Week 5 Practice Lecture Exercise Series

Constants

At the top of your code, define the following constants for width, height, radius, and window size. Using constants helps keep the code organized and easy to modify.

```
WIDTH = 100   # Width and height of each rectangle
HEIGHT = 100  # Height of each rectangle (same as width in this case for
squares)
RADIUS = 50   # Radius for circles
WIN_SIZE = 500  # Window size
```

Exercise 1: Draw a Rectangle on Screen

Question: Write a function called draw_single_rectangle that draws a single 100x100 rectangle in a 500x500 window at the position (0, 0).

Solution:

```
from graphix import Window, Rectangle, Point

WIDTH = 100   # Width and height of each rectangle
HEIGHT = 100  # Height of each rectangle (same as width in this case for
squares)
RADIUS = 50   # Radius for circles
WIN_SIZE = 500  # Window size


def draw_single_rectangle():
    win = Window("Rectangle", WIN_SIZE, WIN_SIZE)
    p1 = Point(0, 0)
    p2 = Point(WIDTH, HEIGHT)
    rect = Rectangle(p1, p2)
    rect.draw(win)
    win.get_mouse()  # Pause to view result
    win.close()
```

```
draw_single_rectangle()
```

Exercise 1a: Draw a Parameterized Rectangle

Question: Write a function called draw_rectangle that takes a window, two points, and a color as parameters. This function should draw a rectangle on the given window, with its top-left and bottom-right corners specified by the points. Create a main function to set up the window and call draw_rectangle with the points (0, 0) and (WIDTH, HEIGHT).

Solution:

```python
from graphix import Window, Rectangle, Point

WIDTH = 100   # Width and height of each rectangle
HEIGHT = 100  # Height of each rectangle (same as width in this case for squares)
RADIUS = 50   # Radius for circles
WIN_SIZE = 500  # Window size

# Function to draw a rectangle with specified parameters
def draw_rectangle(win, point1, point2, colour):
    rect = Rectangle(point1, point2)
    rect.fill_colour = colour
    rect.draw(win)

# Main function to set up the window and call draw_rectangle
def main():
    win = Window("Rectangle", WIN_SIZE, WIN_SIZE)
    p1 = Point(0, 0)
    p2 = Point(WIDTH, HEIGHT)
    colour = "blue"  # Specify color or ask user for input
    draw_rectangle(win, p1, p2, colour)
    win.get_mouse()  # Pause to view result
    win.close()

main()
```

In this version, draw_rectangle is a generalized function that takes parameters, making it more versatile for different drawing configurations. The main function manages the window setup and provides the necessary arguments to draw_rectangle.

Exercise 2: Add colour to the Rectangle

Question: Modify draw_single_rectangle to accept a colour as a parameter. Draw a single 100x100 rectangle at position (0, 0) with the colour provided by the user.

Solution:

```python
from graphix import Window, Rectangle, Point

WIDTH = 100   # Width and height of each rectangle
HEIGHT = 100  # Height of each rectangle (same as width in this case for squares)
RADIUS = 50   # Radius for circles
WIN_SIZE = 500  # Window size


def draw_single_rectangle(colour):
    win = Window("Rectangle", WIN_SIZE, WIN_SIZE)
    p1 = Point(0, 0)
    p2 = Point(WIDTH, HEIGHT)
    rect = Rectangle(p1, p2)
    rect.fill_colour = colour
    rect.draw(win)
    win.get_mouse()  # Pause to view result
    win.close()

draw_single_rectangle("blue")
```

Exercise 2a: Draw a Colored Rectangle Using Parameters

Question: Modify **Exercise 1a** to use the draw_rectangle function with parameters, allowing the user to specify the color of the rectangle. The function should draw a 100x100 rectangle in a 500x500 window at position (0, 0), with the color passed as an argument.

Solution:

```python
from graphix import Window, Rectangle, Point

WIDTH = 100   # Width and height of each rectangle
HEIGHT = 100  # Height of each rectangle (same as width in this case for squares)
RADIUS = 50   # Radius for circles
WIN_SIZE = 500  # Window size

# Function to draw a rectangle with specified parameters
def draw_rectangle(win, point1, point2, colour):
    rect = Rectangle(point1, point2)
    rect.fill_colour = colour
    rect.draw(win)

# Main function to set up the window, specify points, and call draw_rectangle with color input
def main():
    win = Window("Colored Rectangle", WIN_SIZE, WIN_SIZE)
    p1 = Point(0, 0)
    p2 = Point(WIDTH, HEIGHT)
    colour = input("Enter the color for the rectangle: ")  # Prompt user for color
    draw_rectangle(win, p1, p2, colour)
    win.get_mouse()  # Pause to view result
    win.close()

main()
```

In this exercise:

1. The draw_rectangle function is reused, so it can take any color and rectangle points, making the function versatile.
2. The main function prompts the user for a color, allowing dynamic

   color choice each time the program is run.
3. The window and rectangle setup remains consistent with the 100x100 size requirements and the window dimensions of 500x500.

Exercise 3: Draw Multiple Rectangles Across a Row

Question: Write a function draw_row_of_rectangles that fills the top row of the 500x500 window with 100x100 rectangles. Use a loop to draw five adjacent rectangles across the row.

Solution:

```
from graphix import Window, Rectangle, Point

WIDTH = 100   # Width and height of each rectangle
HEIGHT = 100  # Height of each rectangle (same as width in this case for
squares)
RADIUS = 50   # Radius for circles
WIN_SIZE = 500  # Window size

def draw_row_of_rectangles():
    win = Window("Rectangle Row", WIN_SIZE, WIN_SIZE)
    for i in range(5):
        p1 = Point(i * WIDTH, 0)
        p2 = Point((i + 1) * WIDTH, HEIGHT)
        rect = Rectangle(p1, p2)
        rect.draw(win)
    win.get_mouse()  # Pause to view result
    win.close()

draw_row_of_rectangles()
```

Exercise 3a: Draw a Row of Rectangles Using Parameters

Question: Write a function called draw_row_of_rectangles that fills the top row of the 500x500 window with five 100x100 rectangles. Each rectangle should be created using the draw_rectangle function, allowing for customizable colors for each rectangle.

Solution:

```python
from graphix import Window, Rectangle, Point

WIDTH = 100   # Width and height of each rectangle
HEIGHT = 100  # Height of each rectangle (same as width in this case for
squares)
RADIUS = 50   # Radius for circles
WIN_SIZE = 500  # Window size

# Function to draw a rectangle with specified parameters
def draw_rectangle(win, point1, point2, colour):
    rect = Rectangle(point1, point2)
    rect.fill_colour = colour
    rect.draw(win)

# Main function to set up the window and call draw_rectangle in a loop
def draw_row_of_rectangles():
    win = Window("Row of Rectangles", WIN_SIZE, WIN_SIZE)
    for i in range(5):
        p1 = Point(i * WIDTH, 0)
        p2 = Point((i + 1) * WIDTH, HEIGHT)
        colour = input(f"Enter the color for rectangle {i + 1}: ")  # Prompt
user for each rectangle's color
        draw_rectangle(win, p1, p2, colour)
    win.get_mouse()  # Pause to view result
    win.close()

draw_row_of_rectangles()
```

In this exercise:

1. draw_rectangle is used inside a loop, which iterates five times to draw a row of rectangles.
2. Each rectangle is positioned with p1 and p2 calculated based on i, ensuring they line up horizontally.
3. The colour variable is set for each rectangle based on user input, allowing each

rectangle in the row to have a unique color.

Exercise 4: Alternate colours in a Row of Rectangles

Question: Modify draw_row_of_rectangles to alternate colours between two user-provided colours for each rectangle in the row.

Solution:

```
from graphix import Window, Rectangle, Point

WIDTH = 100   # Width and height of each rectangle
HEIGHT = 100  # Height of each rectangle (same as width in this case for squares)
RADIUS = 50   # Radius for circles
WIN_SIZE = 500  # Window size


def draw_row_of_rectangles(colour1, colour2):
    win = Window("Rectangle Row", WIN_SIZE, WIN_SIZE)
    for i in range(5):
        p1 = Point(i * WIDTH, 0)
        p2 = Point((i + 1) * WIDTH, HEIGHT)
        rect = Rectangle(p1, p2)
        rect.fill_colour = colour1 if i % 2 == 0 else colour2
        rect.draw(win)
    win.get_mouse()  # Pause to view result
    win.close()

draw_row_of_rectangles("blue", "green")
```

Exercise 4a: Alternate Colors in a Row of Rectangles Using Parameters

Question: Write a `main()` function that asks for two colors from the user, then passes those colors to `draw_row_of_rectangles(colour1, colour2)`. The `draw_row_of_rectangles` function should use `draw_rectangle` to create a row of rectangles, alternating between the two colors provided.

Solution:

```
from graphix import Window, Rectangle, Point

WIDTH = 100   # Width and height of each rectangle
HEIGHT = 100  # Height of each rectangle (same as width in this case for
squares)
RADIUS = 50   # Radius for circles
WIN_SIZE = 500  # Window size

# Function to draw a rectangle with specified parameters
def draw_rectangle(win, point1, point2, colour):
    rect = Rectangle(point1, point2)
    rect.fill_colour = colour
    rect.draw(win)

# Function to draw a row of rectangles with alternating colors
def draw_row_of_rectangles(colour1, colour2):
    win = Window("Rectangle Row", WIN_SIZE, WIN_SIZE)
    for i in range(5):
        p1 = Point(i * WIDTH, 0)
        p2 = Point((i + 1) * WIDTH, HEIGHT)
        colour = colour1 if i % 2 == 0 else colour2  # Alternate colors
        draw_rectangle(win, p1, p2, colour)
    win.get_mouse()  # Pause to view result
    win.close()

# Main function to prompt for colors and draw row with alternating colors
def main():
    colour1 = input("Enter the first color: ")
    colour2 = input("Enter the second color: ")
    draw_row_of_rectangles(colour1, colour2)

main()
```

In this exercise:

1. The `main` function prompts the user for two colors, `colour1` and `colour2`.
2. The `draw_row_of_rectangles` function uses a loop to draw five rectangles across the row, alternating colors by checking if `i` is even or odd.
3. The `draw_rectangle` function is reused to draw each rectangle with the alternating colors, maintaining modularity and readability in the code.

Exercise 4b: Alternate colours Using a Boolean Flag

Question: Write a function draw_row_of_rectangles_with_flag that fills the top row of the 500x500 window with 100x100 rectangles. Use a boolean flag (colour_flag) to alternate colours between two user-provided colours for each rectangle in the row.

Solution:

```python
from graphix import Window, Rectangle, Point

WIDTH = 100   # Width and height of each rectangle
HEIGHT = 100  # Height of each rectangle (same as width in this case for
squares)
RADIUS = 50   # Radius for circles
WIN_SIZE = 500  # Window size

def draw_row_of_rectangles_with_flag(colour1, colour2):
    win = Window("Rectangle Row with Flag", WIN_SIZE, WIN_SIZE)
    colour_flag = True  # Initialize the flag to start with colour1
    for i in range(5):
        p1 = Point(i * WIDTH, 0)
        p2 = Point((i + 1) * WIDTH, HEIGHT)
        rect = Rectangle(p1, p2)
        if colour_flag:
            rect.fill_colour = colour1
        else:
            rect.fill_colour = colour2
        rect.draw(win)
        colour_flag = not colour_flag  # Toggle the flag to switch colours
    win.get_mouse()  # Pause to view result
    win.close()

draw_row_of_rectangles_with_flag("blue", "green")
```

Exercise 4b.1: Alternate Colors in a Row Using a Boolean Flag

Question: Write a `main()` function that asks for two colors, then calls `draw_row_of_rectangles_with_flag(colour1, colour2)`. This function should fill the top row with alternating colors by toggling a boolean flag.

Solution:

```python
from graphix import Window, Rectangle, Point

WIDTH = 100   # Width and height of each rectangle
HEIGHT = 100  # Height of each rectangle (same as width in this case for
squares)
RADIUS = 50   # Radius for circles
WIN_SIZE = 500  # Window size

# Function to draw a rectangle with specified parameters
def draw_rectangle(win, point1, point2, colour):
    rect = Rectangle(point1, point2)
    rect.fill_colour = colour
    rect.draw(win)

# Function to draw a row of rectangles with alternating colors using a flag
def draw_row_of_rectangles_with_flag(colour1, colour2):
    win = Window("Rectangle Row with Flag", WIN_SIZE, WIN_SIZE)
    colour_flag = True  # Initialize the flag to start with colour1
    for i in range(5):
        p1 = Point(i * WIDTH, 0)
        p2 = Point((i + 1) * WIDTH, HEIGHT)
        # Use the flag to determine color
        colour = colour1 if colour_flag else colour2
        draw_rectangle(win, p1, p2, colour)
        colour_flag = not colour_flag  # Toggle the flag to switch colors
    win.get_mouse()  # Pause to view result
    win.close()

# Main function to prompt for colors and call the function with the flag
def main():
    colour1 = input("Enter the first color: ")
    colour2 = input("Enter the second color: ")
    draw_row_of_rectangles_with_flag(colour1, colour2)

main()
```

In this exercise:

1. The `main` function prompts the user to input two colors.
2. The `draw_row_of_rectangles_with_flag` function uses a `colour_flag` boolean to alternate colors, toggling it after each rectangle is drawn.
3. `draw_rectangle` is reused to handle the drawing of each rectangle, while `colour_flag` manages the color switching, creating a neat and modular approach.

Exercise 4c: Alternate colours Using a List of colours

Question: Write a function `draw_row_of_rectangles_with_list` that asks the user for two colours, places them in a list, and uses them to alternate colours in the row.

Solution:

```python
from graphix import Window, Rectangle, Point

WIDTH = 100   # Width and height of each rectangle
HEIGHT = 100  # Height of each rectangle (same as width in this case for squares)
RADIUS = 50   # Radius for circles
WIN_SIZE = 500  # Window size

def draw_row_of_rectangles_with_list():
    win = Window("Rectangle Row with colour List", WIN_SIZE, WIN_SIZE)

    # Get colours from the user and store in a list
    colour1 = input("Enter the first colour: ")
    colour2 = input("Enter the second colour: ")
    colours = [colour1, colour2]

    for i in range(5):
        p1 = Point(i * WIDTH, 0)
        p2 = Point((i + 1) * WIDTH, HEIGHT)
        rect = Rectangle(p1, p2)
        rect.fill_colour = colours[i % 2]  # Alternate colours using the list
        rect.draw(win)
    win.get_mouse()  # Pause to view result
    win.close()

draw_row_of_rectangles_with_list()
```

Exercise 4c.1: Alternate Colors in a Row Using a List of Colors

Question: Write a `main()` function that asks for two colors, places them in a list, and passes this list to `draw_row_of_rectangles_with_list`. This function should use the list to alternate colors for each rectangle in the row.

Solution:

```
from graphix import Window, Rectangle, Point

WIDTH = 100   # Width and height of each rectangle
HEIGHT = 100  # Height of each rectangle (same as width in this case for
squares)
RADIUS = 50   # Radius for circles
WIN_SIZE = 500  # Window size

# Function to draw a rectangle with specified parameters
def draw_rectangle(win, point1, point2, colour):
    rect = Rectangle(point1, point2)
    rect.fill_colour = colour
    rect.draw(win)

# Function to draw a row of rectangles with alternating colors using a list
def draw_row_of_rectangles_with_list(colours):
    win = Window("Rectangle Row with Colour List", WIN_SIZE, WIN_SIZE)
    for i in range(5):
        p1 = Point(i * WIDTH, 0)
        p2 = Point((i + 1) * WIDTH, HEIGHT)
        # Use the list to alternate colors
        colour = colours[i % 2]
        draw_rectangle(win, p1, p2, colour)
    win.get_mouse()  # Pause to view result
    win.close()

# Main function to prompt for colors and call the function with the list
def main():
    colour1 = input("Enter the first colour: ")
    colour2 = input("Enter the second colour: ")
    colours = [colour1, colour2]  # Store colors in a list
    draw_row_of_rectangles_with_list(colours)

main()
```

In this exercise:

1. The `main` function collects two color inputs from the user and stores them in a list called `colours`.
2. The `draw_row_of_rectangles_with_list` function uses this list to alternate colors by accessing `colours[i % 2]` within the loop.
3. `draw_rectangle` is called to handle each rectangle's drawing and coloring, leveraging the list for easy alternation between colors.

Exercise 5: Tile Rows of Rectangles to Fill the Window

Question: Create a function `draw_tiled_rectangles` that fills the entire 500x500 window with 100x100 rectangles. Use nested loops to draw each row, filling the window row by row.

Solution:

```
from graphix import Window, Rectangle, Point

WIDTH = 100   # Width and height of each rectangle
HEIGHT = 100  # Height of each rectangle (same as width in this case for
squares)
RADIUS = 50   # Radius for circles
WIN_SIZE = 500  # Window size


def draw_tiled_rectangles():
    win = Window("Tiled Rectangles", WIN_SIZE, WIN_SIZE)
    for row in range(5):
        for col in range(5):
            p1 = Point(col * WIDTH, row * HEIGHT)
            p2 = Point((col + 1) * WIDTH, (row + 1) * HEIGHT)
            rect = Rectangle(p1, p2)
            rect.draw(win)
    win.get_mouse()  # Pause to view result
    win.close()

draw_tiled_rectangles()
```

Exercise 5a: Tile Rows of Rectangles to Fill the Window Using Parameters

Question: Write a `main()` function that calls `draw_tiled_rectangles` to fill the entire 500x500 window with 100x100 rectangles. Use `draw_rectangle` within `draw_tiled_rectangles` to create each rectangle.

Solution:

```
from graphix import Window, Rectangle, Point

WIDTH = 100   # Width and height of each rectangle
HEIGHT = 100  # Height of each rectangle (same as width in this case for
squares)
RADIUS = 50   # Radius for circles
WIN_SIZE = 500  # Window size

# Function to draw a rectangle with specified parameters
def draw_rectangle(win, point1, point2, colour):
    rect = Rectangle(point1, point2)
    rect.fill_colour = colour
    rect.draw(win)

# Function to tile rectangles across the window
def draw_tiled_rectangles(win, colour):
    for row in range(5):
        for col in range(5):
            p1 = Point(col * WIDTH, row * HEIGHT)
            p2 = Point((col + 1) * WIDTH, (row + 1) * HEIGHT)
            draw_rectangle(win, p1, p2, colour)

# Main function to set up the window and call draw_tiled_rectangles
def main():
    win = Window("Tiled Rectangles", WIN_SIZE, WIN_SIZE)
    colour = input("Enter the color for the rectangles: ")  # Prompt user for
color
    draw_tiled_rectangles(win, colour)
    win.get_mouse()  # Pause to view result
    win.close()

main()
```

In this exercise:

1. `draw_rectangle` is reused within `draw_tiled_rectangles` to create each 100x100 rectangle, allowing for modularity and easier customization of individual rectangles.
2. `draw_tiled_rectangles` uses nested loops to position the rectangles row by row across the 500x500 window.
3. The `main` function prompts the user for a color and passes it to `draw_tiled_rectangles`, filling the entire window with rectangles of the specified color.

Exercise 6: Alternate colours for the Entire Tiled Window

Question: Modify `draw_tiled_rectangles` to alternate two colours across the entire window, creating a checkerboard pattern.

Solution:

```python
from graphix import Window, Rectangle, Point

WIDTH = 100   # Width and height of each rectangle
HEIGHT = 100  # Height of each rectangle (same as width in this case for squares)
RADIUS = 50   # Radius for circles
WIN_SIZE = 500  # Window size

def draw_tiled_rectangles(colour1, colour2):
    win = Window("Checkerboard", WIN_SIZE, WIN_SIZE)
    for row in range(5):
        for col in range(5):
            p1 = Point(col * WIDTH, row * HEIGHT)
            p2 = Point((col + 1) * WIDTH, (row + 1) * HEIGHT)
            rect = Rectangle(p1, p2)
            rect.fill_colour = colour1 if (row + col) % 2 == 0 else colour2
            rect.draw(win)
    win.get_mouse()  # Pause to view result
    win.close()

draw_tiled_rectangles("red", "black")
```

Exercise 6a: Checkerboard Pattern Across the Tiled Window Using Parameters

Question: Write a `main()` function that asks the user for two colors and calls `draw_tiled_rectangles(colour1, colour2)` to fill the window in a checkerboard pattern. Use `draw_rectangle` to draw each tile, alternating colors for each rectangle.

Solution:

```python
from graphix import Window, Rectangle, Point


WIDTH = 100   # Width and height of each rectangle
HEIGHT = 100  # Height of each rectangle (same as width in this case for squares)
RADIUS = 50   # Radius for circles
WIN_SIZE = 500  # Window size

# Function to draw a rectangle with specified parameters
def draw_rectangle(win, point1, point2, colour):
    rect = Rectangle(point1, point2)
    rect.fill_colour = colour
    rect.draw(win)

# Function to tile rectangles in a checkerboard pattern
def draw_tiled_rectangles(win, colour1, colour2):
    for row in range(5):
        for col in range(5):
            p1 = Point(col * WIDTH, row * HEIGHT)
            p2 = Point((col + 1) * WIDTH, (row + 1) * HEIGHT)
            # Alternate colors based on position
            colour = colour1 if (row + col) % 2 == 0 else colour2
            draw_rectangle(win, p1, p2, colour)

# Main function to set up the window and call draw_tiled_rectangles
def main():
    colour1 = input("Enter the first colour: ")
    colour2 = input("Enter the second colour: ")
    win = Window("Checkerboard", WIN_SIZE, WIN_SIZE)
    draw_tiled_rectangles(win, colour1, colour2)
    win.get_mouse()  # Pause to view result
    win.close()

main()
```

In this exercise:

1. `main()` prompts the user for two colors, `colour1` and `colour2`, to use in the checkerboard pattern.
2. `draw_tiled_rectangles` calculates the color for each tile using the `(row + col) % 2 == 0` condition to alternate colors, creating a checkerboard effect.
3. The `draw_rectangle` function is used to handle each rectangle's drawing and color assignment, keeping the code modular and clean.

Exercise 7: Draw a Circle in a Specified Position

Question: Write a function `draw_circle` that takes parameters for the center point, radius, and colour of a circle. Draw a single circle in the center of the 500x500 window.

Solution:

```python
from graphix import Circle

WIDTH = 100   # Width and height of each rectangle
HEIGHT = 100  # Height of each rectangle (same as width in this case for squares)
RADIUS = 50   # Radius for circles
WIN_SIZE = 500  # Window size


def draw_circle(center_x, center_y, radius, colour):
    win = Window("Circle", WIN_SIZE, WIN_SIZE)
    center = Point(center_x, center_y)
    circle = Circle(center, radius)
    circle.fill_colour = colour
    circle.draw(win)
    win.get_mouse()  # Pause to view result
    win.close()

draw_circle(WIN_SIZE // 2, WIN_SIZE // 2, RADIUS, "blue")
```

Exercise 7a: Draw a Circle Using Parameters and Main Function

Question: Write a `draw_circle` function that takes a window, center point, radius, and color as parameters. The function should draw a circle with these specifications on the given window. Create a `main()` function to set up the window, define the center point, and call `draw_circle` to draw a circle in the center of a 500x500 window.

Solution:

```python
from graphix import Window, Circle, Point

# Constants
WIDTH = 100       # Width and height of each rectangle
HEIGHT = 100      # Height of each rectangle (same as width in this case for squares)
RADIUS = 50       # Radius for circles
WIN_SIZE = 500    # Window size

# Function to draw a circle with specified parameters
def draw_circle(win, center, radius, colour):
    circle = Circle(center, radius)
    circle.fill_colour = colour
    circle.draw(win)

# Main function to set up the window and call draw_circle
def main():
    win = Window("Circle", WIN_SIZE, WIN_SIZE)
    center_x = WIN_SIZE // 2
    center_y = WIN_SIZE // 2
    center = Point(center_x, center_y)
    radius = RADIUS
    colour = input("Enter the color for the circle: ")  # Prompt user for color
    draw_circle(win, center, radius, colour)
    win.get_mouse()  # Pause to view result
    win.close()

main()
```

Explanation:

- **Imports and Constants:**
    - We import the necessary classes from the `graphix` module: `Window`, `Circle`, and `Point`.
    - Constants like `WIDTH`, `HEIGHT`, `RADIUS`, and `WIN_SIZE` are defined for consistency.
- **`draw_circle` Function:**
    - Takes `win`, `center`, `radius`, and `colour` as parameters.
    - Creates a `Circle` object with the provided center and radius.
    - Sets the `fill_colour` of the circle.
    - Draws the circle on the provided window.
- **`main` Function:**
    - Sets up the window with the specified `WIN_SIZE`.
    - Calculates the center point of the window.
    - Prompts the user to enter a color for the circle.
    - Calls `draw_circle`, passing the window, center point, radius, and color.
    - Waits for a mouse click (`win.get_mouse()`) before closing the window.

This structure separates the drawing logic from the window setup and user input, making the code modular and easier to maintain.

Exercise 8: Fill the Window with Circles

Question: Create a function draw_tiled_circles that tiles the 500x500 window with circles of radius 50 (so they fit in a 100x100 square each).

Solution:

```
from graphix import Window, Circle, Point

# Constants
WIDTH = 100         # Width and height of each rectangle
HEIGHT = 100        # Height of each rectangle (same as width in this case for
squares)
RADIUS = 50         # Radius for circles
WIN_SIZE = 500      # Window size



def draw_tiled_circles():
    win = Window("Tiled Circles", WIN_SIZE, WIN_SIZE)
    for row in range(5):
        for col in range(5):
            center = Point(col * WIDTH + RADIUS, row * HEIGHT + RADIUS)
            circle = Circle(center, RADIUS)
            circle.draw(win)
    win.get_mouse()  # Pause to view result
    win.close()

draw_tiled_circles()
```

Exercise 8a: Tile the Window with Circles Using Parameters

Question: Write a `draw_circle` function that takes a window, center point, radius, and color as parameters. Then, create a `draw_tiled_circles` function that tiles the window with circles of radius 50 (arranged in a 100x100 grid pattern). Use `main()` to set up the window and call `draw_tiled_circles`.

Solution:

```
from graphix import Window, Circle, Point

# Constants
WIDTH = 100        # Width and height of each rectangle
HEIGHT = 100       # Height of each rectangle
RADIUS = 50        # Radius for circles
WIN_SIZE = 500     # Window size

# Function to draw a circle with specified parameters
def draw_circle(win, center, radius, colour):
    circle = Circle(center, radius)
    circle.fill_colour = colour
    circle.draw(win)

# Function to tile circles across the window
def draw_tiled_circles(win, colour):
    for row in range(5):
        for col in range(5):
            center = Point(col * WIDTH + RADIUS, row * HEIGHT + RADIUS)
            draw_circle(win, center, RADIUS, colour)

# Main function to set up the window and call draw_tiled_circles
def main():
    win = Window("Tiled Circles", WIN_SIZE, WIN_SIZE)
    colour = input("Enter the color for the circles: ")  # Prompt user for
color
    draw_tiled_circles(win, colour)
    win.get_mouse()  # Pause to view result
    win.close()

main()
```

Explanation:

- **`draw_circle` Function:**
  - Takes the `win`, `center`, `radius`, and `colour` as parameters.
  - Creates a `Circle` object with the specified center and radius.
  - Sets the circle’s `fill_colour`.
  - Draws the circle on the specified window.
- **`draw_tiled_circles` Function:**
  - Loops over rows and columns to position circles in a 5x5 grid.
  - Calculates the center of each circle based on `col * WIDTH + RADIUS` and `row * HEIGHT + RADIUS`, ensuring they are spaced within a 100x100 square.
  - Calls `draw_circle` with each calculated center and the user-specified color.
- **`main` Function:**
  - Sets up a `500x500` window.
  - Prompts the user to enter a color, which is passed to `draw_tiled_circles`.
  - Waits for a mouse click before closing the window.

This modular approach keeps the circle drawing separate from the tiling logic, making the code cleaner and more reusable.

Exercise 9: Alternate colours for Tiled Circles

Question: Modify `draw_tiled_circles` to alternate between two user-provided colours across the window to create a checkerboard pattern.

Solution:

```python
from graphix import Window, Circle, Point

# Constants
WIDTH = 100        # Width and height of each rectangle
HEIGHT = 100       # Height of each rectangle (same as width in this case for
squares)
RADIUS = 50        # Radius for circles
WIN_SIZE = 500     # Window size



def draw_tiled_circles(colour1, colour2):
    win = Window("Checkerboard Circles", WIN_SIZE, WIN_SIZE)
    for row in range(5):
        for col in range(5):
            center = Point(col * WIDTH + RADIUS, row * HEIGHT + RADIUS)
            circle = Circle(center, RADIUS)
            circle.fill_colour = colour1 if (row + col) % 2 == 0 else colour2
            circle.draw(win)
    win.get_mouse()  # Pause to view result
    win.close()

draw_tiled_circles("yellow", "purple")
```

Exercise 9a: Checkerboard Pattern with Tiled Circles Using Parameters

Question: Write a `draw_circle` function that takes a window, center point, radius, and color as parameters. Then, create a `draw_tiled_circles` function that tiles the window with circles, alternating between two user-provided colors in a checkerboard pattern. Use `main()` to set up the window and call `draw_tiled_circles`.

Solution:

```python
from graphix import Window, Circle, Point

# Constants
WIDTH = 100        # Width and height of each rectangle
HEIGHT = 100       # Height of each rectangle
RADIUS = 50        # Radius for circles
WIN_SIZE = 500     # Window size

# Function to draw a circle with specified parameters
def draw_circle(win, center, radius, colour):
    circle = Circle(center, radius)
    circle.fill_colour = colour
    circle.draw(win)

# Function to tile circles in a checkerboard pattern
def draw_tiled_circles(win, colour1, colour2):
    for row in range(5):
        for col in range(5):
            center = Point(col * WIDTH + RADIUS, row * HEIGHT + RADIUS)
            # Alternate colors based on position
            colour = colour1 if (row + col) % 2 == 0 else colour2
            draw_circle(win, center, RADIUS, colour)

# Main function to set up the window and call draw_tiled_circles
def main():
    colour1 = input("Enter the first colour: ")
    colour2 = input("Enter the second colour: ")
    win = Window("Checkerboard Circles", WIN_SIZE, WIN_SIZE)
    draw_tiled_circles(win, colour1, colour2)
    win.get_mouse()  # Pause to view result
    win.close()

main()
```

Explanation:

- **`draw_circle` Function:**
    - Takes `win`, `center`, `radius`, and `colour` as parameters.
    - Creates a `Circle` object with the specified center and radius.
    - Sets the circle's `fill_colour` to the specified color.
    - Draws the circle on the provided window.
- **`draw_tiled_circles` Function:**
    - Loops over rows and columns to create a 5x5 grid.
    - Calculates the center of each circle so that it fits within a 100x100 square.
    - Alternates colors using `(row + col) % 2 == 0` to create the checkerboard effect, selecting `colour1` or `colour2`.
    - Calls `draw_circle` for each circle with the calculated center and alternating color.
- **`main` Function:**
    - Prompts the user to input two colors for the checkerboard pattern.
    - Sets up a `500x500` window and passes the colors to `draw_tiled_circles`.
    - Waits for a mouse click before closing the window.

This approach keeps the checkerboard logic in `draw_tiled_circles` while making the `draw_circle` function reusable for other purposes.

Exercise 10: Alternate Rectangles and Circles Across the Window

Question: Write a function `draw_alternating_shapes` that fills the window with alternating 100x100 rectangles and circles.

Solution:

```python
from graphix import Window, Circle, Point

# Constants
WIDTH = 100       # Width and height of each rectangle
HEIGHT = 100      # Height of each rectangle (same as width in this case for
squares)
RADIUS = 50       # Radius for circles
WIN_SIZE = 500    # Window size


def draw_alternating_shapes(colour1, colour2):
    win = Window("Alternating Shapes", WIN_SIZE, WIN_SIZE)
    for row in range(5):
        for col in range(5):
            p1 = Point(col * WIDTH, row * HEIGHT)
            p2 = Point((col + 1) * WIDTH, (row + 1) * HEIGHT)
            if (row + col) % 2 == 0:
                shape = Rectangle(p1, p2)
                shape.fill_colour = colour1
            else:
                center = Point(col * WIDTH + RADIUS, row * HEIGHT + RADIUS)
                shape = Circle(center, RADIUS)
                shape.fill_colour = colour2
            shape.draw(win)
    win.get_mouse()  # Pause to view result
    win.close()

draw_alternating_shapes("blue", "orange")
```

Exercise 10a: Alternate Rectangles and Circles Across the Window Using Parameters

Question: Write a draw_rectangle and draw_circle function to handle individual shapes with specified parameters. Then, create draw_alternating_shapes, which fills the window with alternating rectangles and circles, each 100x100 in size. Use main() to set up the window and call draw_alternating_shapes with two user-provided colors.

Solution:

```python
from graphix import Window, Rectangle, Circle, Point

# Constants
WIDTH = 100        # Width and height of each rectangle
HEIGHT = 100       # Height of each rectangle
RADIUS = 50        # Radius for circles
WIN_SIZE = 500     # Window size

# Function to draw a rectangle with specified parameters
def draw_rectangle(win, point1, point2, colour):
    rect = Rectangle(point1, point2)
    rect.fill_colour = colour
    rect.draw(win)

# Function to draw a circle with specified parameters
def draw_circle(win, center, radius, colour):
    circle = Circle(center, radius)
    circle.fill_colour = colour
    circle.draw(win)

# Function to alternate rectangles and circles across the window
def draw_alternating_shapes(win, colour1, colour2):
    for row in range(5):
        for col in range(5):
            if (row + col) % 2 == 0:
                # Draw rectangle
                p1 = Point(col * WIDTH, row * HEIGHT)
                p2 = Point((col + 1) * WIDTH, (row + 1) * HEIGHT)
                draw_rectangle(win, p1, p2, colour1)
            else:
                # Draw circle
                center = Point(col * WIDTH + RADIUS, row * HEIGHT + RADIUS)
                draw_circle(win, center, RADIUS, colour2)

# Main function to set up the window and call draw_alternating_shapes
def main():
    colour1 = input("Enter the color for rectangles: ")
    colour2 = input("Enter the color for circles: ")
    win = Window("Alternating Shapes", WIN_SIZE, WIN_SIZE)
    draw_alternating_shapes(win, colour1, colour2)
    win.get_mouse()  # Pause to view result
    win.close()

main()
```

Explanation:

- **`draw_rectangle` Function:**
    - Takes `win`, `point1`, `point2`, and `colour` as parameters.
    - Creates a `Rectangle` with the specified points and sets the `fill_colour`.
    - Draws the rectangle on the window.
- **`draw_circle` Function:**
    - Takes `win`, `center`, `radius`, and `colour` as parameters.
    - Creates a `Circle` with the specified center and radius and sets the `fill_colour`.
    - Draws the circle on the window.
- **`draw_alternating_shapes` Function:**
    - Loops through a 5x5 grid to position shapes.
    - Uses `(row + col) % 2 == 0` to alternate between rectangles and circles.
    - Calls `draw_rectangle` for even positions and `draw_circle` for odd positions, assigning `colour1` to rectangles and `colour2` to circles.
- **`main` Function:**
    - Prompts the user to enter two colors, one for rectangles and one for circles.
    - Initializes a `500x500` window and calls `draw_alternating_shapes` with the chosen colors.
    - Waits for a mouse click before closing the window.

This structure keeps the code modular, with separate functions for drawing each shape, making it easy to maintain and modify.

Test Script

```
from graphix import Window, Rectangle, Circle, Point

# Constants
WIN_SIZE = 500
WIDTH = 100
HEIGHT = 100
RADIUS = 50

# Function to draw a rectangle with specified parameters
def draw_rectangle(win, point1, point2, colour):
    rect = Rectangle(point1, point2)
    rect.fill_colour = colour
    rect.draw(win)

# Function to draw a circle with specified parameters
def draw_circle(win, center, radius, colour):
    circle = Circle(center, radius)
    circle.fill_colour = colour
    circle.draw(win)

# Exercise examples

def exercise_1a():
    win = Window("Exercise 1a", WIN_SIZE, WIN_SIZE)
    p1 = Point(0, 0)
    p2 = Point(WIDTH, HEIGHT)
    draw_rectangle(win, p1, p2, "blue")
    win.get_mouse()
    win.close()

def exercise_2a():
    win = Window("Exercise 2a", WIN_SIZE, WIN_SIZE)
    p1 = Point(0, 0)
    p2 = Point(WIDTH, HEIGHT)
    colour = input("Enter the color for the rectangle: ")
    draw_rectangle(win, p1, p2, colour)
    win.get_mouse()
    win.close()

def exercise_3a():
    win = Window("Exercise 3a", WIN_SIZE, WIN_SIZE)
    for i in range(5):
        p1 = Point(i * WIDTH, 0)
```

```python
        p2 = Point((i + 1) * WIDTH, HEIGHT)
        colour = input(f"Enter color for rectangle {i + 1}: ")
        draw_rectangle(win, p1, p2, colour)
    win.get_mouse()
    win.close()

def exercise_4a():
    win = Window("Exercise 4a", WIN_SIZE, WIN_SIZE)
    colour1 = input("Enter the first color: ")
    colour2 = input("Enter the second color: ")
    for i in range(5):
        p1 = Point(i * WIDTH, 0)
        p2 = Point((i + 1) * WIDTH, HEIGHT)
        colour = colour1 if i % 2 == 0 else colour2
        draw_rectangle(win, p1, p2, colour)
    win.get_mouse()
    win.close()

def exercise_5a():
    win = Window("Exercise 5a", WIN_SIZE, WIN_SIZE)
    colour = input("Enter the color for the rectangles: ")
    for row in range(5):
        for col in range(5):
            p1 = Point(col * WIDTH, row * HEIGHT)
            p2 = Point((col + 1) * WIDTH, (row + 1) * HEIGHT)
            draw_rectangle(win, p1, p2, colour)
    win.get_mouse()
    win.close()

def exercise_6a():
    win = Window("Exercise 6a", WIN_SIZE, WIN_SIZE)
    colour1 = input("Enter the first color: ")
    colour2 = input("Enter the second color: ")
    for row in range(5):
        for col in range(5):
            p1 = Point(col * WIDTH, row * HEIGHT)
            p2 = Point((col + 1) * WIDTH, (row + 1) * HEIGHT)
            colour = colour1 if (row + col) % 2 == 0 else colour2
            draw_rectangle(win, p1, p2, colour)
    win.get_mouse()
    win.close()

def exercise_7a():
    win = Window("Exercise 7a", WIN_SIZE, WIN_SIZE)
    center = Point(WIN_SIZE // 2, WIN_SIZE // 2)
    colour = input("Enter the color for the circle: ")
    draw_circle(win, center, RADIUS, colour)
    win.get_mouse()
    win.close()

def exercise_8a():
    win = Window("Exercise 8a", WIN_SIZE, WIN_SIZE)
    colour = input("Enter the color for the circles: ")
    for row in range(5):
        for col in range(5):
            center = Point(col * WIDTH + RADIUS, row * HEIGHT + RADIUS)
            draw_circle(win, center, RADIUS, colour)
    win.get_mouse()
    win.close()

def exercise_9a():
    win = Window("Exercise 9a", WIN_SIZE, WIN_SIZE)
    colour1 = input("Enter the first color: ")
    colour2 = input("Enter the second color: ")
    for row in range(5):
        for col in range(5):
            center = Point(col * WIDTH + RADIUS, row * HEIGHT + RADIUS)
            colour = colour1 if (row + col) % 2 == 0 else colour2
            draw_circle(win, center, RADIUS, colour)
    win.get_mouse()
    win.close()

def exercise_10a():
    win = Window("Exercise 10a", WIN_SIZE, WIN_SIZE)
    colour1 = input("Enter the color for rectangles: ")
    colour2 = input("Enter the color for circles: ")
    for row in range(5):
        for col in range(5):
            if (row + col) % 2 == 0:
                p1 = Point(col * WIDTH, row * HEIGHT)
                p2 = Point((col + 1) * WIDTH, (row + 1) * HEIGHT)
                draw_rectangle(win, p1, p2, colour1)
            else:
                center = Point(col * WIDTH + RADIUS, row * HEIGHT + RADIUS)
                draw_circle(win, center, RADIUS, colour2)
    win.get_mouse()
    win.close()

# Console menu to select exercises
def menu():
    exercises = {
        "1": exercise_1a,
        "2": exercise_2a,
        "3": exercise_3a,
        "4": exercise_4a,
        "5": exercise_5a,
        "6": exercise_6a,
        "7": exercise_7a,
        "8": exercise_8a,
        "9": exercise_9a,
        "10": exercise_10a
    }

    while True:
        print("\nSelect an exercise to run:")
        for num in exercises:
            print(f"Exercise {num}a")
        choice = input("Enter the exercise number (1-10) or 'q' to quit: ")

        if choice in exercises:
            exercises[choice]()
        elif choice.lower() == 'q':
            print("Exiting.")
            break
        else:
            print("Invalid selection. Please try again.")

# Run the menu
menu()
```