graphics

Objects:
  Window:
  An object to control window settings, such as size, background, etc. Also provides functionality for obtaining mouse coordinates.
  Shapes:
  The included shapes include: Rectangle, Oval, Polygon, Text

API:
  Blue parameters are required, pink are optional
  Window:
  ● Window(title, width, height): Sets up a Window of the given title (a string, eg. “My Screen”), width, and height, or creates an object to interact with the current Window if one has already been created and applies the parameters accordingly.

  window = Window("My Window", 600, 600)

  ● setBackground(color): Sets the background to the given color. The parameter color can be a string (eg. “red”), or a tuple corresponding to RGB values (eg. (255, 0, 0)).

  window = Window("My Window", 600, 600)
  window.setBackground("Blue")

  ● getMouse(): Pauses until the screen is clicked, then returns the coordinates of the click.

  window = Window("My Window", 600, 600)
  window.getMouse()

  ● close(): Closes the window.

  window = Window("My Window", 600, 600)
  window.close()
**Shapes**: (these methods work for all shapes)

- **draw()**: Draws the given shape to the Window.

  ```python
  square = Rectangle((-50, -50), 100, 100)
square.draw()
  ```

- **undraw()**: Undraws the given shape from the Window.

  ```python
  square = Rectangle((-50, -50), 100, 100)
square.draw()
square.undraw()
  ```

- **move(dx, dy)**: Moves the given shape on the window dx units to the right (or |dx| units to the left, if negative) and dy units upward (or |dy| units downward, if negative).

  ```python
  square = Rectangle((-50, -50), 100, 100)
square.draw()
square.move(50, 50)
  ```

- **rotate(dTheta)**: Rotates the given shape dTheta degrees counterclockwise (or |dTheta| degree clockwise if negative) about the shape's center.

  ```python
  square = Rectangle((-50, -50), 100, 100)
square.draw()
square.rotate(30)
  ```

- **containsPoint(point)**: Returns True if the given point is contained within the boundary of the shape, otherwise returns False.

  ```python
  square = Rectangle((-50, -50), 100, 100)
square.draw()

  # Prints True
  print(square.containsPoint((0, 0))

  # Prints False
  print(square.containsPoint((-100, -100)))
  ```
- **setLabel**(text, size, font, color, offset): Attaches a Text object to the shape at an offset offset (eg. (3, 7) from the center. The optional parameters size, font, and color can be used to control the font.

  ```python
  square = Rectangle((-50, -50), 100, 100)
square.draw()
square.setLabel("My Label", 12, "Helvetica", "Red", (0, -8))
  ```

- **removeLabel**(): Removes the label from the shape.

  ```python
  square = Rectangle((-50, -50), 100, 100)
square.draw()
square.setLabel("My Label")
square.removeLabel()
  ```

- **setFill**(color): Configures the given shape to be filled with the given parameter color.

  ```python
  square = Rectangle((-50, -50), 100, 100)
square.draw()
square.setFill("Red")
  ```

- **setLineColor**(color): Configures the given shape to have lines colored by the given color.

  ```python
  square = Rectangle((-50, -50), 100, 100)
square.draw()
square.setLineColor("Red")
  ```

- **setLineWidth**(color): Configures the given shape to have lines of the given width.

  ```python
  square = Rectangle((-50, -50), 100, 100)
square.draw()
square.setLineWidth(2)
  ```

- **getPoints**(): Returns the list of points defining this shape.
square = Rectangle((-50, -50), 100, 100)
print(square.getPoints())

**Rectangle**: (in addition to the methods listed for Shapes)

- **Rectangle**(lowerLeftCorner, width, height, lineWidth, lineColor, fillColor): Creates a Rectangle object of the given width and height where lowerLeftCorner is a tuple (eg. (100, 50)) designating the lower left corner of the Rectangle. The optional parameters lineWidth, lineColor, and fillColor control the line width, line color, and fill color of the Rectangle.

square = Rectangle((-50, -50), 100, 100)

**Oval**: (in addition to the methods listed for Shapes)

- **Oval**(center, width, height, steps, lineWidth, lineColor, fillColor): Creates an Oval object of the given width and height where center is a tuple (eg. (100, 50)) designating the center of the Oval. The optional parameter steps controls how many points will be used to create the outline of shape; by default, this value is 500. The remaining optional parameters lineWidth, lineColor, and fillColor control the line width, line color, and fill color of the Oval.

circle = Oval((0, 0), 100, 100, 500)

**Polygon**: (in addition to the methods listed for Shapes)

- **Polygon**(points, lineWidth, lineColor, fillColor): Creates a Polygon object using the given list points (eg. [(0, 0), (100, 0), (100, 100), (0, 100)]). Note that the list points will be iterated through as given, so order matters. The optional parameters lineWidth, lineColor, and fillColor control the line width, line color, and fill color of the Rectangle.

    import math
    listOfPoints = []
    for i in range(5):
        listOfPoints.append((
            50*math.cos(i * 2 * math.pi / 5),
            50*math.sin(i * 2 * math.pi / 5)))
50*\text{math.sin}(i * 2 * \text{math.pi} / 5))
pentagon = \text{Polygon}(\text{listOfPoints})
pentagon.draw()

\textbf{Text:} (in addition to the methods listed for Shapes)
- \textbf{Text}(\text{text}, \text{pos}, \text{size}, \text{font}, \text{fontColor}): Creates a Text object using the given string \text{text}, centered at the tuple \text{pos}, where \text{size}, \text{font}, and \text{fontColor} configure the font of the text. Many functions for Shapes are disabled for Text objects (including rotate).

\text{text} = \text{Text}(\text{"Some text"}, (0, 0), 14, \text{"Arial"}, \text{"Blue"})
\text{text}.draw()