Report 07

1. **Reflections**

- What was the easiest and hardest part of this assignment?

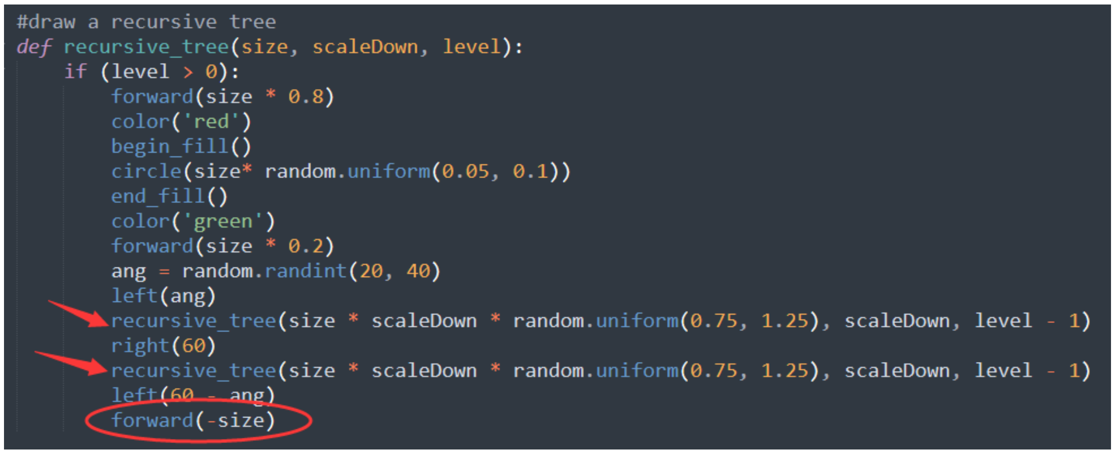
  Easiest part:

  Boxes, circles, and square spirals are easier to implement, mainly because the shape is simple and the base case is very clear.

  Hardest part:

  The recursive tree is the most difficult part, to draw a tree we need to set the heading to the north, however, we should not put the heading north code in the recursive function. It is very important to find out what should be inside the recursive function and what should not.

  After calling recursive_tree twice for the left and right branches, need to go back to where the function starts by forward(-size), similar to the algorithm of backtracking.



```
#draw a recursive tree
def recursive_tree(size, scaleDown, level):
    if (level > 0):
        forward(size * 0.8)
        color('red')
        begin_fill()
        circle(size* random.uniform(0.05, 0.1))
        end_fill()
        color('green')
        forward(size * 0.2)
        ang = random.randint(20, 40)
        left(ang)
        recursive_tree(size * scaleDown * random.uniform(0.75, 1.25), scaleDown, level - 1)
        right(60)
        recursive_tree(size * scaleDown * random.uniform(0.75, 1.25), scaleDown, level - 1)
        left(60 - ang)
        forward(-size)
```

- What did you learn?

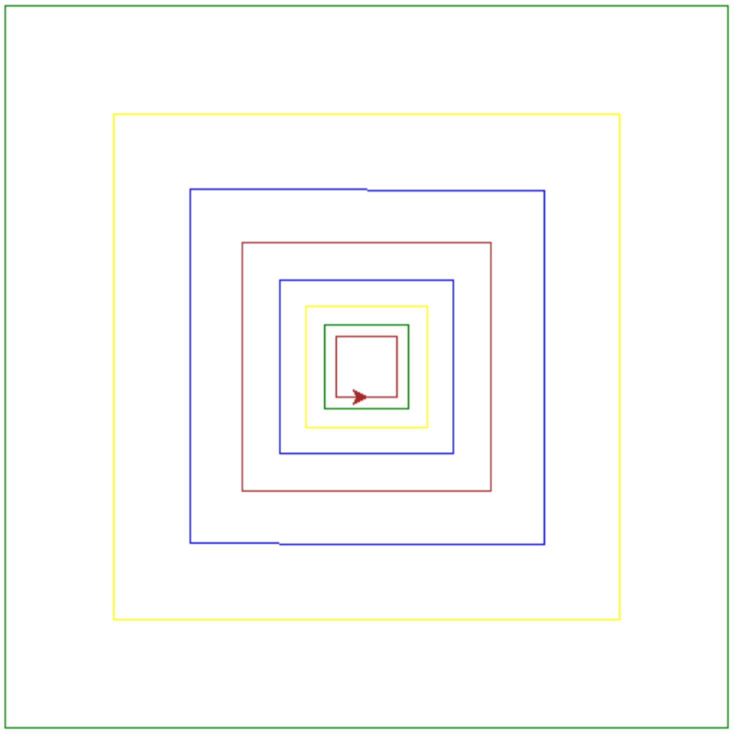
  Recursion: the concept, principle under the hood, base case, and usage. Though recursion is not the most efficient way to write code, it can make the code elegant sometimes.

  Error handling: use try:…except… syntax to catch errors instead of causing termination of the program.

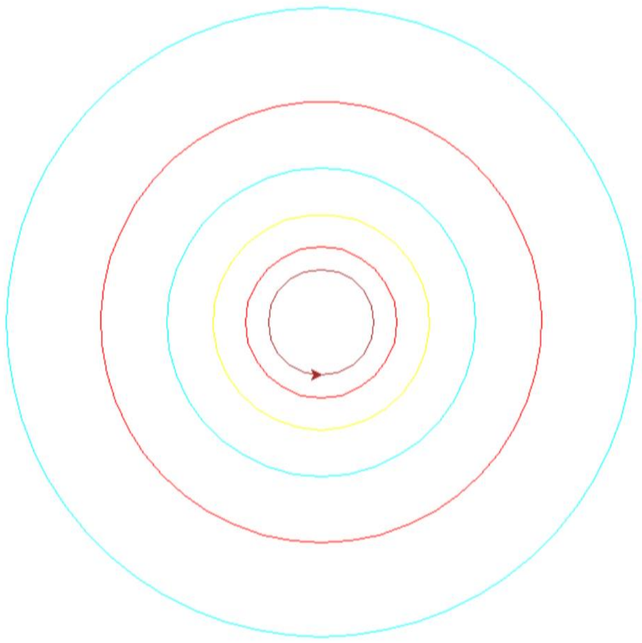
- Did the recitation strategy session help at all?

  In the recitation strategy session, We get an agreement on the general idea and base case to implement simple recursive shapes such as the recursive circle or boxes, this saves a lot of time when I write the code since the idea is clear.

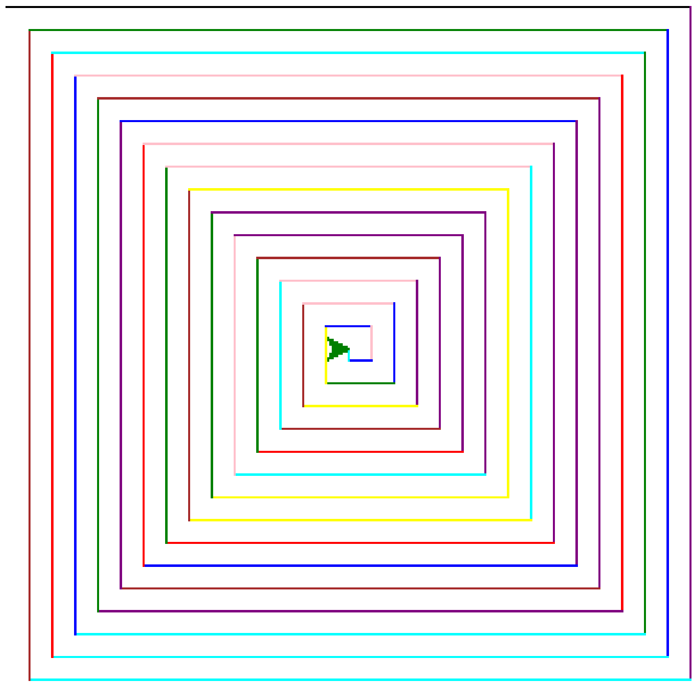
2. **Shapes**



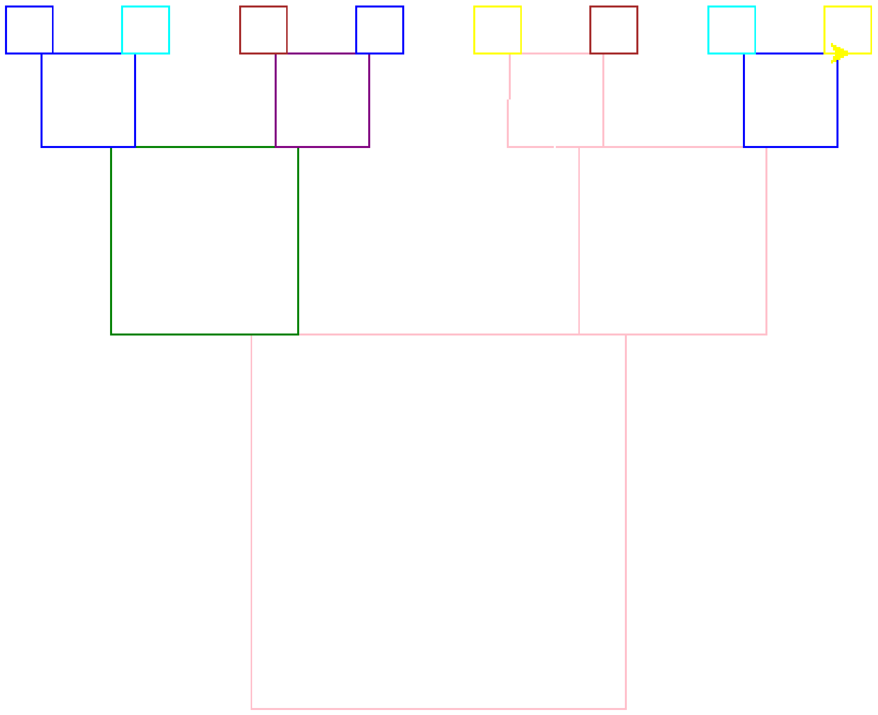
Psychedelic boxes



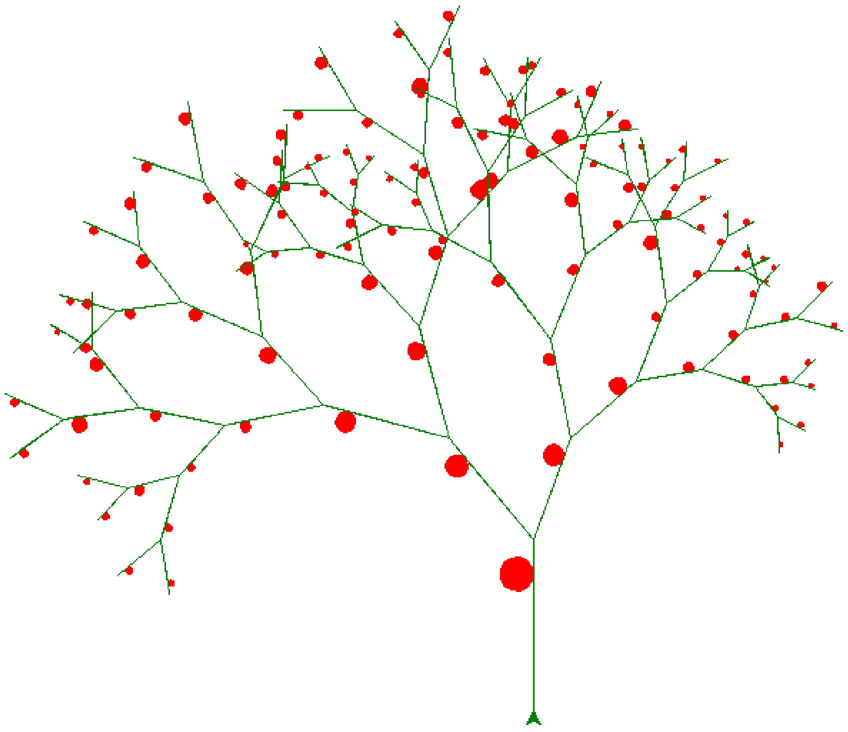
Bull’s Eye



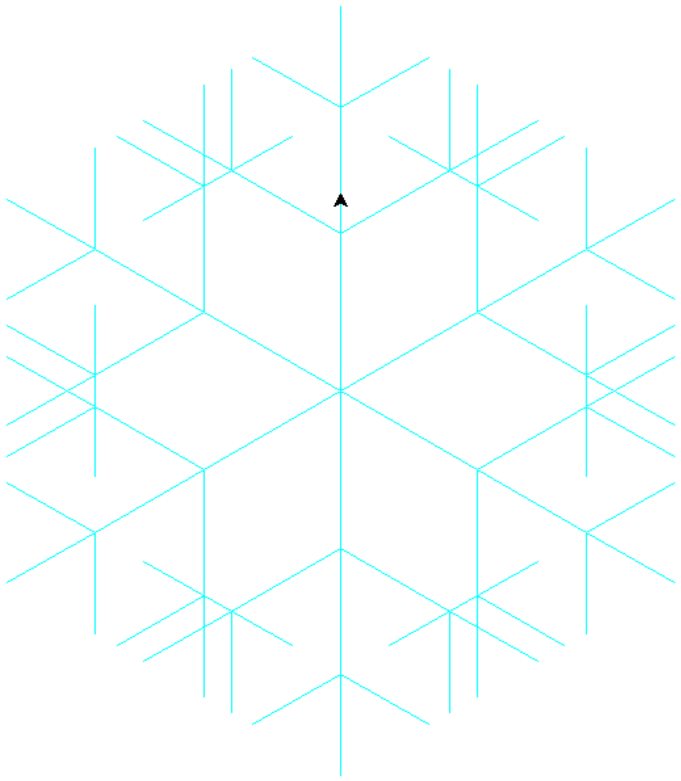
Inward Turning Spiral



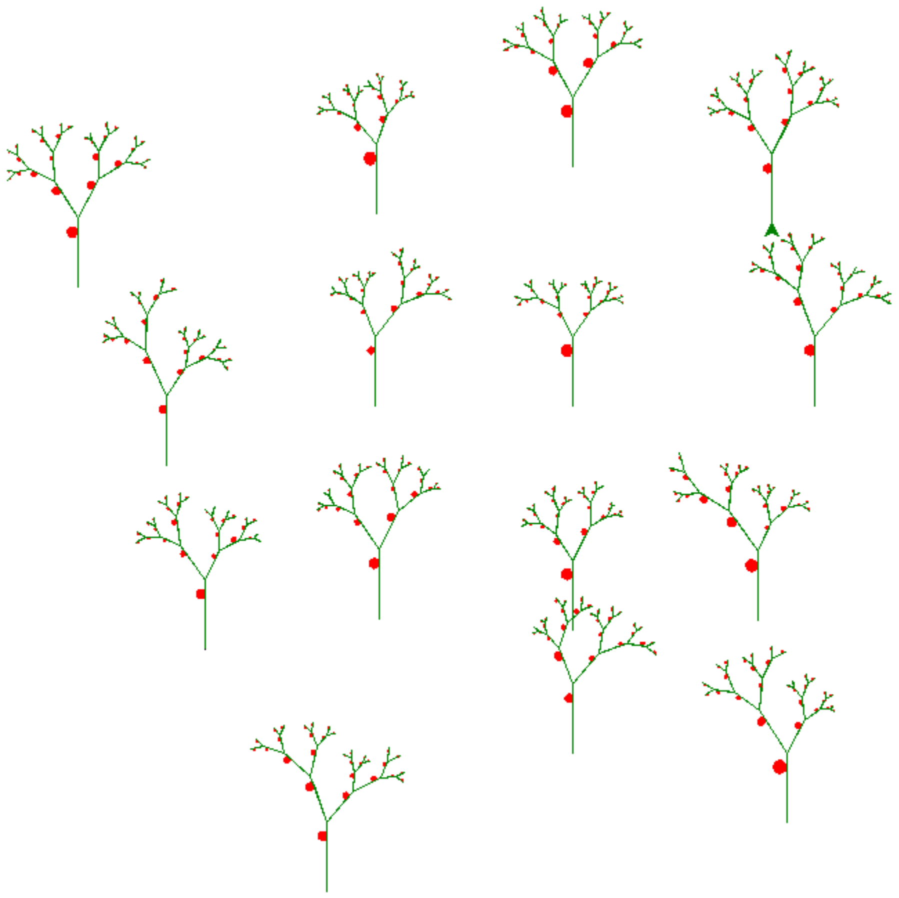
Box Tree



Realistic tree



Snowflake



Forest

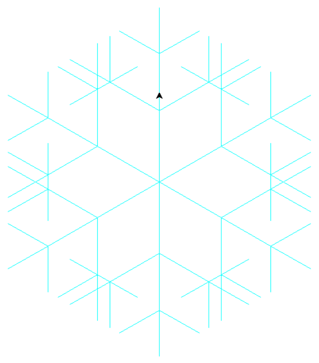
3. Extensions

- **Add some color to make shapes more interesting**

  As can be seen in the above shapes, all the drawings are in a given or random color.

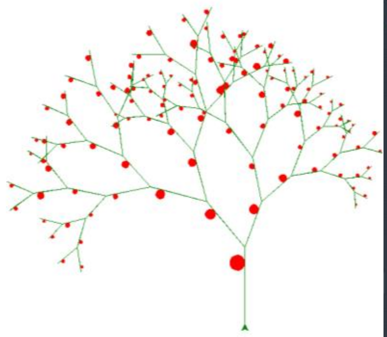
- **Created recursive shapes of own design**

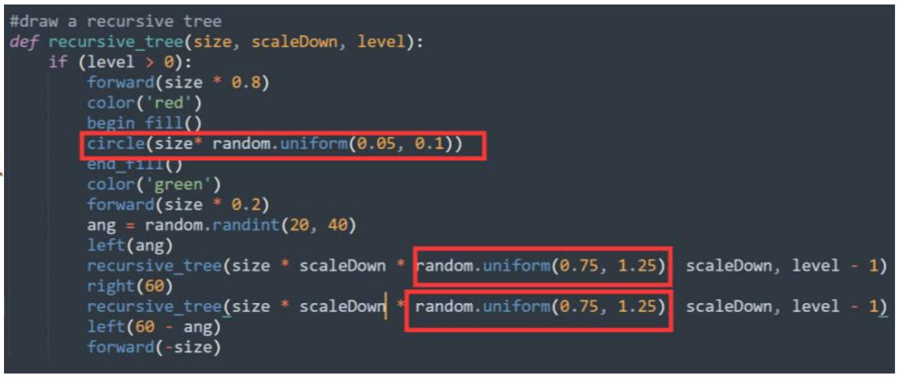
  Created a snowflake shape, which the algorithm is similar to the recursive tree.



- **Add some randomization to your shape creation**

  Random angle and length of the branch and fruit size are implemented to get a more random tree.

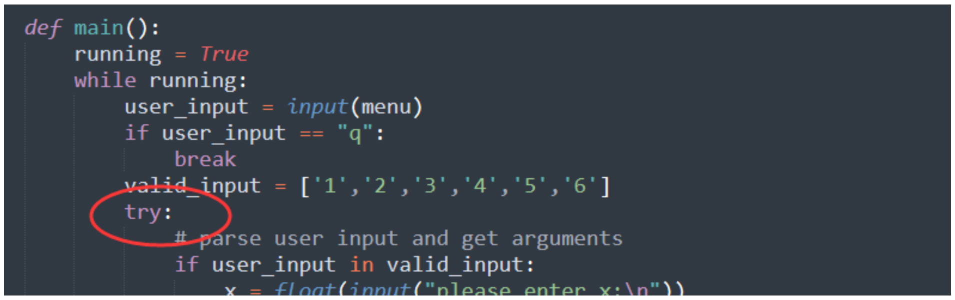




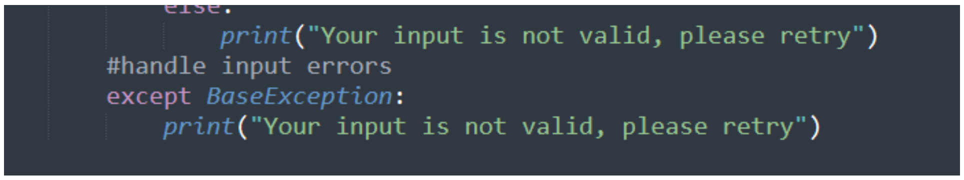
```
#draw a recursive tree
def recursive_tree(size, scaleDown, level):
    if (level > 0):
        forward(size * 0.8)
        color('red')
        begin_fill()
        circle(size* random.uniform(0.05, 0.1))
        end_fill()
        color('green')
        forward(size * 0.2)
        ang = random.randint(20, 40)
        left(ang)
        recursive_tree(size * scaleDown * random.uniform(0.75, 1.25) scaleDown, level - 1)
        right(60)
        recursive_tree(size * scaleDown * random.uniform(0.75, 1.25) scaleDown, level - 1)
        left(60 - ang)
        forward(-size)
```

Also used random coordination for trees in the forest shape, and random colors in the other shapes

- **Use try…except syntax in the CLI user interface for error handling**

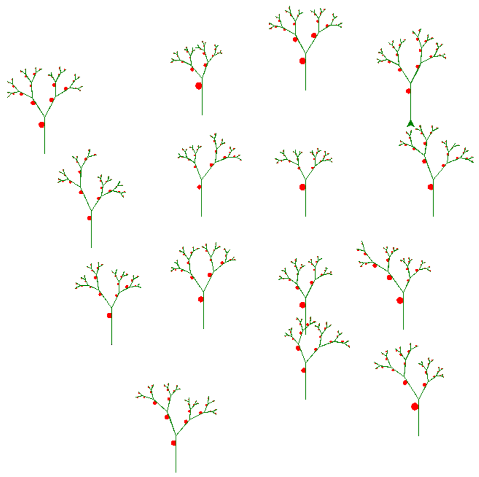


```
def main():
    running = True
    while running:
        user_input = input(menu)
        if user_input == "q":
            break
        valid_input = ['1','2','3','4','5','6']
        try:
            # parse user input and get arguments
            if user_input in valid_input:
                x = float(input("please enter x:\n"))
```



```
                else:
                    print("Your input is not valid, please retry")
            #handle input errors
            except BaseException:
                print("Your input is not valid, please retry")
```

- **Nested for loops to form a forest**



- **Code reused**

Set code blocks that are often used in helper methods to avoid repetitive code:

rect(size) is a function to get a square and this function, is used in:

boxes(x, y, size, scaleDown, min_size)

boxTree(x, y, size, scaleDown, min_size)

reset(x, y, angle) is a function to reset the x,y coordinate and the heading angle, used in:

boxes(x, y, size, scaleDown, min_size)

boxTree(x, y, size, scaleDown, min_size)

circles(x, y, radius, scaleDown, min_radius)

snowflake(x, y, size, scaleDown, level)

and main function

rand_color() is a function to return a random color string, used in all shapes with random color

recursive_tree(size, scaleDown, level) is a function to get a tree, it is reused in forest() function

And for each recursive shape, the shape function calls itself and this is also the reuse of code.

4. Grading Statement

The python program includes all of the shapes requested in the rubric and runs fine without significant errors.

Extension ideas such as color, own designed shape, randomization, error handling, and code reuse were implemented.

For the above reasons, I think this work deserves a full mark.