Factorial and Factor Calculator

```
"""I have use the library pygame found here:
https://www.pygame.org/wiki/about

 in my code, this contains methods and classes that are used to detect user input and display shapes on the screen."

import pygame, sys
from pygame.locals import *

#Initialize the module
pygame.init()

"""Variable Initialization"""

#Width and height of the screen
width = 600
height = 600

appExit = False

#Define colors within a dict object
colors = {"red":(255,0,0),
            "green":(0,255,0),
            "blue":(0,0,255),
            "white":(255,255,255),
            "black":(0,0,0)}

#Initialize an instance of the screen defined by width and height

screen = pygame.display.set_mode((width, height))

pygame.display.set_caption("Factorial and Factor calculator")

#Get the system font
font = pygame.font.SysFont(None, 25)


"""Creating a class and functions to use in the main loop"""
#Create a button class that stores information about the button
class Button:
    def __init__(self, screen, text, textColor, coordX, coordY, sizeX, sizeY, borderColor, innerColor, borderThickness = 10, fill = False):
        self.screen = screen
        self.text = text
        self.textColor = textColor
        self.coordX = coordX
        self.coordY = coordY
        self.sizeX = sizeX
        self.sizeY = sizeY
        self.borderColor = borderColor
        self.innerColor = innerColor
        self.borderThickness = borderThickness
        self.fill = fill


    #Once the instance of the button is in the button list, it will be drawing with this method using the pygame update method
    def DrawButton(self):
        #Optional fill mode that draws a rect to the screen using the variables defined in the constructor
        if self.fill == False:
            pygame.draw.rect(self.screen, self.borderColor, [self.coordX, self.coordY, self.sizeX, self.sizeY])
            pygame.draw.rect(self.screen, self.innerColor, [self.coordX + self.borderThickness/2, self.coordY + self.borderThickness/2, self.sizeX - self.borderThickness

            DrawText(self.text, self.textColor, self.coordX + self.sizeX/2, self.coordY + self.sizeY/2)
        elif self.fill:
            pygame.draw.rect(self.screen, self.borderColor, [self.coordX, self.coordY, self.sizeX, self.sizeY])

            DrawText(self.text, self.textColor, self.coordX + self.sizeX/2, self.coordY + self.sizeY/2)

#Use pygame module to "blit" text to a certain coordinate on screen
def DrawText(msg, color, msgX, msgY):
    text = font.render(msg, True, color)
    text_rect = text.get_rect(center=(msgX, msgY))
    screen.blit(text, text_rect)


#Use the button class and parameters defined below to create all of the button instances needed in the app
def CreateButtonInstance(nameList, positionDict, numberOfButtons, btnSizeX, btnSizeY):
    btnList_ = []

    #Iterate over the position dict to assign each button class a position
    for i in range(numberOfButtons):
        btnList_.append(Button(screen, "", (0,0,0), buttonPositions[i][0], buttonPositions[i][1], btnSizeX, btnSizeY, colors["black"], colors["white"]))

    #Use the button colors and names list to assign each button in the list a text color and text value
    for btn in btnList_:
        btn.text = nameList[btnList_.index(btn)][0]
        btn.textColor = colors[nameList[btnList_.index(btn)][1]]

    return btnList_

#Check if the mouse is clicked within the bounds of a certain button and return that buttons text value using the pygame module
def CheckButtonPress(mousePos, btnList_):

    for btn in btnList_:
        if mousePos[0] > btn.coordX and mousePos[0] < btn.coordX + btn.sizeX and mousePos[1] > btn.coordY and mousePos[1] < btn.coordY + btn.sizeY:
            return btn.text
```

```
            break

#Find the factorial value of the current calculator input
def Factorial(calculatorInput_):

    total = 1

    for i in range(int(calculatorInput_)):
        total = (i + 1) * total

    print(total)
    return total

#Determine whether the number is prime, or what its factors below 10 are
def DeterminePrime(calculatorInput_):

    intInput = int(calculatorInput_)

    factors = []

    for i in range(2,10):
        if intInput % i == 0:
            if i != intInput:
                factors.append(i)

    return factors



"""Defining Button Names, Positions, and Color"""
#The buttons are given coordinates based on what "i" equals within this dict. "i" translates to their order on the screen
buttonPositions = {0:(0, 200), 1:(150,200), 2:(300,200), 3:(450,200),
                   4:(0, 333), 5:(150,333), 6:(300,333), 7:(450,333),
                   8:(0,466), 9:(150,466), 10:(300,466), 11:(450,466), 12:(0,0)}

#The text and color of each button is assigned to the buttons in order from left to right
buttonNames = [("0", "black"), ("1", "black"), ("2", "black"), ("3", "black"), ("Prime", "red"), ("4", "black"), ("5", "black"), ("6", "black"), ("!", "red"), ("7", "blc

#Create button instances before update method
btnList = CreateButtonInstance(buttonNames, buttonPositions, 13, 150, 133)



#Make the clear button a bit smaller
for btn in btnList:
    if btn.text == "Clear":
        btn.sizeX = 100
        btn.sizeY = 50
        btn.borderThickness = 2

#Define the calculator input string which will always be displayed in real time
calculatorInput = ""

"""Update Loop"""
#Start the main app loop which will update the app based on user input
while appExit == False:
    mousePos = [0,0]

    #Set the background color to white using the pygame module
    screen.fill(colors["white"])

    #Handle events through the pygame module
    for event in pygame.event.get():

        #If the exit button on the app is pressed, quit
        if event.type == QUIT:
            appExit = True

        #Store the mouse position when it is clicked
        if event.type == pygame.MOUSEBUTTONUP:
            mousePos = pygame.mouse.get_pos()

    #Draw every button defined in the btnlist variable
    for btn in btnList:
        btn.DrawButton()

    #If the mouse clicked a button then determine what to do
    if (CheckButtonPress(mousePos, btnList)) != None:

        #Determine the factors if the prime button is pressed
        if(CheckButtonPress(mousePos, btnList)) == "Prime":
            if calculatorInput != "" and "e" not in calculatorInput:
                factorList = DeterminePrime(calculatorInput)

                if len(factorList) > 0:

                    calculatorInput = "Some factors of this number are:"

                    for factor in factorList:
                        calculatorInput += f' {factor},'

                else:
                    calculatorInput = "This number is prime"

        #If the factorial button is pressed perform the "Factorial" function on the current input
        elif (CheckButtonPress(mousePos, btnList)) == "!":
            if calculatorInput != "" and "e" not in calculatorInput:
                    calculatorInput = str(Factorial(calculatorInput))

        #Clear the input if the Clear button is pressed
        elif (CheckButtonPress(mousePos, btnList)) == "Clear":
            calculatorInput = ""
```

```
197
198             #'e' is used to  avoid getting an errror due to trying to perform math operations on text
199             else:
200                 if "e" in calculatorInput:
201                     calculatorInput = ""
202                 calculatorInput += (CheckButtonPress(mousePos, btnList))
203
204         #use the draw text functions just like the buttons to constantly update the calculator input
205         DrawText(calculatorInput, colors["blue"], width/2, height/4)
206
207         #use the pygame module to update whats on the screen
208         pygame.display.update()
```

PDF document made with CodePrint using Prism