Drawing with Py5

Complete Guide to Graphics Programming

Course Topics:

- Basic Shapes
- Colors & Styles
- Transformations
- Curves & Custom Shapes
- Images & Text
- Advanced Techniques

What You'll Learn

✅ Draw basic shapes (circles, rectangles, lines)
✅ Use colors and styling
✅ Apply transformations (rotate, scale, translate)
✅ Create custom shapes and curves
✅ Work with images and text
✅ Build interactive graphics
✅ Create animations

Prerequisites: Basic Python knowledge

Course Outline

1. **Setup & Canvas**
2. **Basic Shapes**
3. **Colors & Fills**
4. **Lines & Strokes**
5. **Transformations**
6. **Custom Shapes**
7. **Curves**
8. **Images**
9. **Text**
10. **Animation**
11. **Interactivity**

Part 1: Setup & Canvas

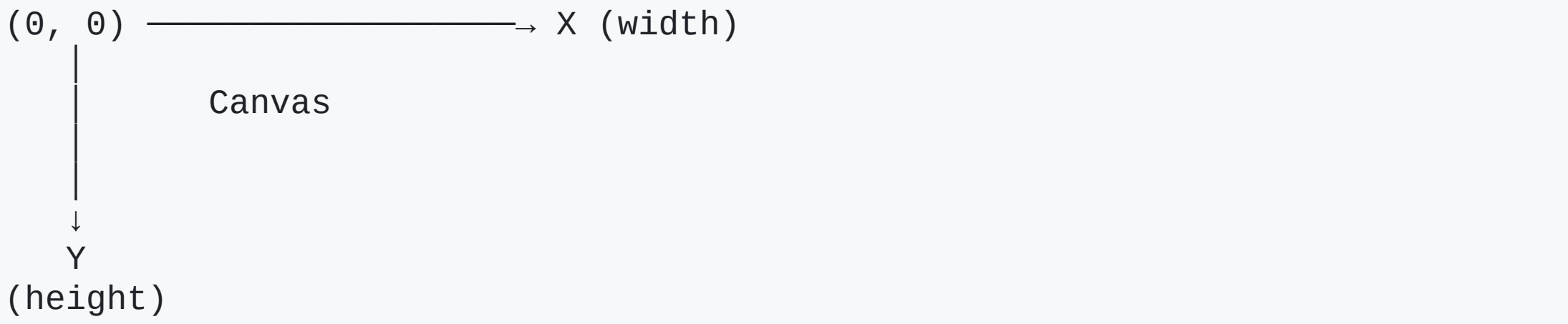
Installing Py5

```
# Install py5
pip install py5

# Verify installation
python -c "import py5; print(py5.__version__)"
```

Quick Test:

```
import py5

def setup():
    py5.size(600, 400)

def draw():
    py5.background(220)
    py5.circle(300, 200, 100)

py5.run_sketch()
```

The Canvas Coordinate System



- **Origin (0, 0)** is top-left corner
- **X increases** going right
- **Y increases** going down

Setup and Draw Functions

```
import py5

def setup():
    # Runs ONCE at start
    py5.size(800, 600)  # Set canvas size
    py5.background(255) # White background

def draw():
    # Runs REPEATEDLY (default 60 times/sec)
    py5.circle(py5.mouse_x, py5.mouse_y, 50)

py5.run_sketch()
```

Key Concepts:

- `setup()` : Initialize your sketch
- `draw()` : Animation loop
- `py5.run_sketch()` : Start the sketch

Canvas Size Options

```
# Fixed size window
py5.size(800, 600)

# Square canvas
py5.size(600, 600)

# Fullscreen
py5.full_screen()

# With renderer
py5.size(800, 600, py5.P2D)  # 2D renderer
py5.size(800, 600, py5.P3D)  # 3D renderer
```

Part 2: Basic Shapes

Points and Lines

Point:

```
py5.point(x, y)
```

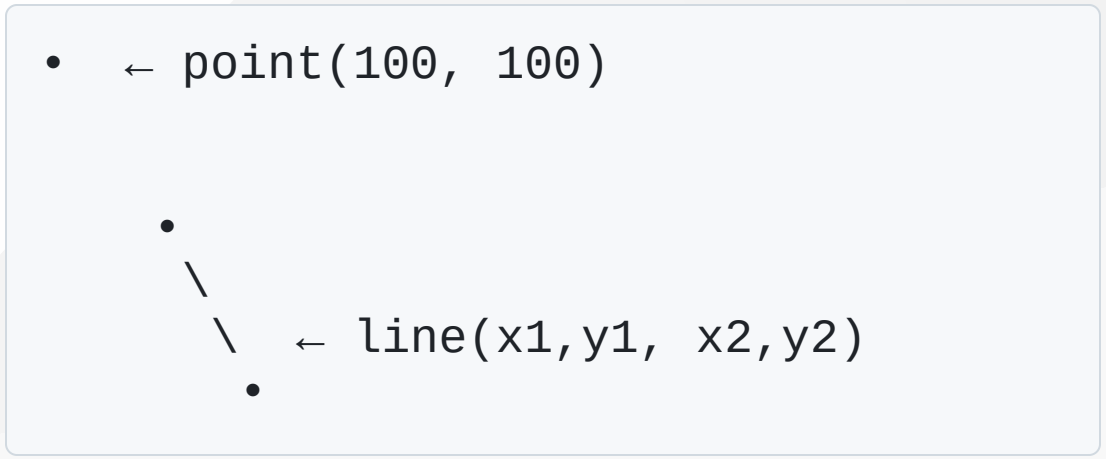
Line:

```
py5.line(x1, y1, x2, y2)
```

Example:

```
py5.point(100, 100)
py5.line(50, 50, 200, 200)
```

Visual:



Rectangles

```
# Basic rectangle
py5.rect(x, y, width, height)

# Examples:
py5.rect(100, 100, 200, 150)  # Rectangle at (100,100)

# Rectangle with rounded corners
py5.rect(100, 100, 200, 150, 20)  # 20px corner radius
```

Anchor Modes:

- `py5.rect_mode(py5.CORNER)` - x,y is top-left (default)
- `py5.rect_mode(py5.CENTER)` - x,y is center
- `py5.rect_mode(py5.CORNERS)` - x,y to x2,y2

Circles and Ellipses

```
# Circle (equal width and height)
py5.circle(x, y, diameter)
py5.circle(300, 200, 100)

# Ellipse (can be oval)
py5.ellipse(x, y, width, height)
py5.ellipse(300, 200, 150, 100)
```

Circle

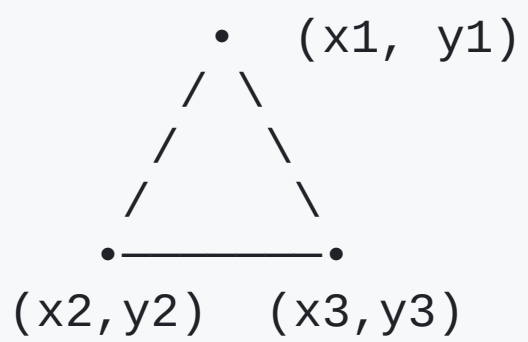
Equal diameter

Ellipse

Width ≠ Height

Triangles

```
# Triangle with three points
py5.triangle(x1, y1, x2, y2, x3, y3)

# Example: pointing up
py5.triangle(250, 100,   # Top point
             150, 300,   # Bottom left
             350, 300)   # Bottom right
```



Quads (4-sided shapes)

```
# Quadrilateral - any 4-sided shape
py5.quad(x1, y1, x2, y2, x3, y3, x4, y4)

# Example: diamond
py5.quad(250, 100,   # Top
         350, 200,   # Right
         250, 300,   # Bottom
         150, 200)   # Left
```

Use for: Parallelograms, trapezoids, irregular shapes

Arcs

```
# Arc (part of a circle/ellipse)
py5.arc(x, y, width, height, start, stop)

# Examples:
py5.arc(300, 200, 200, 200, 0, py5.PI)        # Semi-circle
py5.arc(300, 200, 200, 200, 0, py5.HALF_PI) # Quarter circle
py5.arc(300, 200, 200, 200, 0, py5.TWO_PI)  # Full circle

# Arc modes:
py5.arc(x, y, w, h, start, stop, py5.PIE)    # Pie slice
py5.arc(x, y, w, h, start, stop, py5.CHORD)  # Chord
py5.arc(x, y, w, h, start, stop, py5.OPEN)   # Open arc
```

Shape Drawing Example

```
import py5

def setup():
    py5.size(800, 600)
    py5.background(220)

    # Draw various shapes
    py5.circle(200, 150, 100)           # Circle
    py5.rect(350, 100, 150, 100)        # Rectangle
    py5.triangle(600, 100, 550, 200, 650, 200)  # Triangle
    py5.line(100, 400, 700, 400)        # Line
    py5.point(400, 300)                 # Point

py5.run_sketch()
```

Part 3: Colors & Fills

RGB Color Model

Red

0-255

Green

0-255

Blue

0-255

```
# RGB format
py5.fill(255, 0, 0)       # Pure red
py5.fill(0, 255, 0)       # Pure green
py5.fill(0, 0, 255)       # Pure blue
py5.fill(255, 255, 0)     # Yellow (red + green)
```

Fill and Stroke

```
# Fill (interior color)
py5.fill(255, 100, 150)   # RGB
py5.circle(200, 200, 100)

# No fill
py5.no_fill()
py5.circle(400, 200, 100)

# Stroke (outline color)
py5.stroke(0, 0, 255)     # Blue outline
py5.circle(600, 200, 100)

# No stroke
py5.no_stroke()
py5.circle(800, 200, 100)
```

Remember: Set fill/stroke BEFORE drawing the shape!

Grayscale and Transparency

```
# Grayscale (one value)
py5.fill(128)         # Medium gray
py5.fill(0)           # Black
py5.fill(255)         # White

# With transparency (alpha)
py5.fill(255, 0, 0, 128)    # Semi-transparent red
#     R    G   B   Alpha (0-255)

# Transparent shapes
py5.fill(0, 0, 255, 100)    # 100/255 = ~40% opacity
py5.circle(300, 200, 150)
```

Alpha: 0 = fully transparent, 255 = fully opaque

Background Colors

```
def draw():
    # Clear canvas each frame
    py5.background(220)                # Gray
    py5.background(255, 200, 200)      # Light pink
    py5.background(0)                  # Black

    # No background = trails effect!
    # (don't call background() to keep previous frames)
```

Tip: Call `background()` in `draw()` to clear previous frames

Color Example

```
import py5

def setup():
    py5.size(800, 400)

def draw():
    py5.background(240)

    # Red circle with blue outline
    py5.fill(255, 0, 0)
    py5.stroke(0, 0, 255)
    py5.stroke_weight(3)
    py5.circle(200, 200, 150)

    # Semi-transparent green square, no outline
    py5.fill(0, 255, 0, 128)
    py5.no_stroke()
    py5.rect(400, 125, 150, 150)

py5.run_sketch()
```

Part 4: Lines & Strokes

Stroke Weight

```
# Set line thickness
py5.stroke_weight(1)    # Thin (default)
py5.line(100, 100, 300, 100)

py5.stroke_weight(5)    # Medium
py5.line(100, 150, 300, 150)

py5.stroke_weight(10)   # Thick
py5.line(100, 200, 300, 200)

py5.stroke_weight(20)   # Very thick
py5.line(100, 250, 300, 250)
```

Affects: Lines, shape outlines, points

Stroke Cap and Join

Stroke Cap:

```
# Line endings
py5.stroke_cap(py5.ROUND)
py5.stroke_cap(py5.SQUARE)
py5.stroke_cap(py5.PROJECT)
```

Stroke Join:

```
# Corner style
py5.stroke_join(py5.MITER)
py5.stroke_join(py5.BEVEL)
py5.stroke_join(py5.ROUND)
```

ROUND: ●——●

SQUARE: ▪——▪

PROJECT: ■——■

MITER: ┐

BEVEL: ⌐

ROUND: ╮

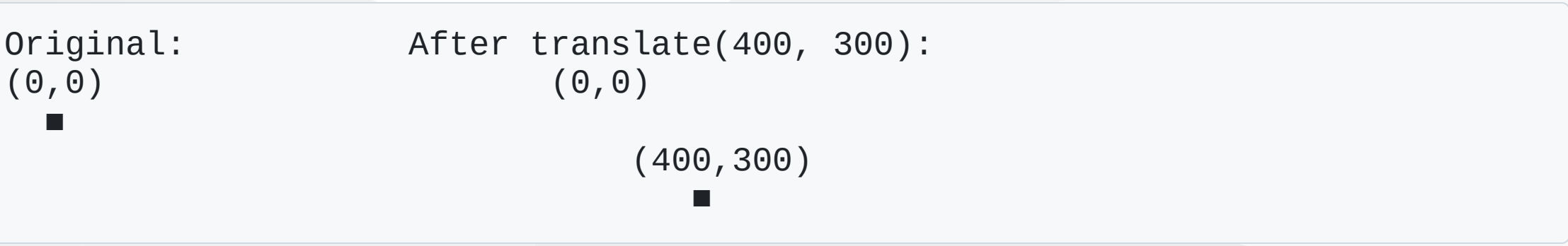
Drawing Multiple Lines

```
def setup():
    py5.size(600, 400)
    py5.background(255)
    py5.stroke(0)
    py5.stroke_weight(2)

    # Grid pattern
    for x in range(0, 600, 50):
        py5.line(x, 0, x, 400)

    for y in range(0, 400, 50):
        py5.line(0, y, 600, y)

py5.run_sketch()
```

Part 5: Transformations

Translation (Moving)

```
# Move the origin point
py5.translate(x, y)

# Example: Draw at new origin
py5.translate(400, 300)  # Move origin to center
py5.rect(0, 0, 100, 100) # Now draws at (400, 300)
```



Rotation

```
# Rotate around origin (in radians)
py5.rotate(angle)

# Example: Rotating square
py5.translate(400, 300)      # Move to center first
py5.rotate(py5.PI / 4)       # Rotate 45 degrees
py5.rect(-50, -50, 100, 100) # Draw centered on origin
```

Common angles:

- `py5.PI / 4` = 45°
- `py5.PI / 2` = 90°
- `py5.PI` = 180°
- `py5.TWO_PI` = 360°

Convert degrees: `py5.radians(45)`

Scaling

```
# Scale (resize) everything
py5.scale(factor)
py5.scale(x_factor, y_factor)

# Example: Double size
py5.scale(2.0)
py5.rect(100, 100, 50, 50)  # Draws as 100x100

# Example: Flip horizontally
py5.scale(-1, 1)
```

Scale values:

- `< 1.0` = shrink
- `1.0` = normal
- `> 1.0` = grow
- Negative = flip

Push and Pop Matrix

```
def draw():
    py5.background(220)

    # Regular drawing
    py5.rect(100, 100, 50, 50)

    # Save state
    py5.push_matrix()
    py5.translate(400, 300)
    py5.rotate(py5.PI / 4)
    py5.rect(0, 0, 50, 50)  # Rotated square
    py5.pop_matrix()  # Restore original state

    # Back to normal
    py5.rect(600, 100, 50, 50)
```

Use `push/pop` to isolate transformations!

Transformation Example

```
import py5

angle = 0

def setup():
    py5.size(800, 600)

def draw():
    global angle
    py5.background(220)

    # Spinning square
    py5.push_matrix()
    py5.translate(400, 300)
    py5.rotate(angle)
    py5.rect_mode(py5.CENTER)
    py5.rect(0, 0, 100, 100)
    py5.pop_matrix()

    angle += 0.05

py5.run_sketch()
```

Part 6: Custom Shapes

Begin Shape / End Shape

```
# Create custom shapes
py5.begin_shape()
py5.vertex(x1, y1)
py5.vertex(x2, y2)
py5.vertex(x3, y3)
# ... more vertices
py5.end_shape()

# Close the shape
py5.end_shape(py5.CLOSE)
```

Use for: Polygons, stars, custom designs

Custom Shape Example - Star

```
def draw_star(x, y, radius):
    py5.push_matrix()
    py5.translate(x, y)
    py5.begin_shape()

    for i in range(10):
        angle = py5.TWO_PI / 10 * i
        r = radius if i % 2 == 0 else radius / 2
        px = py5.cos(angle) * r
        py5 = py5.sin(angle) * r
        py5.vertex(px, py)

    py5.end_shape(py5.CLOSE)
    py5.pop_matrix()

# Usage:
draw_star(400, 300, 100)
```

Shape Modes

```
# Different shape types
py5.begin_shape()              # Default polygon
py5.begin_shape(py5.POINTS)    # Individual points
py5.begin_shape(py5.LINES)     # Separate line segments
py5.begin_shape(py5.TRIANGLES) # Separate triangles
py5.begin_shape(py5.TRIANGLE_STRIP)
py5.begin_shape(py5.TRIANGLE_FAN)
py5.begin_shape(py5.QUADS)     # Separate quads
py5.begin_shape(py5.QUAD_STRIP)
```

Contours (Holes in Shapes)

```
def draw():
    py5.background(220)

    # Outer shape
    py5.begin_shape()
    py5.vertex(200, 200)
    py5.vertex(600, 200)
    py5.vertex(600, 400)
    py5.vertex(200, 400)

    # Inner shape (hole)
    py5.begin_contour()
    py5.vertex(300, 250)
    py5.vertex(500, 250)
    py5.vertex(500, 350)
    py5.vertex(300, 350)
    py5.end_contour()

    py5.end_shape(py5.CLOSE)
```

Part 7: Curves

Bezier Curves

```
# Cubic Bezier curve
py5.bezier(x1, y1,      # Start point
           cx1, cy1,    # Control point 1
           cx2, cy2,    # Control point 2
           x2, y2)      # End point

# Example:
py5.bezier(100, 300,    # Start
           150, 100,    # Pull up-right
           450, 100,    # Pull up-left
           500, 300)    # End
```

Control points "pull" the curve toward them

Curve Vertices

```
# Smooth curves through points
py5.begin_shape()
py5.curve_vertex(100, 300)  # Control point (not drawn)
py5.curve_vertex(200, 200)  # Start
py5.curve_vertex(400, 150)  # Middle
py5.curve_vertex(600, 250)  # End
py5.curve_vertex(700, 200)  # Control point (not drawn)
py5.end_shape()
```

First and last points are control points only

Quadratic Bezier

```
# Simpler curve with one control point
py5.begin_shape()
py5.vertex(100, 300)                      # Start
py5.quadratic_vertex(300, 100, 500, 300) # Control + End
py5.end_shape()
```

Easier than cubic bezier for simple curves

Curve Tightness

```
# Adjust curve smoothness
py5.curve_tightness(0)    # Loose (default)
py5.curve_tightness(1)    # Tight
py5.curve_tightness(-1)   # Very loose

# Then draw curves
py5.begin_shape()
py5.curve_vertex(...)
# ...
py5.end_shape()
```

Part 8: Images

Loading Images

```
# Load image in setup
img = None

def setup():
    global img
    py5.size(800, 600)
    img = py5.load_image("photo.jpg")  # From data folder

def draw():
    py5.background(220)
    py5.image(img, 0, 0)  # Draw at (0, 0)
```

Supported formats: JPG, PNG, GIF

Image Display Options

```
# Draw at position
py5.image(img, x, y)

# Draw with size
py5.image(img, x, y, width, height)

# Example: Resize image
py5.image(img, 100, 100, 400, 300)

# Get image dimensions
w = img.width
h = img.height
```

Image Transparency

```
# Set image transparency
py5.tint(255, 128)  # 50% transparent

# Color tint
py5.tint(255, 100, 100)  # Reddish tint

# Remove tint
py5.no_tint()

# Then draw image
py5.image(img, x, y)
```

Image Mode

```
# Change how x, y, width, height work
py5.image_mode(py5.CORNER)  # Default: x,y = top-left
py5.image_mode(py5.CENTER)  # x,y = center
py5.image_mode(py5.CORNERS) # x,y to x2,y2

# Example: Center an image
py5.image_mode(py5.CENTER)
py5.image(img, py5.width/2, py5.height/2)
```

Part 9: Text

Drawing Text

```
# Basic text
py5.text("Hello, World!", x, y)

# Text with width box
py5.text("Longer text here...", x, y, width, height)

# Example:
py5.text("Hello!", 100, 100)
```

Position: x, y is the baseline of the text

Text Properties

```
# Text size
py5.text_size(32)

# Text alignment
py5.text_align(py5.LEFT)    # Default
py5.text_align(py5.CENTER)
py5.text_align(py5.RIGHT)

# Vertical alignment (optional second parameter)
py5.text_align(py5.CENTER, py5.CENTER)

# Example:
py5.text_size(48)
py5.text_align(py5.CENTER)
py5.text("Centered Text", 400, 300)
```

Text Fonts

```
# Load custom font
font = None

def setup():
    global font
    py5.size(800, 600)
    font = py5.create_font("Arial", 32)
    py5.text_font(font)

def draw():
    py5.background(220)
    py5.text("Custom Font!", 100, 100)
```

Available fonts: System fonts on your computer

Text Styling

```
# Text color
py5.fill(255, 0, 0)  # Red text
py5.text("Red text", 100, 100)

# Text with outline
py5.fill(255)
py5.stroke(0)
py5.stroke_weight(2)
py5.text("Outlined", 100, 150)

# Get text width
w = py5.text_width("Hello")
```

Dynamic Text Example

```
import py5

def setup():
    py5.size(800, 400)
    py5.text_size(32)
    py5.text_align(py5.CENTER, py5.CENTER)

def draw():
    py5.background(220)

    # Show mouse position
    py5.fill(0)
    message = f"Mouse: ({py5.mouse_x}, {py5.mouse_y})"
    py5.text(message, py5.width/2, py5.height/2)

py5.run_sketch()
```

Part 10: Animation

Frame Rate

```
def setup():
    py5.size(800, 600)
    py5.frame_rate(60)  # 60 frames per second (default)
    # py5.frame_rate(30)  # Slower
    # py5.frame_rate(120) # Faster

# Check current frame rate
fps = py5.get_frame_rate()
```

Higher frame rate = smoother but more CPU intensive

Animation Variables

```
import py5

x = 0

def setup():
    py5.size(800, 400)

def draw():
    global x
    py5.background(220)

    # Moving circle
    py5.circle(x, 200, 50)

    # Update position
    x += 5

    # Wrap around
    if x > py5.width:
        x = 0

py5.run_sketch()
```

Easing and Interpolation

```
# Smooth movement toward target
target_x = py5.mouse_x
current_x = 0
easing = 0.1  # Smoothness (0.0 to 1.0)

def draw():
    global current_x
    py5.background(220)

    # Ease toward target
    current_x += (target_x - current_x) * easing

    py5.circle(current_x, 200, 50)
```

Smaller easing = slower, smoother movement

Using `lerp()` for Animation

```
# Linear interpolation
py5.lerp(start, stop, amount)

# Example: Fade from black to white
amount = 0

def draw():
    global amount
    color = py5.lerp(0, 255, amount)
    py5.background(color)

    amount += 0.01
    if amount > 1:
        amount = 0
```

amount: 0.0 = start, 1.0 = stop

Rotation Animation

```
import py5

angle = 0

def setup():
    py5.size(600, 600)

def draw():
    global angle
    py5.background(220)

    py5.push_matrix()
    py5.translate(300, 300)
    py5.rotate(angle)
    py5.rect_mode(py5.CENTER)
    py5.fill(100, 150, 255)
    py5.rect(0, 0, 150, 150)
    py5.pop_matrix()

    angle += 0.05

py5.run_sketch()
```

Part 11: Interactivity

Mouse Position

```
# Built-in variables
py5.mouse_x  # Current X position
py5.mouse_y  # Current Y position
py5.pmouse_x # Previous X position
py5.pmouse_y # Previous Y position

# Example: Drawing app
def draw():
    py5.line(py5.pmouse_x, py5.pmouse_y,
             py5.mouse_x, py5.mouse_y)
```

Mouse Events

```
def mouse_pressed():
    # When mouse button is pressed
    print(f"Clicked at {py5.mouse_x}, {py5.mouse_y}")

def mouse_released():
    # When mouse button is released
    pass

def mouse_clicked():
    # When mouse is clicked (pressed + released)
    pass

def mouse_moved():
    # When mouse moves (button not pressed)
    pass

def mouse_dragged():
    # When mouse moves (button pressed)
    pass
```

Mouse Buttons

```
def mouse_pressed():
    if py5.mouse_button == py5.LEFT:
        print("Left click")
    elif py5.mouse_button == py5.RIGHT:
        print("Right click")
    elif py5.mouse_button == py5.CENTER:
        print("Middle click")

# Check if mouse is pressed
if py5.is_mouse_pressed:
    # Do something while mouse is down
    pass
```

Keyboard Input

```
def key_pressed():
    print(f"Pressed: {py5.key}")

    # Check specific keys
    if py5.key == 'a':
        print("A pressed")
    elif py5.key == ' ':
        print("Space pressed")

def key_released():
    print(f"Released: {py5.key}")

# Check if any key is pressed
if py5.is_key_pressed:
    # Do something while key is down
    pass
```

Special Keys

```
def key_pressed():
    # For special keys, check key_code
    if py5.key == py5.CODED:
        if py5.key_code == py5.UP:
            print("Up arrow")
        elif py5.key_code == py5.DOWN:
            print("Down arrow")
        elif py5.key_code == py5.LEFT:
            print("Left arrow")
        elif py5.key_code == py5.RIGHT:
            print("Right arrow")
        elif py5.key_code == py5.SHIFT:
            print("Shift")
```

Interactive Drawing Example

```
import py5

def setup():
    py5.size(800, 600)
    py5.background(255)

def draw():
    # Draw when mouse is pressed
    if py5.is_mouse_pressed:
        if py5.mouse_button == py5.LEFT:
            py5.fill(0)
        else:
            py5.fill(255)  # Erase
        py5.no_stroke()
        py5.circle(py5.mouse_x, py5.mouse_y, 20)

def key_pressed():
    if py5.key == 'c':
        py5.background(255)  # Clear

py5.run_sketch()
```

Part 12: Advanced Techniques

Random Values

```
# Random float
r = py5.random(10)          # 0.0 to 10.0
r = py5.random(5, 10)       # 5.0 to 10.0

# Random integer
i = int(py5.random(10))     # 0 to 9

# Example: Random circles
def draw():
    x = py5.random(py5.width)
    y = py5.random(py5.height)
    py5.circle(x, y, 50)
```

Noise (Perlin Noise)

```
# Smooth random values
n = py5.noise(x)
n = py5.noise(x, y)
n = py5.noise(x, y, z)

# Example: Organic movement
offset = 0

def draw():
    global offset
    x = py5.noise(offset) * py5.width
    y = py5.noise(offset + 100) * py5.height
    py5.circle(x, y, 50)
    offset += 0.01
```

Noise is smoother than random!

Map Function

```
# Remap value from one range to another
value = py5.remap(value, start1, stop1, start2, stop2)

# Example: Map mouse X to circle size
size = py5.remap(py5.mouse_x, 0, py5.width, 10, 200)
py5.circle(400, 300, size)

# Constrain to range
value = py5.constrain(value, min_val, max_val)
```

Distance and Angles

```
# Distance between two points
d = py5.dist(x1, y1, x2, y2)

# Angle from point 1 to point 2
angle = py5.atan2(y2 - y1, x2 - x1)

# Example: Point toward mouse
angle = py5.atan2(py5.mouse_y - 300,
                  py5.mouse_x - 400)
py5.push_matrix()
py5.translate(400, 300)
py5.rotate(angle)
py5.rect(0, -10, 50, 20)
py5.pop_matrix()
```

Blend Modes

```
# Change how colors mix
py5.blend_mode(py5.BLEND)      # Default
py5.blend_mode(py5.ADD)        # Additive
py5.blend_mode(py5.SUBTRACT)   # Subtractive
py5.blend_mode(py5.MULTIPLY)   # Multiply
py5.blend_mode(py5.SCREEN)     # Screen
py5.blend_mode(py5.OVERLAY)    # Overlay

# Example:
py5.blend_mode(py5.ADD)
py5.fill(255, 0, 0, 128)
py5.circle(300, 300, 200)
py5.fill(0, 0, 255, 128)
py5.circle(350, 300, 200)
```

Save Frame

```
def key_pressed():
    if py5.key == 's':
        # Save current frame
        py5.save_frame("output.png")

        # Save with timestamp
        py5.save_frame("output-####.png")
        # Creates: output-0001.png, output-0002.png, etc.
```

Smooth and No Smooth

```
def setup():
    py5.size(800, 600)

    # Anti-aliasing (default)
    py5.smooth()

    # No anti-aliasing (faster, sharper)
    py5.no_smooth()
```

Smooth: Softer edges, slower

No Smooth: Sharp edges, faster

Part 13: Complete Examples

Example 1: Generative Art

```
import py5

def setup():
    py5.size(800, 800)
    py5.background(255)
    py5.no_fill()
    py5.stroke(0, 100)

    for i in range(100):
        x = py5.random(py5.width)
        y = py5.random(py5.height)
        size = py5.random(50, 200)
        py5.circle(x, y, size)

py5.run_sketch()
```

Example 2: Interactive Particles

```
import py5

particles = []

def setup():
    py5.size(800, 600)

def draw():
    py5.background(0, 20)  # Trails effect

    for p in particles:
        py5.fill(p['color'])
        py5.no_stroke()
        py5.circle(p['x'], p['y'], p['size'])
        p['x'] += p['vx']
        p['y'] += p['vy']

def mouse_pressed():
    particles.append({
        'x': py5.mouse_x, 'y': py5.mouse_y,
        'vx': py5.random(-3, 3), 'vy': py5.random(-3, 3),
        'size': py5.random(10, 30),
        'color': (py5.random(255), py5.random(255), py5.random(255))
    })

py5.run_sketch()
```

Example 3: Clock

```
import py5
from datetime import datetime

def setup():
    py5.size(600, 600)

def draw():
    py5.background(220)
    py5.translate(300, 300)

    now = datetime.now()
    h = now.hour % 12
    m = now.minute
    s = now.second

    # Hour hand
    py5.push_matrix()
    py5.rotate((h + m/60) * py5.TWO_PI / 12)
    py5.stroke(0)
    py5.stroke_weight(8)
    py5.line(0, 0, 0, -100)
    py5.pop_matrix()

    # Minute hand
    py5.push_matrix()
    py5.rotate((m + s/60) * py5.TWO_PI / 60)
    py5.stroke_weight(4)
    py5.line(0, 0, 0, -150)
    py5.pop_matrix()

    # Second hand
    py5.push_matrix()
    py5.rotate(s * py5.TWO_PI / 60)
    py5.stroke(255, 0, 0)
    py5.stroke_weight(2)
    py5.line(0, 0, 0, -180)
    py5.pop_matrix()

py5.run_sketch()
```

Example 4: Grid Pattern

```
import py5

def setup():
    py5.size(800, 800)
    py5.background(255)
    py5.no_fill()

    spacing = 50

    for x in range(0, py5.width, spacing):
        for y in range(0, py5.height, spacing):
            size = py5.dist(x, y, py5.width/2, py5.height/2)
            size = py5.remap(size, 0, 400, 40, 5)

            hue = py5.remap(size, 5, 40, 0, 360)
            py5.color_mode(py5.HSB, 360, 100, 100)
            py5.stroke(hue, 80, 80)
            py5.stroke_weight(2)

            py5.circle(x, y, size)

py5.run_sketch()
```

Tips & Best Practices

Performance Tips

✅ DO:

- Call `background()` at the start of `draw()`
- Use `no_stroke()` when you don't need outlines
- Use smaller canvas sizes for complex animations
- Use `py5.P2D` renderer for 2D graphics

❌ DON'T:

- Load images in `draw()` (load in `setup()`)
- Create objects in `draw()` unnecessarily
- Draw thousands of shapes without optimization

Code Organization

```
# Good structure
import py5

# Global variables
x = 0
y = 0

# Helper functions
def draw_star(x, y, size):
    # Draw a star
    pass

# Setup
def setup():
    py5.size(800, 600)

# Main loop
def draw():
    py5.background(220)
    draw_star(400, 300, 50)

# Event handlers
def mouse_pressed():
    pass

py5.run_sketch()
```

Common Mistakes

1. Forgetting to set fill/stroke before drawing

```
# Wrong:
py5.circle(100, 100, 50)
py5.fill(255, 0, 0)

# Right:
py5.fill(255, 0, 0)
py5.circle(100, 100, 50)
```

2. Not using `global` for variables

```
x = 0

def draw():
    global x  # Don't forget!
    x += 1
```

Debugging Tips

Print values:

```
print(f"x: {x}, y: {y}")
```

Visual debugging:

```
# Draw helper lines
py5.stroke(255, 0, 0)
py5.line(0, py5.height/2, py5.width, py5.height/2)  # Center line
```

Frame rate:

```
print(f"FPS: {py5.get_frame_rate():.1f}")
```

Color Modes

```
# RGB (default)
py5.color_mode(py5.RGB, 255)
py5.fill(255, 0, 0)  # Red

# HSB (Hue, Saturation, Brightness)
py5.color_mode(py5.HSB, 360, 100, 100)
py5.fill(0, 100, 100)    # Red
py5.fill(120, 100, 100)  # Green
py5.fill(240, 100, 100)  # Blue
```

HSB is great for:

- Rainbows
- Color animations
- Color harmonies

Resources

- **Py5 Documentation:** https://py5coding.org
- **Py5 Examples:** https://py5coding.org/examples
- **Processing Reference:** https://processing.org/reference
- **OpenProcessing:** https://openprocessing.org (inspiration)

Books:

- "The Nature of Code" by Daniel Shiffman
- "Generative Design" by Benedikt Groß

Summary

You now know how to:

- ✅ Draw basic and custom shapes
- ✅ Use colors, fills, and strokes
- ✅ Apply transformations (rotate, scale, translate)
- ✅ Create curves and complex paths
- ✅ Work with images and text
- ✅ Create animations
- ✅ Handle user input
- ✅ Use advanced techniques

Keep practicing and experimenting! 🎨

Next Steps

Projects to try:

1. Drawing app with different brushes
2. Bouncing balls with physics
3. Procedural pattern generator
4. Interactive data visualization
5. Generative art piece
6. Simple game (Pong, Snake)
7. Animation with tweening
8. Particle system

Challenge yourself! 💪

Thank You! 🎨

Questions?

Get Started:

```
pip install py5
```

Happy Coding!