,
                               justify = tkinter.LEFT)
    todo = tkinter.Entry(app, width=30)
    time = tkinter.Entry(app, width=15)
    send = tkinter.Button(app, text='Add task', fg="#ffffff", bg='#6186AC', height
    quit = tkinter.Button(app, text='Exit', fg="#ffffff", bg='#EB6464', height=3, 
    todolist = tkinter.Listbox(app)
    if tasks != "":
        real_time()


    app.bind('<Return>', get_entry)


    label.place(x=0, y=10, width=200, height=25)
    label_hour.place(x=235, y=10, width=200, height=25)
    todo.place(x=62, y=30, width=200, height=25)
    time.place(x=275, y=30, width=50, height=25)
    send.place(x=62, y=60, width=50, height=25)
    quit.place(x=302, y=60, width=50, height=25)
    todolist.place(x=60, y = 100, width=300, height=300)

    app.mainloop()
    ok_thread = False
    sys.exit("FINISHED")
```

*7.2 TO DO LIST WITH GUI (ANOTHER)

In [ ]:

```
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import sqlite3 as sql


def add_task():
    task_string = task_field.get()

    if len(task_string) == 0:
        messagebox.showinfo("Error", "Field is Empty.")
    else:
        tasks.append(task_string)

        the_cursor.execute("insert into tasks values (?)", (task_string,))

        list_update()

        task_field.delete(0, "end")


def list_update():
    clear_list()

    for task in tasks:
        task_listbox.insert("end", task)


def delete_task():
    try:
        the_value = task_listbox.get(task_listbox.curselection())

        if the_value in tasks:
            tasks.remove(the_value)

            list_update()

            the_cursor.execute("delete from tasks where title = ?", (the_value,))
    except:
        messagebox.showinfo("Error", "No Task Selected. Cannot Delete.")


def delete_all_tasks():
    message_box = messagebox.askyesno("Delete All", "Are you sure?")

    if message_box == True:
        while len(tasks) != 0:
            tasks.pop()

        the_cursor.execute("delete from tasks")

        list_update()


def clear_list():
    task_listbox.delete(0, "end")


def close():
    print(tasks)

    guiWindow.destroy()
```

```
def retrieve_database():
    while len(tasks) != 0:
        tasks.pop()

    for row in the_cursor.execute("select title from tasks"):
        tasks.append(row[0])


if __name__ == "__main__":
    guiWindow = tk.Tk()

    guiWindow.title("To-Do List Manager - JAVATPOINT")

    guiWindow.geometry("500x450+750+250")

    guiWindow.resizable(0, 0)

    guiWindow.configure(bg="#FAEBD7")

    the_connection = sql.connect("listOfTasks.db")

    the_cursor = the_connection.cursor()

    the_cursor.execute("create table if not exists tasks (title text)")

    tasks = []

    header_frame = tk.Frame(guiWindow, bg="#FAEBD7")
    functions_frame = tk.Frame(guiWindow, bg="#FAEBD7")
    listbox_frame = tk.Frame(guiWindow, bg="#FAEBD7")

    header_frame.pack(fill="both")
    functions_frame.pack(side="left", expand=True, fill="both")
    listbox_frame.pack(side="right", expand=True, fill="both")

    header_label = ttk.Label(
        header_frame,
        text="The To-Do List",
        font=("Brush Script MT", "30"),
        background="#FAEBD7",
        foreground="#8B4513",
    )

    header_label.pack(padx=20, pady=20)

    task_label = ttk.Label(
        functions_frame,
        text="Enter the Task:",
        font=("Consolas", "11", "bold"),
        background="#FAEBD7",
        foreground="#000000",
    )

    task_label.place(x=30, y=40)

    task_field = ttk.Entry(
        functions_frame,
        font=("Consolas", "12"),
        width=18,
        background="#FFF8DC",
        foreground="#A52A2A",
    )
```

```
    task_field.place(x=30, y=80)

    add_button = ttk.Button(
        functions_frame, text="Add Task", width=24, command=add_task
    )
    del_button = ttk.Button(
        functions_frame, text="Delete Task", width=24, command=delete_task
    )
    del_all_button = ttk.Button(
        functions_frame, text="Delete All Tasks", width=24, command=delete_all_tasl
    )
    exit_button = ttk.Button(functions_frame, text="Exit", width=24, command=close

    add_button.place(x=30, y=120)
    del_button.place(x=30, y=160)
    del_all_button.place(x=30, y=200)
    exit_button.place(x=30, y=240)

    task_listbox = tk.Listbox(
        listbox_frame,
        width=26,
        height=13,
        selectmode="SINGLE",
        background="#FFFFFF",
        foreground="#000000",
        selectbackground="#CD853F",
        selectforeground="#FFFFFF",
    )

    task_listbox.place(x=10, y=20)

    retrieve_database()
    list_update()

    guiWindow.mainloop()

    the_connection.commit()
    the_cursor.close()
```

8.1 MUSIC PLAYER

Create a simple music player that can play MP3 files and allows the user to create and manage playlists.

In [ ]:

8.2 MUSIC PLAYER WITH GUI

In [ ]:
```
!pip install pygame

import os
import pickle
import tkinter as tk
from tkinter import filedialog
from tkinter import PhotoImage
from pygame import mixer
```

```
class Player(tk.Frame):
    def __init__(self, master=None):
        super().__init__(master)
        self.master = master
        self.pack()
        mixer.init()

        if os.path.exists("songs.pickle"):
            with open("songs.pickle", "rb") as f:
                self.playlist = pickle.load(f)
        else:
            self.playlist = []

        self.current = 0
        self.paused = True
        self.played = False

        self.create_frames()
        self.track_widgets()
        self.control_widgets()
        self.tracklist_widgets()

    def create_frames(self):
        self.track = tk.LabelFrame(
            self,
            text="Song Track",
            font=("times new roman", 15, "bold"),
            bg="grey",
            fg="white",
            bd=5,
            relief=tk.GROOVE,
        )
        self.track.config(width=410, height=300)
        self.track.grid(row=0, column=0, padx=10)

        self.tracklist = tk.LabelFrame(
            self,
            text=f"PlayList - {str(len(self.playlist))}",
            font=("times new roman", 15, "bold"),
            bg="grey",
            fg="white",
            bd=5,
            relief=tk.GROOVE,
        )
        self.tracklist.config(width=190, height=400)
        self.tracklist.grid(row=0, column=1, rowspan=3, pady=5)

        self.controls = tk.LabelFrame(
            self,
            font=("times new roman", 15, "bold"),
            bg="white",
            fg="white",
            bd=2,
            relief=tk.GROOVE,
        )
        self.controls.config(width=410, height=80)
        self.controls.grid(row=2, column=0, pady=5, padx=10)

    def track_widgets(self):
        self.canvas = tk.Label(self.track, image=img)
        self.canvas.configure(width=400, height=240)
        self.canvas.grid(row=0, column=0)

        self.songtrack = tk.Label(
```

```
            self.track, font=("times new roman", 16, "bold"), bg="white", fg="dark
        )
        self.songtrack["text"] = "Musicxy MP3 Player"
        self.songtrack.config(width=30, height=1)
        self.songtrack.grid(row=1, column=0, padx=10)

    def control_widgets(self):
        self.loadSongs = tk.Button(self.controls, bg="green", fg="white", font=10)
        self.loadSongs["text"] = "Load Songs"
        self.loadSongs["command"] = self.retrieve_songs
        self.loadSongs.grid(row=0, column=0, padx=10)

        self.prev = tk.Button(self.controls, image=prev)
        self.prev["command"] = self.prev_song
        self.prev.grid(row=0, column=1)

        self.pause = tk.Button(self.controls, image=pause)
        self.pause["command"] = self.pause_song
        self.pause.grid(row=0, column=2)

        self.next = tk.Button(self.controls, image=next_)
        self.next["command"] = self.next_song
        self.next.grid(row=0, column=3)

        self.volume = tk.DoubleVar(self)
        self.slider = tk.Scale(self.controls, from_=0, to=10, orient=tk.HORIZONTAL
        self.slider["variable"] = self.volume
        self.slider.set(8)
        mixer.music.set_volume(0.8)
        self.slider["command"] = self.change_volume
        self.slider.grid(row=0, column=4, padx=5)

    def tracklist_widgets(self):
        self.scrollbar = tk.Scrollbar(self.tracklist, orient=tk.VERTICAL)
        self.scrollbar.grid(row=0, column=1, rowspan=5, sticky="ns")

        self.list = tk.Listbox(
            self.tracklist,
            selectmode=tk.SINGLE,
            yscrollcommand=self.scrollbar.set,
            selectbackground="sky blue",
        )
        self.enumerate_songs()
        self.list.config(height=22)
        self.list.bind("<Double-1>", self.play_song)

        self.scrollbar.config(command=self.list.yview)
        self.list.grid(row=0, column=0, rowspan=5)

    def retrieve_songs(self):
        self.songlist = []
        directory = filedialog.askdirectory()
        for root_, dirs, files in os.walk(directory):
            for file in files:
                if os.path.splitext(file)[1] == ".mp3":
                    path = (root_ + "/" + file).replace("\\", "/")
                    self.songlist.append(path)

        with open("songs.pickle", "wb") as f:
            pickle.dump(self.songlist, f)
        self.playlist = self.songlist
        self.tracklist["text"] = f"PlayList - {str(len(self.playlist))}"
        self.list.delete(0, tk.END)
        self.enumerate_songs()
```

```
    def enumerate_songs(self):
        for index, song in enumerate(self.playlist):
            self.list.insert(index, os.path.basename(song))

    def play_song(self, event=None):
        if event is not None:
            self.current = self.list.curselection()[0]
            for i in range(len(self.playlist)):
                self.list.itemconfigure(i, bg="white")

        print(self.playlist[self.current])
        mixer.music.load(self.playlist[self.current])
        self.songtrack["anchor"] = "w"
        self.songtrack["text"] = os.path.basename(self.playlist[self.current])

        self.pause["image"] = play
        self.paused = False
        self.played = True
        self.list.activate(self.current)
        self.list.itemconfigure(self.current, bg="sky blue")

        mixer.music.play()

    def pause_song(self):
        if not self.paused:
            self.paused = True
            mixer.music.pause()
            self.pause["image"] = pause
        else:
            if self.played == False:
                self.play_song()
            self.paused = False
            mixer.music.unpause()
            self.pause["image"] = play

    def prev_song(self):
        if self.current > 0:
            self.current -= 1
        else:
            self.current = 0
        self.list.itemconfigure(self.current + 1, bg="white")
        self.play_song()

    def next_song(self):
        if self.current < len(self.playlist) - 1:
            self.current += 1
        else:
            self.current = 0
        self.list.itemconfigure(self.current - 1, bg="white")
        self.play_song()

    def change_volume(self, event=None):
        self.v = self.volume.get()
        mixer.music.set_volume(self.v / 10)

root = tk.Tk()
root.geometry("600x400")
root.wm_title("Musicxy")

img = PhotoImage(file="images/music.gif")
next_ = PhotoImage(file="images/next.gif")
prev = PhotoImage(file="images/previous.gif")
play = PhotoImage(file="images/play.gif")
```

```
pause = PhotoImage(file="images/pause.gif")

app = Player(master=root)
app.mainloop()
```

9.1 RANDOM PASSWORD GENERATOR

Create a program that generates a random password of a user-defined length.

In [ ]:
```
import random
import string
print("    ***WELCOME TO PASSWORD GENERATOR***")

def main():
    length = int(input("Enter The Length Of Password:- "))

    lowerD = string.ascii_lowercase
    upperD = string.ascii_uppercase
    digitD = string.digits
    symbolD = string.punctuation

    combine = lowerD + upperD + digitD + symbolD

    x =random.sample(combine,length)
    password = "".join(x)
    print(password)
main()
```

9.2 RANDOM PASSWORD GENERATOR WITH GUI

In [ ]:
```
from tkinter import *
import string
import random

def generator():
    small_alphabets=string.ascii_lowercase
    capital_alphabets=string.ascii_uppercase
    numbers=string.digits
    special_charecters=string.punctuation

    all=small_alphabets+capital_alphabets+numbers+special_charecters
    password_length=int(length_Box.get())

    if choice.get()==1:
        passwordField.insert(0,random.sample(small_alphabets,password_length))

    if choice.get()==2:
        passwordField.insert(0,random.sample(small_alphabets+capital_alphabets,pas

    if choice.get()==3:
        passwordField.insert(0,random.sample(all,password_length))


def copy():
    random_password=passwordField.get()
    pyperclip.copy(random_password)
```

```
root=Tk()
root.config(bg='gray20')
choice=IntVar()
Font=('arial',13,'bold')
passwordLabel=Label(root,text='Password Generator',font=('times new roman',20,'bol
passwordLabel.grid(pady=10)
weakradioButton=Radiobutton(root,text='Weak',value=1,variable=choice,font=Font)
weakradioButton.grid(pady=5)

mediumradioButton=Radiobutton(root,text='Medium',value=2,variable=choice,font=Font
mediumradioButton.grid(pady=5)

strongradioButton=Radiobutton(root,text='Strong',value=3,variable=choice,font=Font
strongradioButton.grid(pady=5)

lengthLabel=Label(root,text='Password Length',font=Font,bg='gray20',fg='white')
lengthLabel.grid(pady=5)

length_Box=Spinbox(root,from_=5,to_=18,width=5,font=Font)
length_Box.grid(pady=5)

generateButton=Button(root,text='Generate',font=Font,command=generator)
generateButton.grid(pady=5)

passwordField=Entry(root,width=25,bd=2,font=Font)
passwordField.grid()

copyButton=Button(root,text='Copy',font=Font,command=copy)
copyButton.grid(pady=5)

root.mainloop()
```

In [ ]: