Activity 1.1.9 - Algorithms and Art

CSP Period 5 - 9/29/21

Kyle Huang

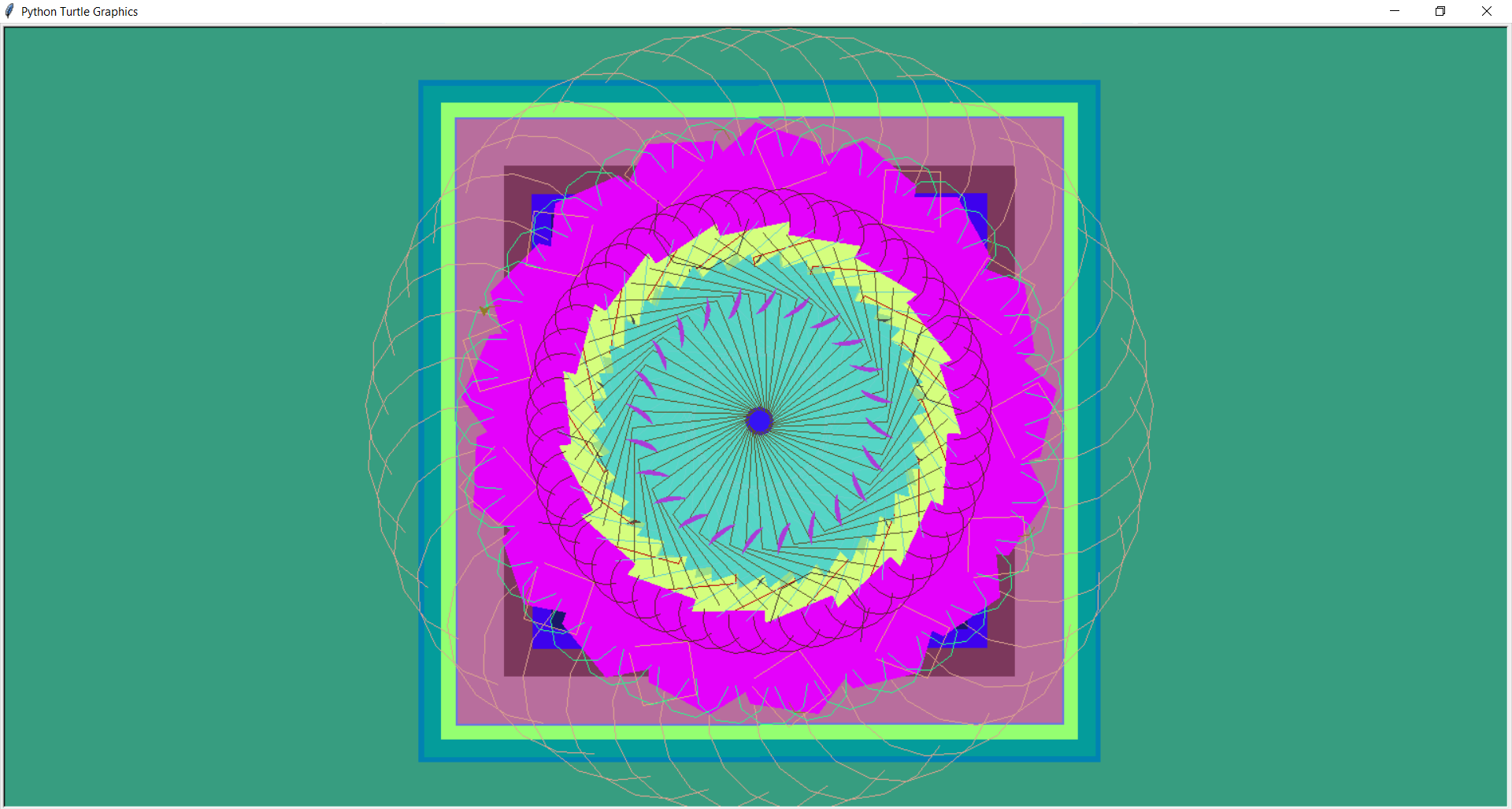
[Python file (plaintext, not formatted)](https://drive.google.com/file/d/18niHiba3_Xvl60tzaPdiiw_yT-9yBBUn/view?usp=sharing)

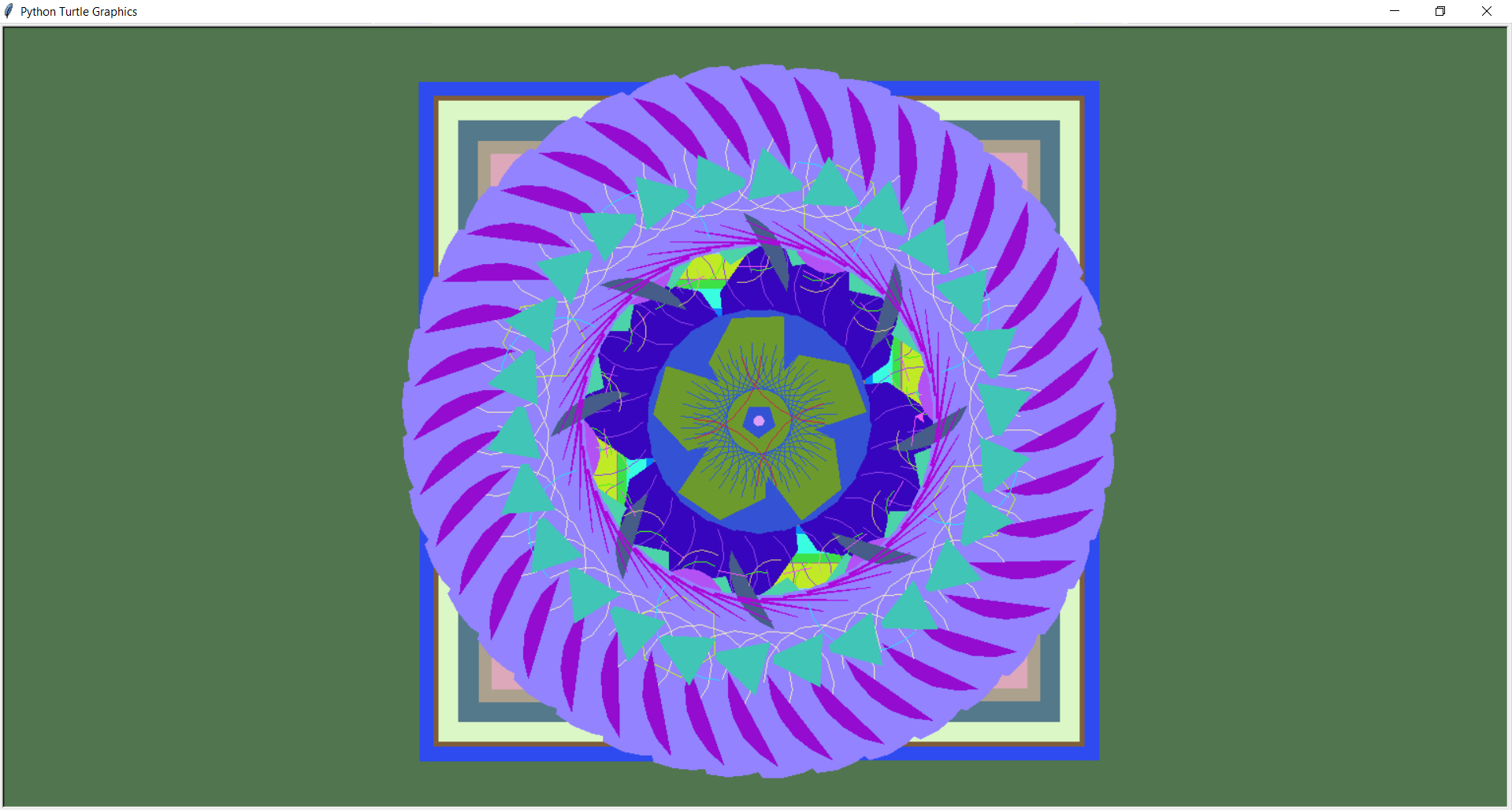
[Python file (with VSC text formatting)](https://docs.google.com/document/d/1peWqcl2v-AJTypkWHwj6mOx0ldEgMVflyzunpKRkLQI/edit?usp=sharing)

[Link to original google doc (if reading from PDF)](https://docs.google.com/document/d/1YKw5RD9RCT6DwiN7EZhw9wbvRekLw0_lbsviJ1oFrKM/edit?usp=sharing)

Screenshots of program:







3. The purpose of this program is to generate random geometric patterns, similar to mosaic tiles. This is done by combining a background made up of squares with random colors and circular patterns with random colors. Although these drawings are not meant to specifically illustrate anything, they may seem like flowers or mosaic tiles.

4.

Code segment 1:

| # Generate random list of square side lengths  squares = []  for i in range(random.randint(5,20)):  squares.append(random.randint(10,750))  # Create background square  draw\_square(10000, [random.randint(0,255),random.randint(0,255),random.randint(0,255)])  # Sort side lengths from greatest to least  squares.sort(reverse=True)  # Draw squares with random colors  for i in squares:  color = [random.randint(0,255),random.randint(0,255),random.randint(0,255)]  draw\_square(i, color) |
| --- |

This code segment draws random squares in the background of the image. In the first segment, the list squares is populated with 5 to 20 random numbers from 10 to 750, which represent the side lengths of the squares. Then, a giant square with side length 10000 and random colors is drawn in the background, using the draw\_square() function that was defined earlier in the program. Then, the list of square side lengths is sorted from greatest to least so larger squares do not accidentally cover smaller squares, and then a square is drawn with each side length with a random color by iterating through the list of side lengths.

Code segment 2:

| # Generate a random amount of random circular patterns  for i in range(random.randint(10,50)):  # Generate pattern attributes  radius = random.randint(10,300)  amount = random.randint(2,50)  shape\_side\_length = random.randint(10,100)  shape\_sides = random.randint(0,8)  shape\_color = [random.randint(0,255),random.randint(0,255),random.randint(0,255)]  shape\_extent = random.randint(30,360) \* int(random.choice([-1,1]))  shape\_offset = random.randint(0,360)  fill = bool(random.randint(0,1))  # Draw pattern  circular\_pattern(radius, amount, shape\_side\_length, shape\_sides, shape\_offset, shape\_color, shape\_extent, fill) |
| --- |

This code segment is similar to the previous one, but instead of squares, it draws circular patterns. First of all, attributes of the circular pattern, such as the radius and color, which is stored in a list, are randomly generated using random.randint(). Then, these attributes are used in the circular\_pattern() function, which was also defined earlier in the program, to draw the circular pattern. This is then repeated from 10 to 50 times to generate multiple circular patterns.