```
# Project:      HW4 (VoMikeHW4SecHY02Ver02.py)
# Name:         Mike Vo
# Date:         03/07/2017
# Description:  A dice game

# Import the required modules
from graphics import *
from random import randint

# Init and return generic button object
def InitButton(p1, p2, strLabel):

    button = [Rectangle(p1, p2)]
    button.append(Text(button[0].getCenter(), strLabel))

    return button

# Render a button object onto a GraphWin screen
def DrawButton(button, GraphWinObj):

    button[0].draw(GraphWinObj)
    button[1].draw(GraphWinObj)

# Initialize user interface and return the interface itself as an object
def InitAllButtons(GraphWinObj):

    # Create and configure Exit button
    btnExit = InitButton(Point(GraphWinObj.getWidth() / 2 + 5,
GraphWinObj.getHeight() - 50),
                         Point(GraphWinObj.getWidth() / 2 + 85,
GraphWinObj.getHeight() - 20),
                         "Exit"
                         )
    btnExit[0].setFill("white")
    btnExit[0].setOutline("gray")
    DrawButton(btnExit, GraphWinObj)

    # Create and configure Reset button
    btnReset = InitButton(Point(GraphWinObj.getWidth() / 2 - 85,
GraphWinObj.getHeight() - 50),
                          Point(GraphWinObj.getWidth() / 2 - 5,
GraphWinObj.getHeight() - 20),
                          "Reset"
                          )
    btnReset[0].setFill("white")
    btnReset[0].setOutline("gray")
    DrawButton(btnReset, GraphWinObj)

    return [
        btnExit,
        btnReset,
    ]
```

```
# Check if a single button object has been clicked given last mouse
position (Point object)
def ButtonClicked(btnRect, pMouse):

    p1 = btnRect.getP1()
    p2 = btnRect.getP2()

    if (pMouse.getX() > p1.getX() and pMouse.getX() < p2.getX()
        and pMouse.getY() > p1.getY() and pMouse.getY() < p2.getY()):
        return True

    return False

# From a list of button object return the name of the button clicked by
mouse input (Point object)
# If no button has been clicked, return ""
def CheckButtonClicked(lstButtons, pMouse):

    for button in lstButtons:
        if ButtonClicked(button[0], pMouse):
            return button[1].getText()

    return ""

# Initialize all the dices' data
def InitDices(GraphWinObj):

    intNumOfDices = 5

    # Coordinate in form of Point object for each dice
    lstDiceCoords = [
        [Point(20, 20), Point(120, 120)],
        [Point(130, 20), Point(230, 120)],
        [Point(240, 20), Point(340, 120)],
        [Point(350, 20), Point(450, 120)],
        [Point(460, 20), Point(560, 120)],
    ]

    # Initial value of every dice is 0
    lstDiceValue = [
        0, 0, 0, 0, 0
    ]

    # Render outline for each of intNumOfDice dices
    for intIndex in range(intNumOfDices):
        bound = Rectangle(lstDiceCoords[intIndex][0],
                          lstDiceCoords[intIndex][1])
        bound.setOutline("gray")
        bound.setWidth(3)
        bound.draw(GraphWinObj)

    return [
        lstDiceCoords, lstDiceValue, intNumOfDices
    ]
```

```
# Check if every dice has been thrown
def DicesFull(lstDices):

    for intIndex in range(lstDices[2]):
        if lstDices[1][intIndex] == 0:
            return False

    return True

# Check if a dice has been clicked, works based on ButtonClicked()
# If no dice has been clicked, return error code -1
def CheckDiceClick(lstDices, pMouse):

    for intIndex in range(lstDices[2]):
        btnDiceRect = Rectangle(lstDices[0][intIndex][0],
lstDices[0][intIndex][1])
        if ButtonClicked(btnDiceRect, pMouse):
            return intIndex

    return -1

# Render a single black dice dot (Circle object) at pCenter with radius
10px onto GraphWin object
# Also return the resulting dice dot as an object for compatibility with
TwoDots()
def Dot(pCenter, GraphWinObj):

    # Initialize
    pDot = Circle(pCenter, 10)
    pDot.setOutline("black")
    pDot.setFill("black")

    # Rendering
    pDot.draw(GraphWinObj)

    return pDot

# Render 2 black dice dots onto GraphWin object based from pCenter, with
the options of the dots:
#   1. Running diagonal
#   2. Flip image of 1
#   3. Running horizontal
# Works based on Dot()
# Also return the resulting 2 dice dots as a list of objects for
compatibility with further development
def TwoDots(pCenter, GraphWinObj, strOrientation="diag normal"):

    # Init and render 2 dots (rendering happens in Dot())
    pDot1 = Dot(pCenter, GraphWinObj)
    pDot2 = Dot(pCenter, GraphWinObj)

    # if-elif to select from 3 different 2 dots configuration, then apply
the effect
```

```
    if strOrientation == "diag normal":
        pDot1.move(25, -25)
        pDot2.move(-25, 25)
    elif strOrientation == "diag inverted":
        pDot1.move(-25, -25)
        pDot2.move(25, 25)
    elif strOrientation == "horizontal":
        pDot1.move(25, 0)
        pDot2.move(-25, 0)

    return [
        pDot1, pDot2
    ]

# Combine Dot() and TwoDots() to draw onto GraphWin object any number of
black dice dot from 1 to 6
def RenderDots(pCenter, intNum, GraphWinObj):

    if intNum == 1 or intNum == 5:
        Dot(pCenter, GraphWinObj)
    if intNum >= 2:
        TwoDots(pCenter, GraphWinObj)
    if intNum == 3:
        Dot(pCenter, GraphWinObj)
    if intNum >= 4:
        TwoDots(pCenter, GraphWinObj, strOrientation="diag inverted")
    if intNum == 6:
        TwoDots(pCenter, GraphWinObj, strOrientation="horizontal")

# Render the white body of a specified dice, then uses RenderDots() to
render the dots based
# on the dice's value
def RenderDice(lstDices, intDice, GraphWinObj):

    # Preparation
    p1 = lstDices[0][intDice][0]
    p2 = lstDices[0][intDice][1]

    # Render the white body
    p1.move(4, 4)
    p2.move(-4, -4)
    diceRect = Rectangle(p1, p2)
    diceRect.setFill("white")
    diceRect.setOutline("white")
    diceRect.draw(GraphWinObj)

    # Render the black border
    p1.move(2, 2)
    p2.move(-2, -2)
    diceBound = Rectangle(p1, p2)
    diceBound.setOutline("black")
    diceBound.draw(GraphWinObj)

    # Render the dots
```

```
    RenderDots(diceRect.getCenter(), lstDices[1][intDice], GraphWinObj)


# Main program
def main():

    # Main loop
    blnMainRun = True
    while blnMainRun:

        # Initialize a 580x280 khaki graphic window titled "Dice"
        # Autoflush set to False to avoid screen flickering while drawing
the dice dots
        win = GraphWin("Dice", 580, 280, autoflush=False)
        win.setBackground("khaki")

        # Initialize dice data
        lstDices = InitDices(win)
        intDiceTotal = 0

        # Initialize message
        lblMessage = Text(Point(win.getWidth() / 2, 180), "Dice
Total\n{0:0}".format(intDiceTotal))
        lblMessage.setTextColor("red")
        lblMessage.setStyle("bold")
        lblMessage.draw(win)

        # Initialize user interface
        lstButtons = InitAllButtons(win)

        # Logic loop
        blnDiceRun = True
        while blnDiceRun:

            # Display congratulations if thrown all dices
            if DicesFull(lstDices):
                lblMessage.setText("Dice Total\n{0:0}\nLuckiest Person On
Earth!".format(intDiceTotal))

            # Check mouse input
            pMouse = win.checkMouse()
            # This if to avoid the program crashing prematurely due to no
mouse click at the beginning
            if pMouse != None:

                # Check on the dices
                intDiceClicked = CheckDiceClick(lstDices, pMouse)
                if intDiceClicked != -1:

                    # If dice is not clicked yet
                    if lstDices[1][intDiceClicked] == 0:

                        # Update dice value with a random integer from 1
to 6, then add that to the total
```

```
                        lstDices[1][intDiceClicked] = randint(1, 6)
                        intDiceTotal += lstDices[1][intDiceClicked]

                        # Render the updated dice, and update message
with updated total
                        RenderDice(lstDices, intDiceClicked, win)
                        lblMessage.setText("Dice
Total\n{0:0}".format(intDiceTotal))

                        # Update the whole window
                        win.update()

                # Check on the buttons
                strButtonClicked = CheckButtonClicked(lstButtons, pMouse)
                if strButtonClicked == "Exit":
                    # Exit every loop, terminate program
                    blnDiceRun = False
                    blnMainRun = False
                elif strButtonClicked == "Reset":
                    # Exit only the logic loop, stay in the Main loop
                    blnDiceRun = False

        # Close the current graphic window for reset/exit
        win.close()

main()
```