

Course Name	ITD 2313 – Script Programming
Instructor	Andy Tripp
Student Name	Timothy Obinda
Due date	11/02/2025
Grade	Put grade earned here
Grading Comments	Put instructor comments here

## INSTRUCTIONS FOR THE EXERCISE

You should always read the instructions in full. It is best to do a full read through before starting the assignment. Each screenshot needs to be appropriately labeled.

In this instruction, you are given information about a specific project in the text. In some cases, specific instructions for a particular project will be given. There also may be specific test data given. All specific instructions should be followed. When test data is given, screen shots must include that test data, and those results will need to be grabbed via screen shot. All screen shots will go into your submission document.

Some advice, copy this instruction set into your submission document and then put the screen shots under each numbered task. This will take care of making the appropriate labels for you. Each individual book page in the instructions should be in a different screen shot. For any single book page, you may have all the numbered tasks on that page to be in a single screen shot.

**Page 218-219**

**Project # 1**

```
*Circle.py - C:\Users\student\Desktop\Circle.py (3.13.7)*
File Edit Format Run Options Window Help
import math
import turtle

def drawCircle(t, centerX, centerY, radius):
    """
    Draws a circle using a Turtle object.

    Parameters:
        t (turtle.Turtle): The Turtle object used for drawing
        centerX (float): x-coordinate of the circle's center
        centerY (float): y-coordinate of the circle's center
        radius (float): radius of the circle
    """

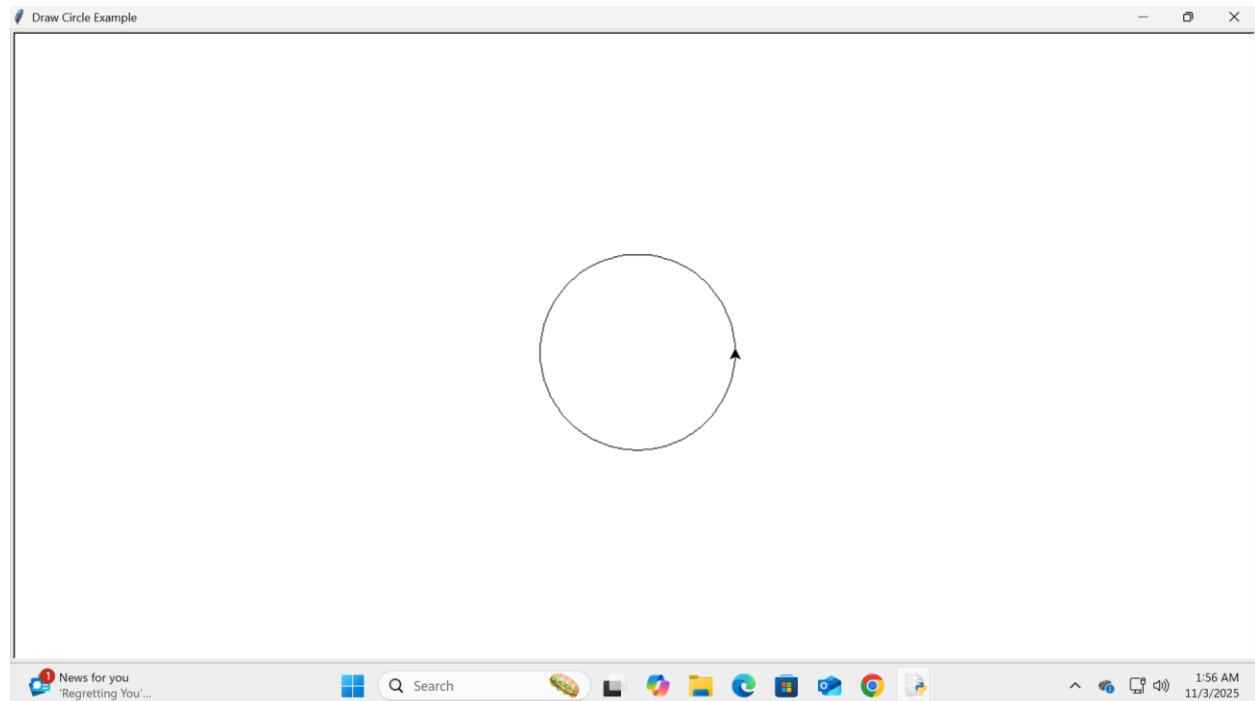
    # Move turtle to the starting point (rightmost point of the circle)
    t.penup()
    t.goto(centerX + radius, centerY)
    t.setheading(90) # Point upward
    t.pendown()

    # Calculate step size for 120 segments (each 3")
    step_length = 2.0 * math.pi * radius / 120.0

    # Draw the circle
    for _ in range(120):
        t.left(3)
        t.forward(step_length)

    # Example test (can be removed when imported)
if __name__ == "__main__":
    wn = turtle.Screen()
    wn.title("Draw Circle Example")
    t = turtle.Turtle()
    t.speed(0)
    t.circle(100)

Ln: 16 Col: 4
2:04 AM 11/3/2025
```



**Special Instructions:**

1. This project is to be done in a Python program file. The file needs to be submitted as part of the submission for this assignment. Place the program file into the zip file that is submitted for this assignment.

**Project # 3**

**Special Instructions:**

1. This project is to be done in a Python program file. The file needs to be submitted as part of the submission for this assignment. Place the program file into the zip file that is submitted for this assignment.

The screenshot shows a Windows desktop environment. In the foreground, there is a window titled "koch.py - C:/Users/student/Desktop/koch.py (3.13.7)". The code inside the window is a Python script for drawing a Koch fractal snowflake. The script uses the turtle module to draw three sides of an equilateral triangle. It then recursively divides each side into four segments and adds the middle segment, creating a fractal pattern. The script ends with a check for the main module name.

```

# koch.py - C:/Users/student/Desktop/koch.py (3.13.7)
File Edit Format Run Options Window Help
    drawFractalLine(first, direction, level)
    drawFractalLine(second, direction, level)
    drawFractalLine(third, direction, level - 1)

def main():
    # Requested parameters
    width = 200
    height = 200
    size = 150
    level = 4

    screen = turtle.Screen()
    screen.setup(width, height)
    screen.title(f"Koch Snowflake - Level {level}")

    turtle.hideturtle()
    turtle.speed(0)
    turtle.pensize(1)
    turtle.penup()

    # Centering: start at lower-left vertex of an equilateral triangle of side size
    tri_height = math.sqrt(3) / 2 * size
    start_x = -size / 2
    start_y = -tri_height / 3
    turtle.goto(start_x, start_y)
    turtle.pendown()

    # Draw the three Koch sides (absolute headings 0, -120, 120)
    drawFractalLine(size, 0, level)
    drawFractalLine(size, -120, level)
    drawFractalLine(size, 120, level)

    turtle.penup()
    turtle.hideturtle()
    turtle.done()

if __name__ == "__main__":
    main()

```

The taskbar at the bottom of the screen shows various pinned icons, including a weather widget showing 48°F, a search bar, and several application icons like File Explorer, Edge, and Google Chrome. The system tray indicates the date and time as 11/3/2025, 1:10 AM.

The screenshot shows a Windows desktop environment. In the foreground, there is a window titled "IDLE Shell 3.13.7". The shell window displays the Python version and a restart message. Below the shell window, there is a separate window titled "Koc..." which displays a Koch fractal snowflake. The fractal is a complex, self-similar shape formed by iterative division of a triangle's sides. The taskbar at the bottom of the screen shows various pinned icons, including a weather widget showing 48°F, a search bar, and several application icons like File Explorer, Edge, and Google Chrome. The system tray indicates the date and time as 11/3/2025, 1:04 AM.

```

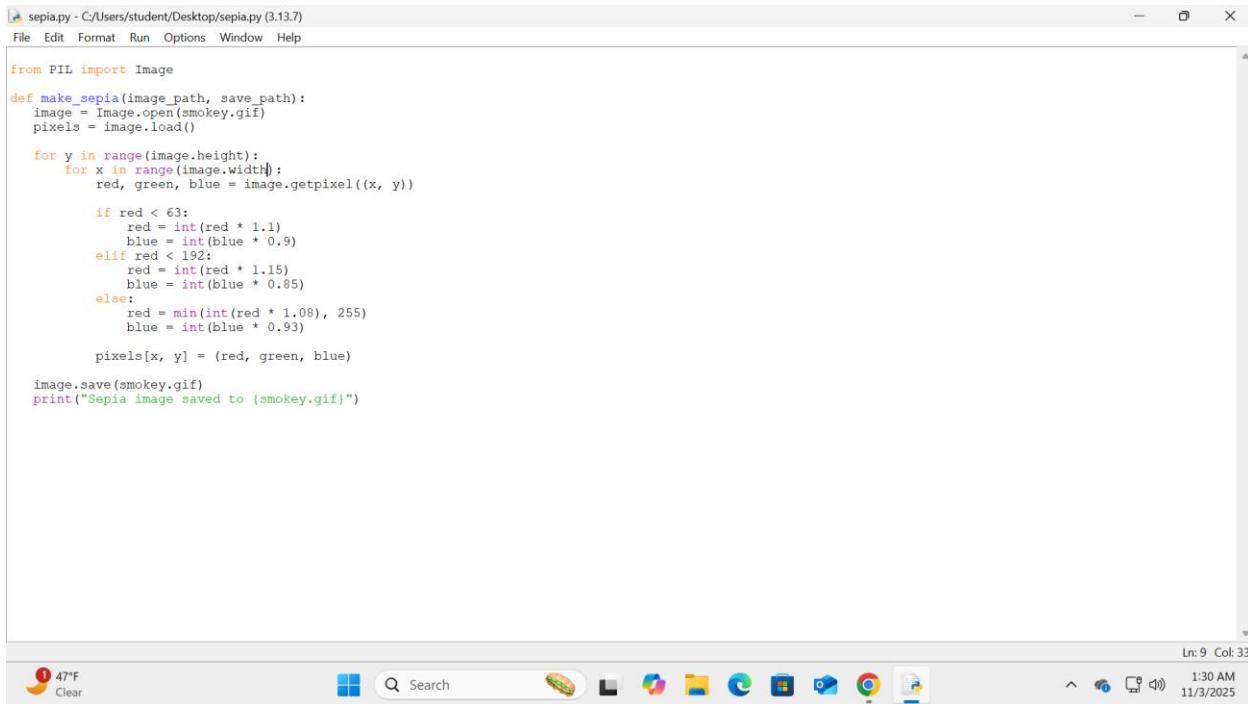
*IDLE Shell 3.13.7*
File Edit Shell Debug Options Window Help
Python 3.13.7 (tags/v3.13.7:bceec3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>> ===== RESTART: C:/Users/student/Desktop/koch.py =====

```

## Project # 8

### Special Instructions:

1. This project is to be done in a Python program file. The file needs to be submitted as part of the submission for this assignment. Place the program file into the zip file that is submitted for this assignment.



A screenshot of a Windows desktop environment. In the center is a code editor window titled "sepia.py - C:/Users/student/Desktop/sepia.py (3.13.7)". The window contains Python code for a sepia filter. The code imports the PIL module and defines a function "make\_sepia" that takes an image path and a save path. It opens a GIF file, loads pixels, and iterates through each pixel. For each pixel, it checks its red value against three thresholds (63, 192, and 255) and applies a sepia tone formula. Finally, it saves the modified image and prints a success message. The code editor has a menu bar with File, Edit, Format, Run, Options, Window, Help. The status bar at the bottom right shows "Ln: 9 Col: 33". Below the code editor is the Windows taskbar with icons for Start, Search, File Explorer, Control Panel, Task View, Mail, Microsoft Edge, Google Chrome, and File Explorer. The system tray shows the date and time as "11/3/2025 1:30 AM".

```
from PIL import Image

def make_sepia(image_path, save_path):
    image = Image.open(smokey.gif)
    pixels = image.load()

    for y in range(image.height):
        for x in range(image.width):
            red, green, blue = image.getpixel((x, y))

            if red < 63:
                red = int(red * 1.1)
                blue = int(blue * 0.9)
            elif red < 192:
                red = int(red * 1.15)
                blue = int(blue * 0.85)
            else:
                red = min(int(red * 1.08), 255)
                blue = int(blue * 0.93)

            pixels[x, y] = (red, green, blue)

    image.save(sepia.gif)
    print("Sepia image saved to (sepia.gif)")
```

IDLE Shell 3.13.7

```
File Edit Shell Debug Options Window Help
Python 3.13.7 (tags/v3.13.7:bceelc3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.

>>> ===== RESTART: C:\Users\student\Desktop\sepia.py =====
>>> |
```

Ln: 5 Col: 0

47°F  
Clear

Search



1:36 AM  
11/3/2025