Canvas Class Documentation

A Python drawing library built on top of PIL (Pillow) for creating 2D graphics with a simple, chainable API.

Installation Requirements

```
pip install Pillow
```

Quick Start

```
from canvas import Canvas

# Create a canvas and draw some shapes
canvas = Canvas(800, 600, 'lightblue')
canvas.set_pencolor('red').set_penwidth(3).set_fillcolor('yellow')
canvas.rectangle(100, 100, 300, 200)
canvas.circle(400, 300, 50)
canvas.text(400, 400, "Hello World!", center=True)
canvas.show()  # Display the image
```

Class Reference

Canvas(width=500, height=500, color='white')

Creates a new drawing canvas with the specified dimensions and background color.

Parameters: - `width` (int, optional): Canvas width in pixels. Default: 500 - `height` (int, optional): Canvas height in pixels. Default: 500
- `color` (str, optional): Background color name or hex code. Default: 'white'

Example:

```
canvas = Canvas(800, 600, 'lightgray')
```

Properties and State Management

Drawing Properties

The Canvas maintains several drawing properties that affect how shapes are rendered:

- **Pen Color**: Color used for outlines and lines
- **Fill Color**: Color used to fill shapes (None for no fill)
- **Pen Width**: Thickness of lines and outlines
- **Font**: Font used for text rendering

Property Setters (Chainable)

set_pencolor(color) Sets the color for drawing lines and shape outlines.

Parameters: - `color` (str): Color name ('red', 'blue') or hex code ('#FF0000')

Returns: Canvas instance (for method chaining)

set_penwidth(width) Sets the width/thickness of lines and shape outlines.

Parameters: - `width` (int): Line width in pixels

Returns: Canvas instance (for method chaining)

Note: There's a bug in the current implementation - it sets `self.width` instead of `self.penwidth`.

set_fillcolor(color) Sets the fill color for shapes. Set to None for no fill.

Parameters: - `color` (str or None): Fill color or None for transparent fill

Returns: Canvas instance (for method chaining)

set_font(fontname, fontsize) Sets the font for text rendering.

Parameters: - `fontname` (str): Font file name or system font name (e.g., "Arial") - `fontsize` (int): Font size in points

Returns: Canvas instance (for method chaining)

Example:

```
canvas.set_font("Times New Roman", 16)
```

Drawing Methods

Basic Shapes

line(x0, y0, x1, y1) Draws a line from point (x0, y0) to point (x1, y1).

Parameters: - `x0, y0` (int): Starting point coordinates - `x1, y1` (int): Ending point coordinates

Returns: Canvas instance (for method chaining)

rectangle(x0, y0, x1, y1) Draws a rectangle with opposite corners at (x0, y0) and (x1, y1).

Parameters: - `x0, y0` (int): First corner coordinates - `x1, y1` (int): Opposite corner coordinates

Returns: Canvas instance (for method chaining)

Note: Corner order doesn't matter - the method automatically adjusts the bounding box.

circle(x, y, r) Draws a circle centered at (x, y) with radius r.

Parameters: - `x, y` (int): Center coordinates - `r` (int): Radius in pixels

Returns: Canvas instance (for method chaining)

ellipse(x, y, r1, r2) Draws an ellipse centered at (x, y) with horizontal radius r1 and vertical radius r2.

Parameters: - `x, y` (int): Center coordinates - `r1` (int): Horizontal radius - `r2` (int): Vertical radius

Returns: Canvas instance (for method chaining)

Advanced Drawing

point(x, y) Draws a single pixel point at the specified coordinates.

Parameters: - `x, y` (int): Point coordinates

Returns: Canvas instance (for method chaining)

lines(xy) Draws connected line segments through multiple points.

Parameters: - `xy` (list): List of coordinate tuples [(x1,y1), (x2,y2), ...]

Returns: Canvas instance (for method chaining)

Example:

```
points = [(100, 100), (200, 150), (300, 100), (400, 200)]
canvas.lines(points)
```

polygon(xy) Draws a filled polygon connecting the specified points.

Parameters: - `xy` (list): List of coordinate tuples defining the polygon vertices

Returns: Canvas instance (for method chaining)

Example:

```
triangle = [(100, 100), (200, 100), (150, 50)]
canvas.polygon(triangle)
```

Text Rendering

text(x, y, text, center=False) Draws text at the specified position.

Parameters: - `x, y` (int): Text position coordinates - `text` (str): Text string to draw - `center` (bool, optional): If True, centers text at (x,y). If False, positions at top-left. Default: False

Returns: Canvas instance (for method chaining)

Example:

```
canvas.text(100, 50, "Top-left aligned")
canvas.text(400, 300, "Centered text", center=True)
```

Fill Operations

fill(x, y) Flood fills an area starting from point (x, y) with the current fill color.

Parameters: - `x, y` (int): Starting point for flood fill

Returns: Canvas instance (for method chaining)

Note: Similar to the "paint bucket" tool in image editors.

Utility Methods

Canvas Information

get_width() Returns the canvas width in pixels.

Returns: int

get_height() Returns the canvas height in pixels.

Returns: int

Output Methods

show() Returns the PIL Image object for display or further processing.

Returns: PIL.Image object

Example:

```
img = canvas.show()
img.show()  # Display in default image viewer
```

tofile(filename) Saves the canvas to an image file.

Parameters: - `filename` (str): Output filename with extension (e.g., "drawing.png")

Example:

```
canvas.tofile("my_drawing.png")
```

Method Chaining

All drawing and property methods return the Canvas instance, allowing for fluent method chaining:

```
canvas.set_pencolor('blue').set_penwidth(2).set_fillcolor('yellow') \
      .rectangle(50, 50, 150, 100) \
      .circle(200, 75, 25) \
      .text(100, 200, "Chained methods!", center=True)
```

Color Formats

Colors can be specified as: - Named colors: `'red'`, `'blue'`, `'green'`, `'white'`, `'black'`, etc. - Hex codes: `'#FF0000'` (red), `'#00FF00'` (green), `'#0000FF'` (blue) - RGB tuples: `(255, 0, 0)` for red

Complete Example

```
from canvas import Canvas

# Create a 800x600 canvas with light blue background
canvas = Canvas(800, 600, 'lightblue')

# Set drawing properties
canvas.set_pencolor('darkblue').set_penwidth(3)

# Draw a house
canvas.set_fillcolor('brown')
canvas.rectangle(300, 400, 500, 300)  # House body

canvas.set_fillcolor('red')
# House roof (triangle as polygon)
roof_points = [(280, 300), (520, 300), (400, 200)]
canvas.polygon(roof_points)

canvas.set_fillcolor('lightblue')
canvas.rectangle(350, 350, 400, 400)  # Door

canvas.set_fillcolor('yellow')
canvas.rectangle(420, 330, 470, 360)  # Window

# Add some decorative elements
canvas.set_pencolor('green').set_fillcolor('green')
canvas.circle(200, 450, 30)  # Tree
canvas.circle(600, 480, 25)  # Bush

# Add title text
canvas.set_pencolor('black')
canvas.set_font("Arial", 24)
canvas.text(400, 100, "My House", center=True)

# Save the drawing
canvas.tofile("house_drawing.png")

# Display the result
canvas.show()
```