03 Prove Milestone: Writing Functions

Purpose

Prove that you can write functions with parameters and call those functions with arguments.

Problem Statement

Modern computers are capable of performing all sorts of calculations to produce numbers. However, they are also capable of performing calculations to produce art, illustrations, animations, movies, and music. Python includes a library of code named `tkinter` that enables a program to draw shapes in a computer window. You can write a program that calls the `tkinter` drawing functions to draw an illustration in a computer window.

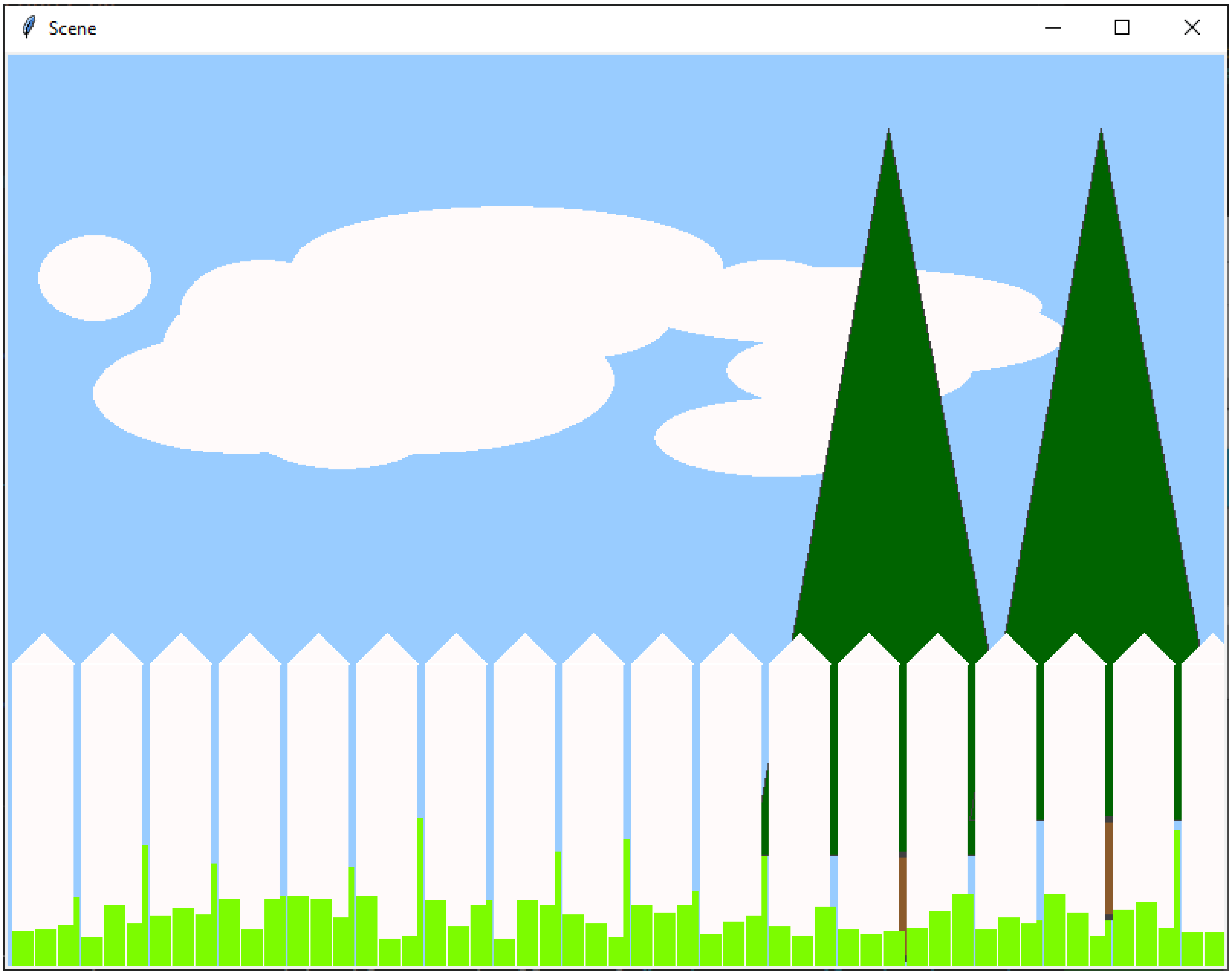
Assignment

During this prove milestone and the next prove assignment, you will write a Python program that uses the `tkinter` library to draw a semi-realistic outdoor scene. Your program can draw any outdoor scene that you like as long as it meets these requirements:

- The scene must be outdoor and include part of the sky.
- The sky must have clouds.
- The scene must include repetitive elements such as blades of grass, trees, leaves on a tree, birds, flowers, insects, fish, pickets in a fence, dashed lines on a road, buildings, bales of hay, snowmen, snowflakes, or icicles.
- Each of the objects in your scene should be drawn in its own function such as `draw_sky`, `draw_cloud`, `draw_ground`, `draw_flower`, `draw_fence`, `draw_fish`, etc.

During this milestone, you will write code that draws the sky, the ground, and clouds in your scene. During the next lesson, you will write code that completes your scene. Be creative.

The example scene below was drawn by a student's Python program and fulfills the requirements of this assignment.



Scene Gallery

If you would like ideas about what your program can draw, look at the scenes in this gallery. All the scenes in the gallery were drawn by programs written by former CSE 111 students.

Helpful Documentation

- The prepare content for this lesson explains how to write functions.
- The following videos walk through examples of using functions to draw with `tkinter`.
  - Drawing with Functions, part 1 (16 minutes)
  - Drawing with Functions, part 2 (15 minutes)
- A tutorial about drawing with tkinter
- References for tkinter Canvas methods
  - create_arc()
  - create_line()
  - create_oval()
  - create_polygon()

create_rectangle()

create_text()

- Chart of named colors in tkinter

Steps

Begin your program by copying and pasting the following code into a new file named `scene.py`. This beginning code imports the `tkinter` library and creates a window and a canvas that your program can draw to.

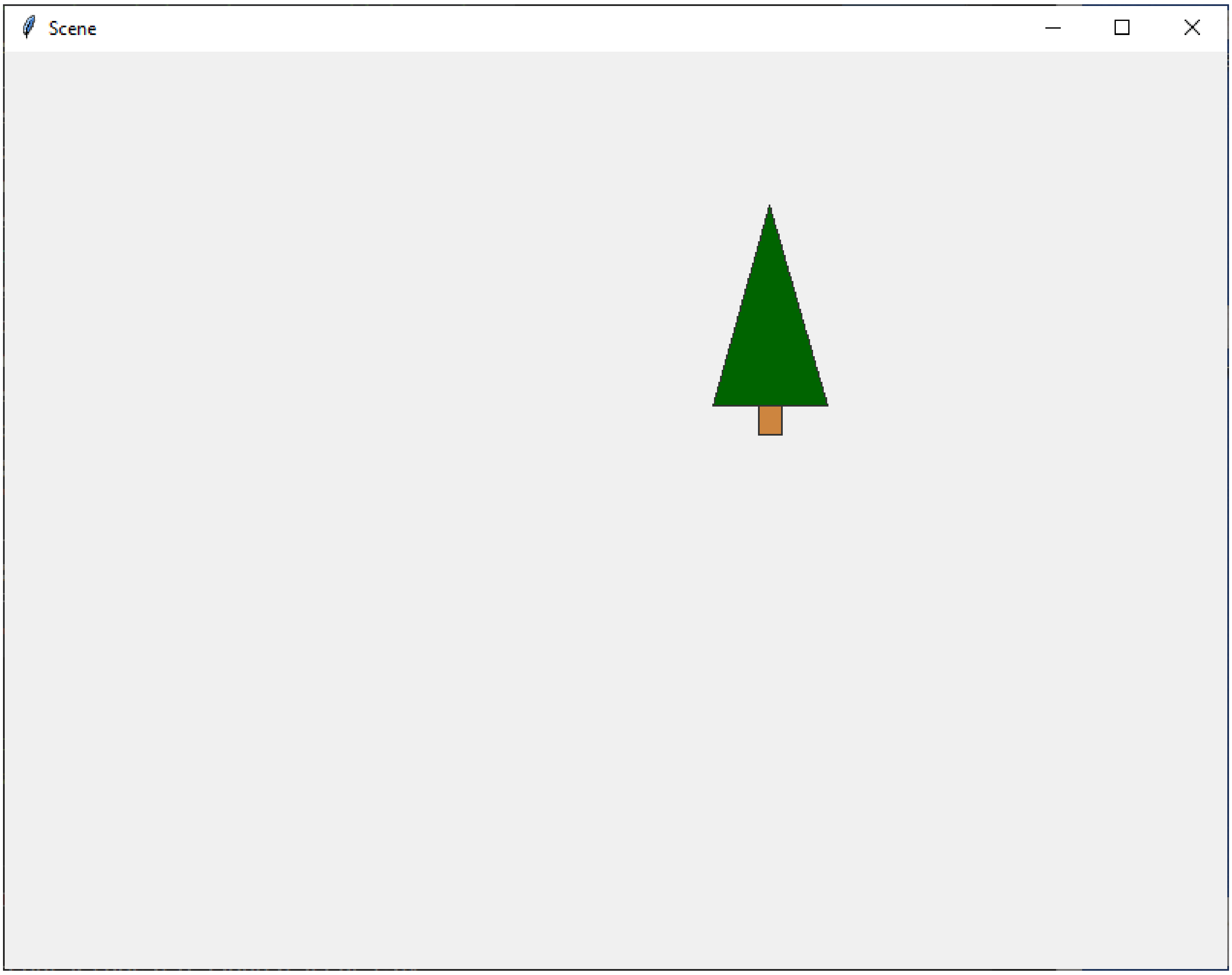
```python
import tkinter as tk


def main():
    # The width and height of the scene window.
    width = 800
    height = 500

    # Create the Tk root object.
    root = tk.Tk()
    root.geometry(f"{width}x{height}")

    # Create a Frame object.
    frame = tk.Frame(root)
    frame.master.title("Scene")
    frame.pack(fill=tk.BOTH, expand=1)

    # Create a canvas object that will draw into the frame.
    canvas = tk.Canvas(frame)
    canvas.pack(fill=tk.BOTH, expand=1)

    # Call the draw_scene function.
    draw_scene(canvas, 0, 0, width-1, height-1)

    root.mainloop()


def draw_scene(canvas, scene_left, scene_top, scene_right,scene_bottom):
    """Draw a scene in the canvas. scene_left, scene_top,
    scene_right, and scene_bottom contain the extent in
    pixels of the region where the scene should be drawn.

    Parameters
        scene_left: left side of the region; less than scene_right
        scene_top: top of the region; less than scene_bottom
        scene_right: right side of the region
        scene_bottom: bottom of the region
    Return: nothing

    If needed, the width and height of the
    region can be calculated like this:
    scene_width = scene_right - scene_left + 1
    scene_height = scene_bottom - scene_top + 1
    """
    # Call your functions here, such as draw_sky, draw_ground,
    # draw_snowman, draw_tree, draw_shrub, etc.
    tree_center = scene_left + 500
    tree_top = scene_top + 100
    tree_height = 150
    draw_pine_tree(canvas, tree_center, tree_top, tree_height)


# Define more functions here, like draw_sky, draw_ground,
# draw_cloud, draw_tree, draw_kite, draw_snowflake, etc.


```

```
57 def draw_pine_tree(canvas, peak_x, peak_y, height):
58     """Draw a single pine tree.
59     Parameters
60         canvas: The tkinter canvas where this
61             function will draw a pine tree.
62         peak_x, peak_y: The x and y location in pixels where
63             this function will draw the top peak of a pine tree.
64         height: The height in pixels of the pine tree that
65             this function will draw.
66     Return: nothing
67     """
68     trunk_width = height / 10
69     trunk_height = height / 8
70     trunk_left = peak_x - trunk_width / 2
71     trunk_right = peak_x + trunk_width / 2
72     trunk_bottom = peak_y + height
73
74     skirt_width = height / 2
75     skirt_height = height - trunk_height
76     skirt_left = peak_x - skirt_width / 2
77     skirt_right = peak_x + skirt_width / 2
78     skirt_bottom = peak_y + skirt_height
79
80     # Draw the trunk of the pine tree.
81     canvas.create_rectangle(trunk_left, skirt_bottom,
82             trunk_right, trunk_bottom,
83             outline="gray20", width=1, fill="tan3")
84
85     # Draw the crown (also called skirt) of the pine tree.
86     canvas.create_polygon(peak_x, peak_y,
87             skirt_right, skirt_bottom,
88             skirt_left, skirt_bottom,
89             outline="gray20", width=1, fill="dark green")
90
91
92 # Call the main function so that
93 # this program will start executing.
94 main()
```

Save your new file as `scene.py` and run your new Python program. When you run it, it should open a window that contains a single pine tree as shown in the image below. The pine tree was drawn because of the following:

1. The code at line 94 called the `main` function.
2. At line 23, the `main` function called the `draw_scene` function.
3. At line 50, the `draw_scene` function called the `draw_pine_tree` function.
4. At lines 81–89 the `draw_pine_tree` function called the `create_rectangle` and `create_polygon` methods.

Delete or comment-out lines 47–50 and lines 57–89 and begin writing your program. As you write your program, be sure that it draws elements in the order of farthest away to nearest. For example, you program should draw the blue sky first, then the ground, then clouds, then trees, then insects in the trees.



Intent

The intent of this assignment is not to train you to become a world-famous cloud artist. Instead, the intent is to teach you how to apply the following principles to a real programming problem:

1. Dividing a large program into parts (functions)
2. Writing a function
3. Deciding what needs to be a parameter in a function definition
4. Calling a function with different argument values to produce different results
5. Understanding how to separate the parts of a function that should stay the same and the parts that depend on parameters, in order to make the function as reusable as possible.

Your program must be divided into functions such as `draw_sky`, `draw_cloud`, `draw_ground`, `draw_bird`, `draw_grass_blade`, and `draw_insect`. The headers for the `draw_scene` and `draw_pine_tree` functions in the example code above are good examples for the headers of all your draw_* functions. Below are a few other examples of headers for functions that you might want to include in your program. However, these are simply examples, and you can change them in your program or even exclude them from your program.

```
def draw_house(canvas, house_left, house_bottom):

def draw_bird(canvas, bird_center, bird_top):

def draw_pebble(canvas, pebble_left, pebble_top, pebble_radius):

def draw_picket(canvas, picket_left, picket_bottom):
```

```
def draw_grass_blade(canvas, blade_left, blade_top, blade_height):
```

Testing Procedure

Verify that your program works correctly by following each step in this testing procedure:

1. Run your program and verify that it correctly opens a window and draws within that window an outdoor scene that contains at least the sky, clouds, and the ground.

Submission

On or before the due date, return to I-Learn and report your progress on this milestone.