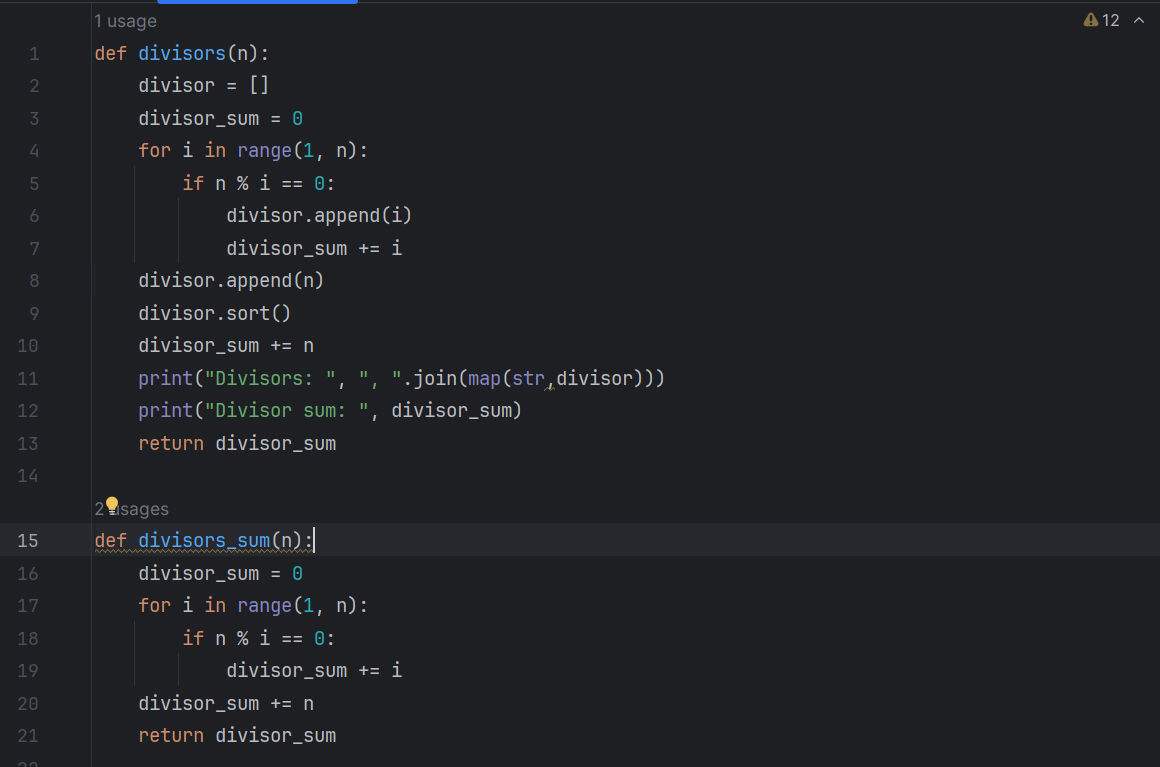
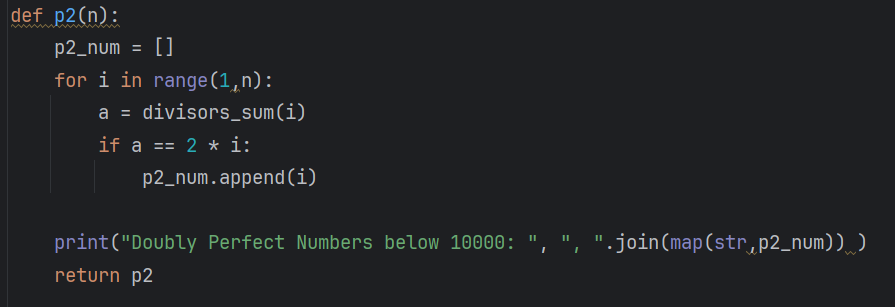
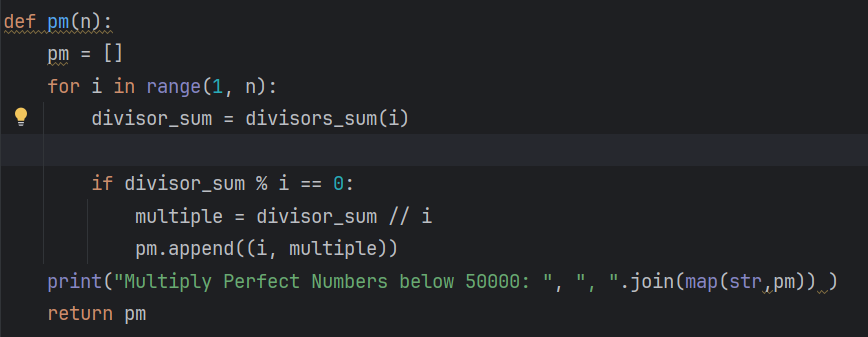
**Information Mid-Term Project Report**

ID: J8240003  
Name: Tausif Ibne Iqbal  
Codes available on GitHub: <https://github.com/Tausif30/Information>  
  
**Problem 1 (Multi-perfect number)**

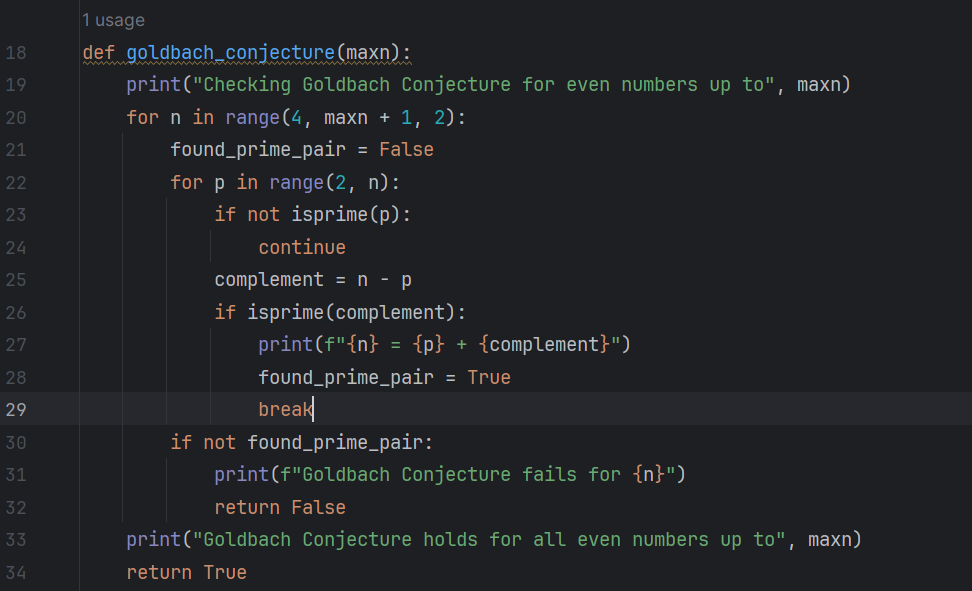
1. I looped through all the numbers up to the given number n. If the remainder of n is 0 while divided by 0 then that number is a factor. I defined a interger sum outside the loop with a value of 0 and inside the loop I kept adding the factors to it.
2. Used the previous divisor\_sum function and if the given number run inside divisor\_sum function equals twice that number then that number is a P2 number.
3. Used the modulus operator to see if the remainder is 0 or not, then used floor to find the quotient and thus defined the degree of the multi perfect number.
4. Sorry didn’t find any better algorithm. However the previous one is already O(n)

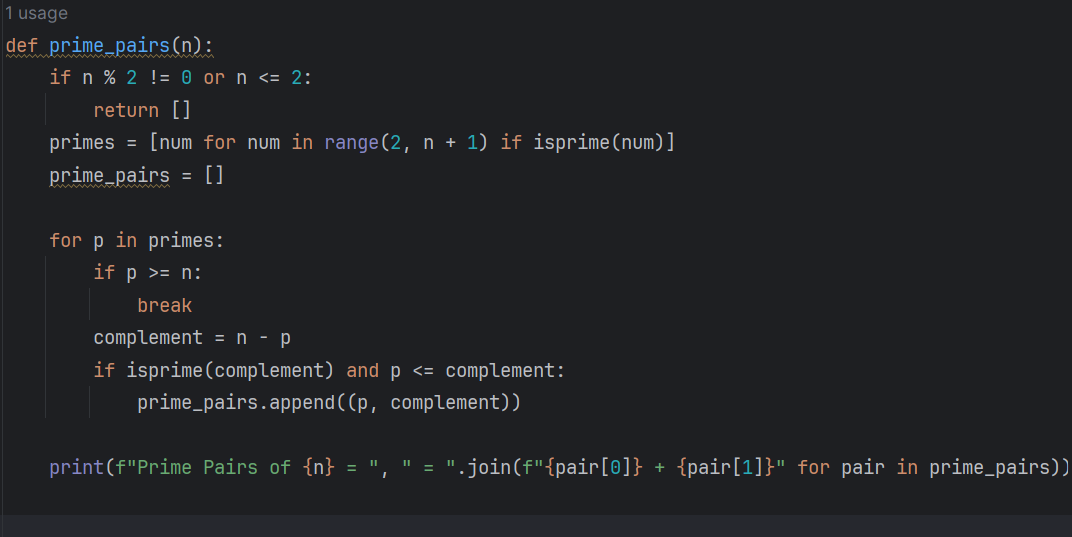
**Problem 2 (Goldbach Conjecture)**

1,2. Used prime number functions made for previous homework.

A computer screen shot of a program code

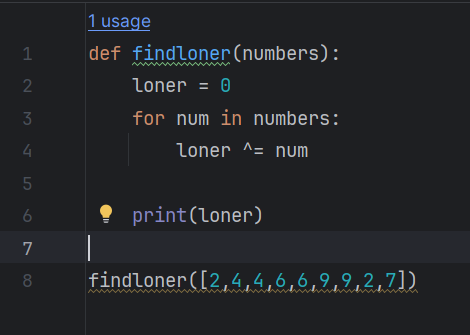
Description automatically generated

1. Ran a loop through all the numbers up to n and found the primes, p and then defined complement = n – p. If complement is also a prime number, then Goldbach Conjecture is proved true for that number. Now I just loop through all the numbers like this.
2. Same thing as the previous one. Made a prime\_pair list, kept looping throughout the number, subtracting primes with n to find the complements. If complement also turns out to be a prime number, then I appended those prime pairs in the list I made.



**Problem 3 (Loner Number)**

Defined a loner integer with initial value of 0. Then looped through the whole list of numbers in the function and used XOR operator with each of the numbers. This cancels out all the number except for the unique one.



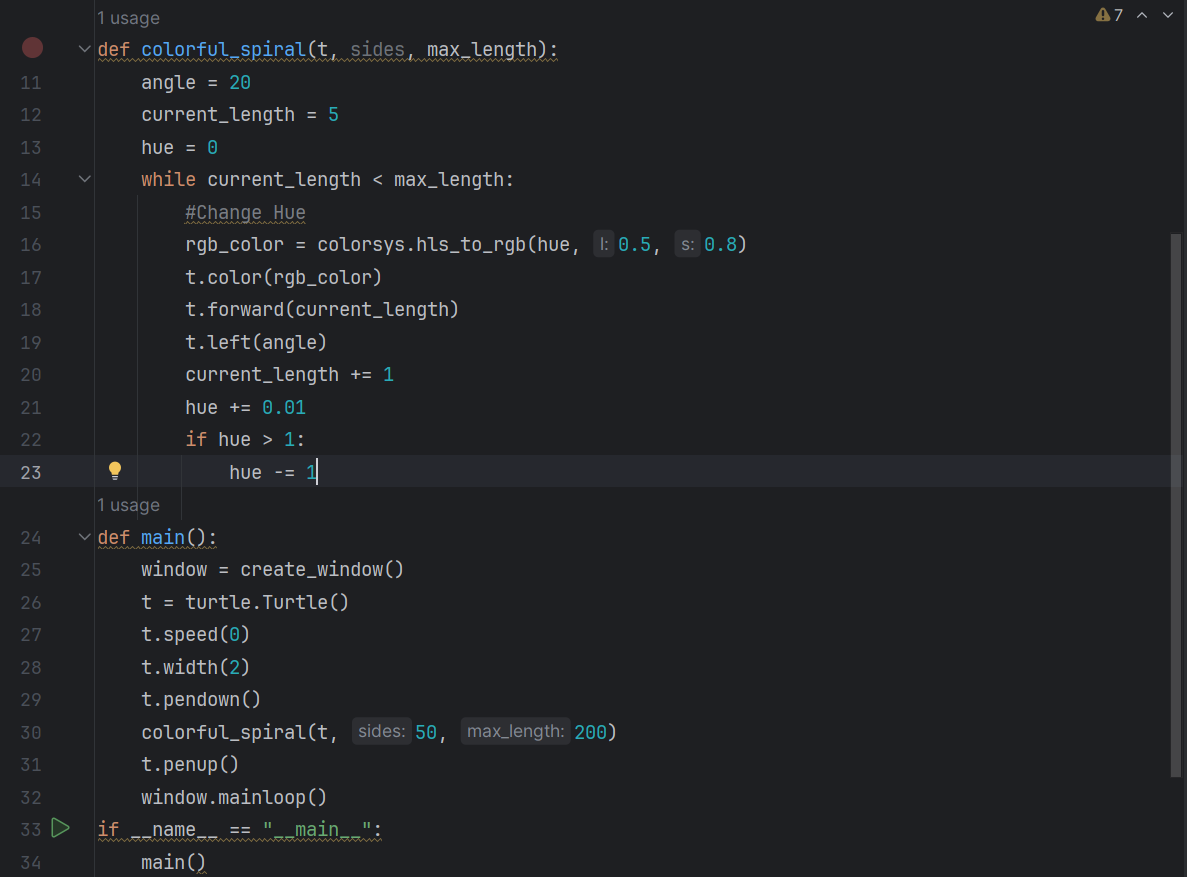
The XOR operator works as follows with the numbers.

0 ^ 2 = 2  
2 ^ 4 = 6  
6 ^ 4 = 2

It continues this way and at the end we get 7 as all of the others cancel each other out.

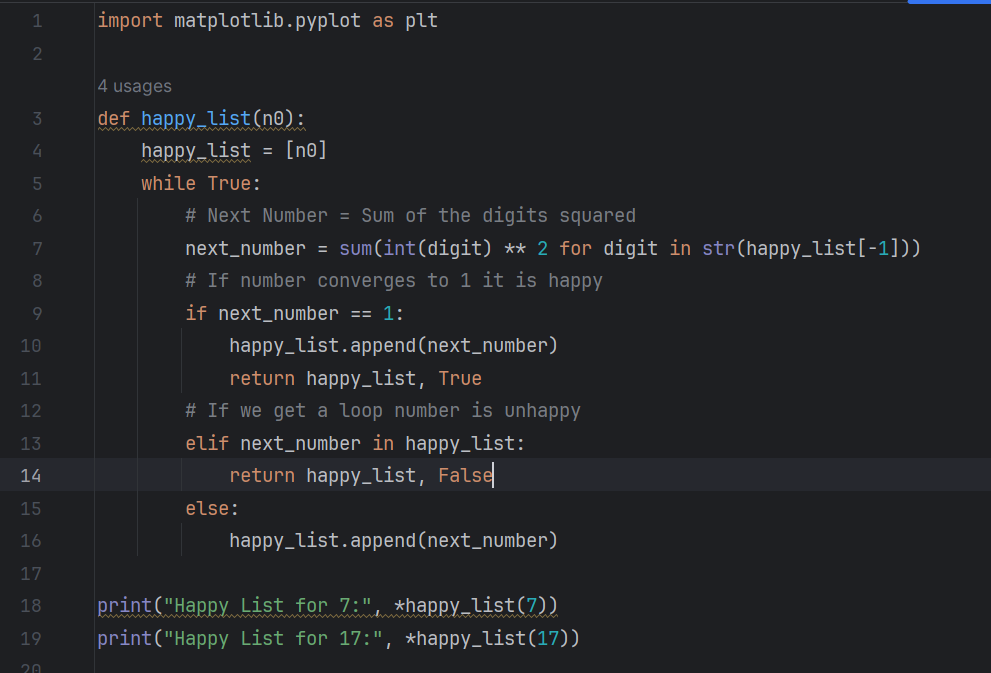
**Problem 4 (Colorful Spiral)**

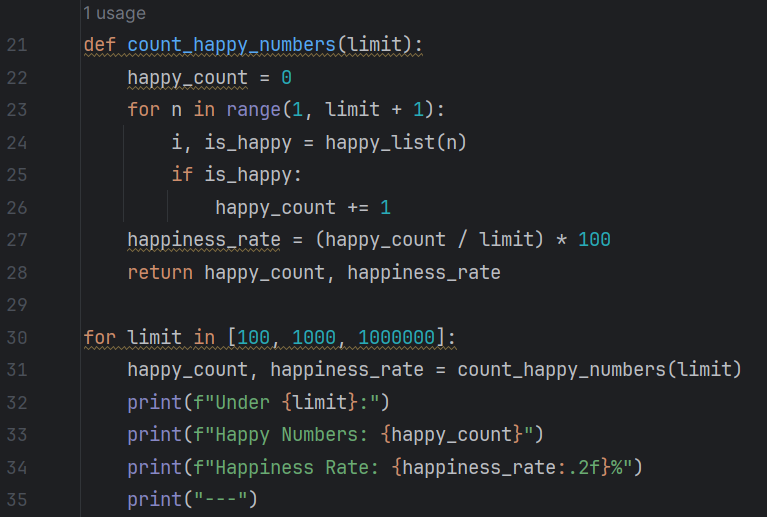
Use the same polygon function to draw a circle except now kept changing the side length of the polygon. The polar function of a spiral is . Used this idea that the radius is continuously changing. For the color change I used HSL color and kept changing the hue of the color.

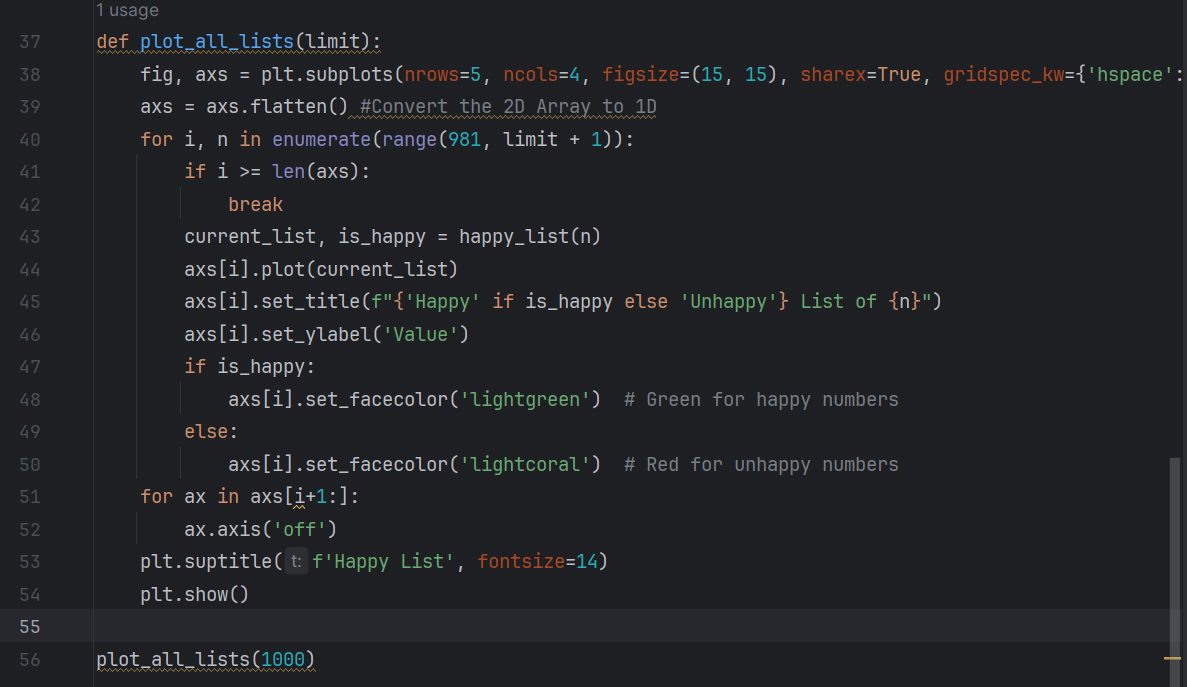


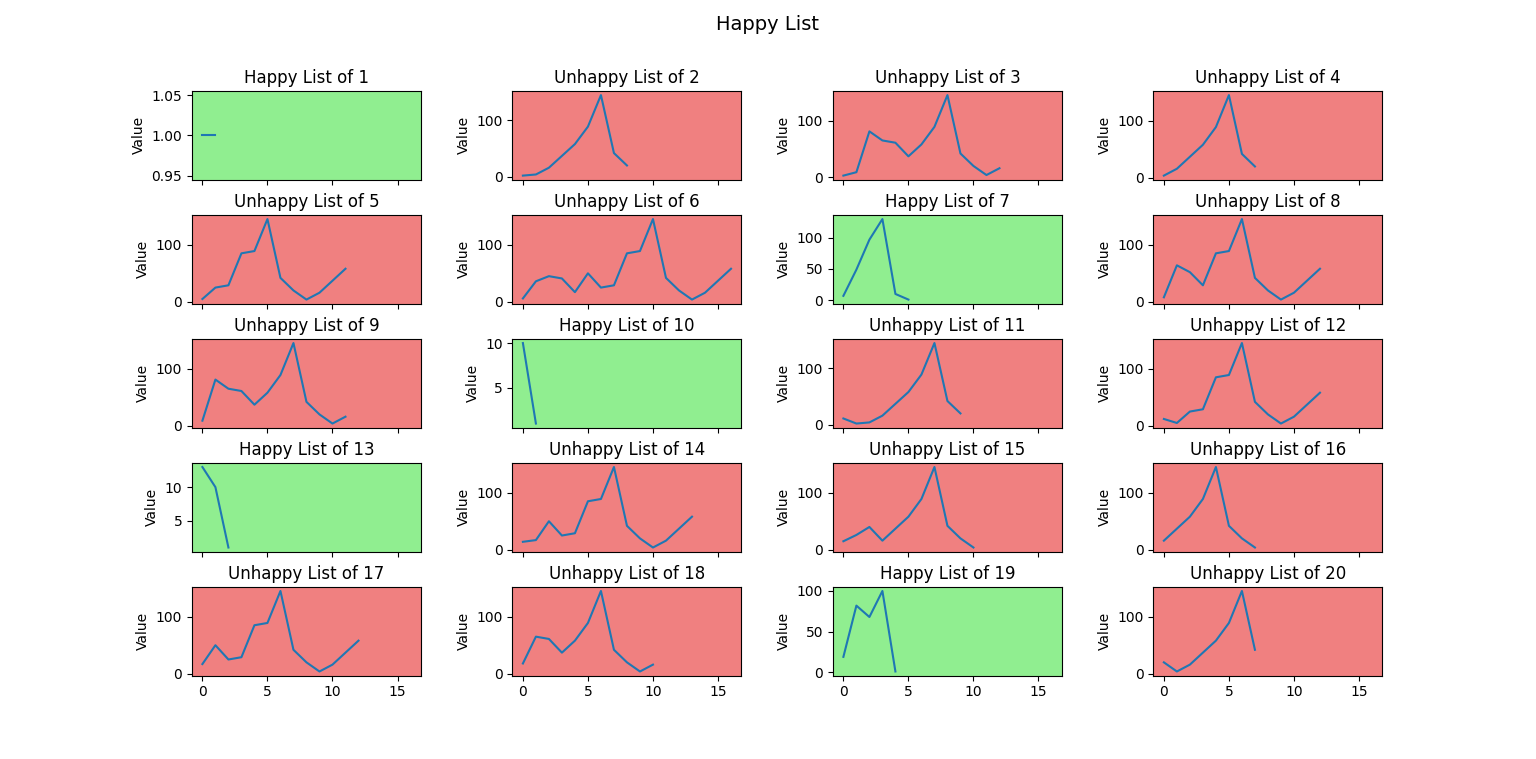
**Problem 5 (Happy Number)**

1. Turned the number into a list and then summed the square of all their digits to get the next numbers and if it converges to 1 then it’s True and if we get the number itself back (forms a loop) then we stop the function and conclude False.

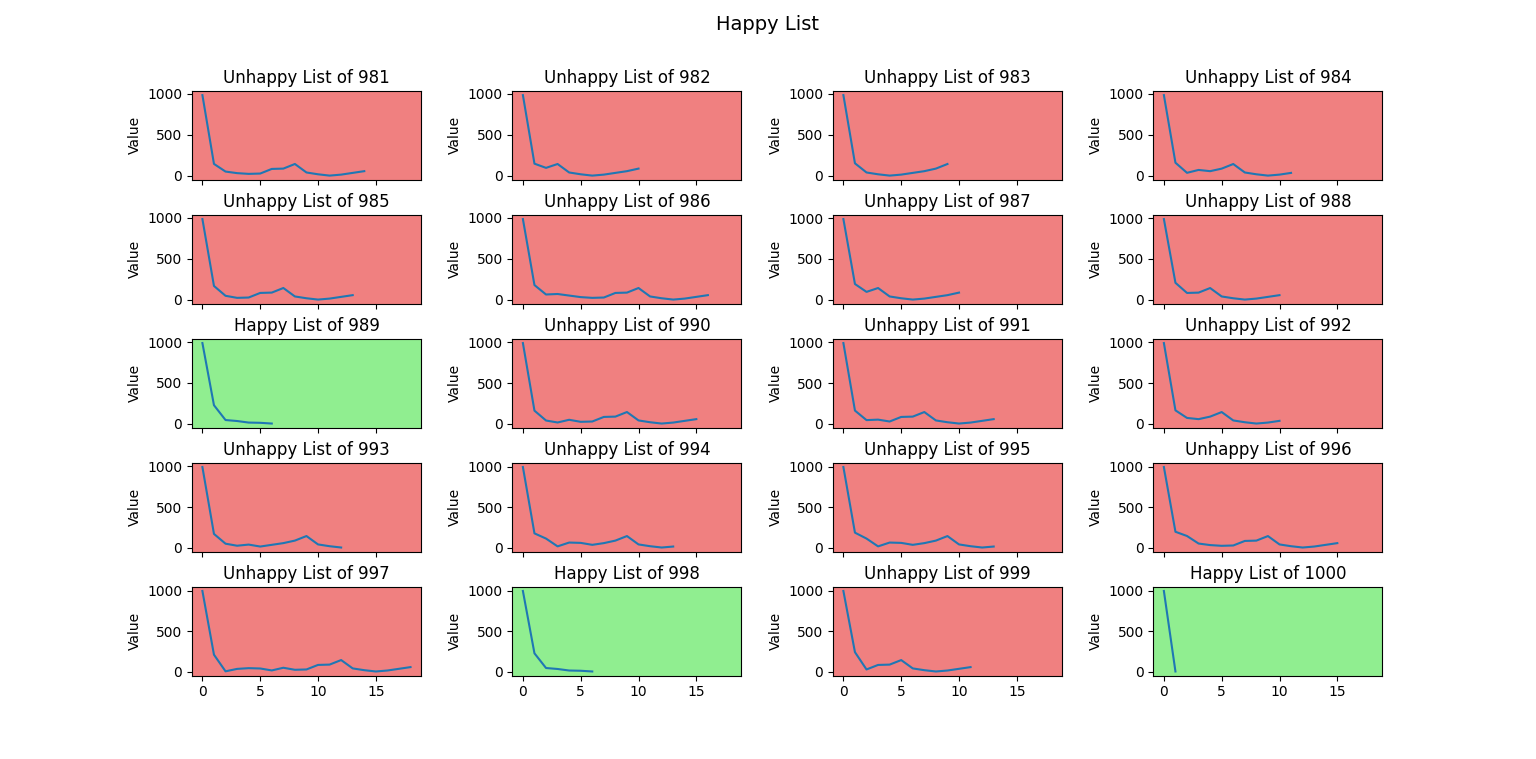


1. Used the same function to calculate the happy numbers and then add 1 to a count variable whenever we get a happy number. For happiness rate I just divided the count with the given number.
2. Used matplotlib to plot the graphs for each of the numbers.

It takes too much processing power to calculate all 1000 Happy numbers together and also gets messy. So, I just did 20 at a time. These are some of the plots I got.

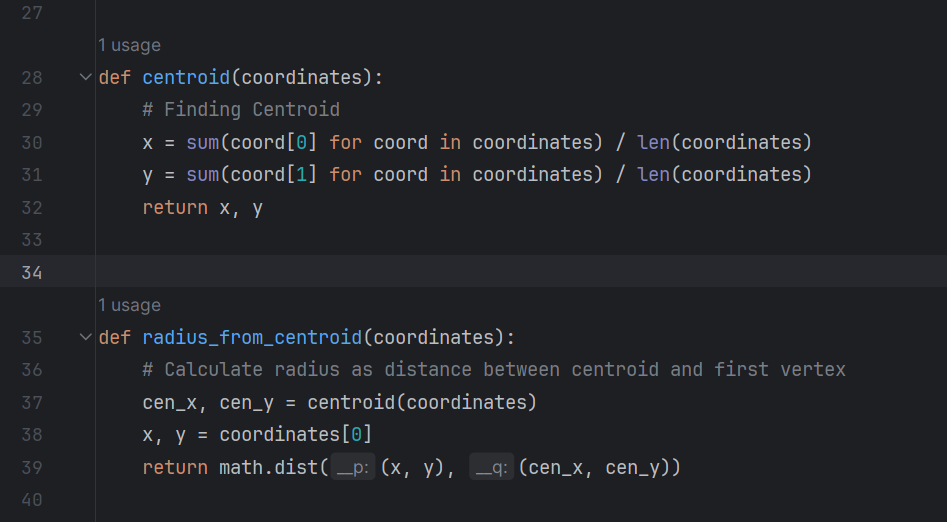


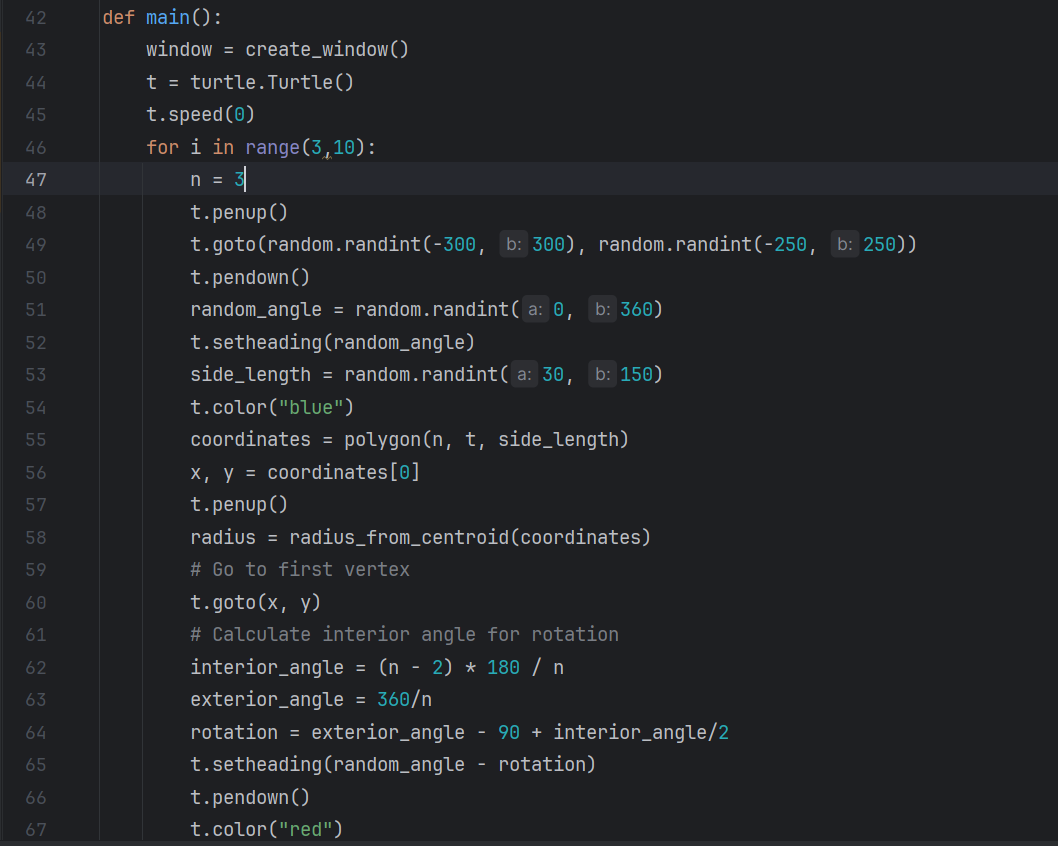
A group of graphs showing different colored lines

Description automatically generated with medium confidence

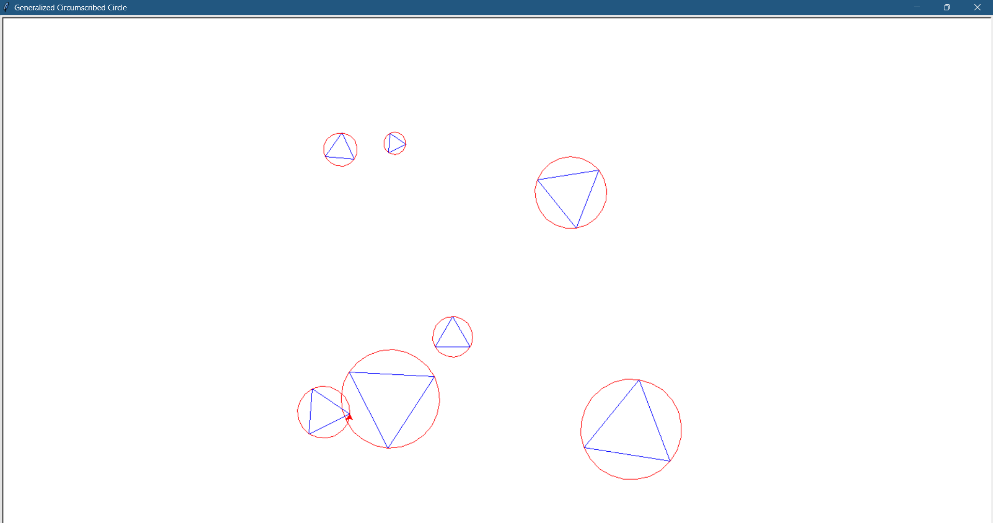
**Problem 5 (Circumscribed Polygons)**

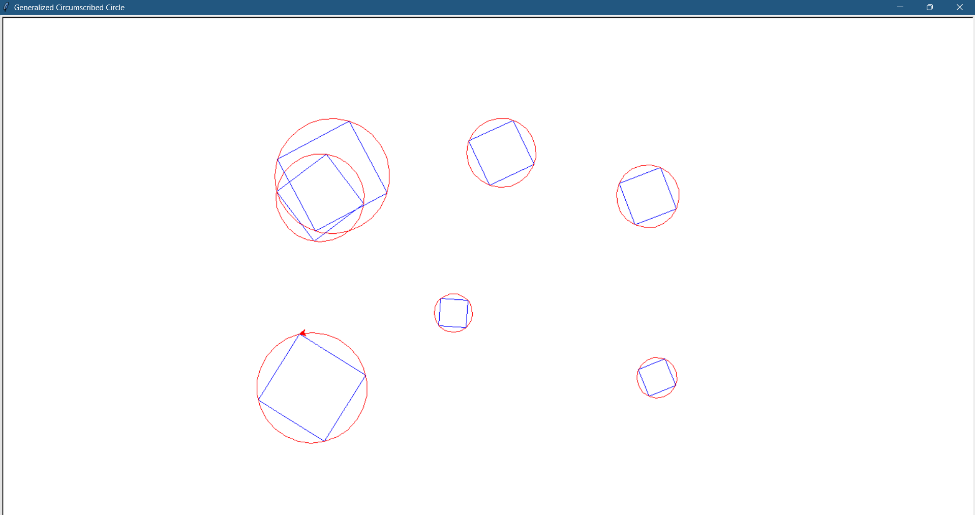
1. Defined polygon function from before for drawing regular polygons. Used random, to get random side lengths and used setheading and random function together to get the random orientation part.
2. I used the circle function in turtle. The centroid of the polygon is just the average of all its vertices. I found the radius by finding the distance between the centroid and any one of the vertices of the polygon. A circumscribed circle will be touching all the sides of the polygon. So, I started drawing the circle from the first vertex specifically vertex[0]. We get the circle, but it ‘s still not correctly oriented. After some drawings and trying out, I found that we need to rotate the circle clockwise by a degree of (Exterior Angle + Half of Interior Angle - 90). This comes mainly from the way I drew the polygon (forward left). It will be different if someone does it differently. Now I add the random angle to the required orientation of the circle and then use it with the setheading function and we get perfect circumscribed circles.

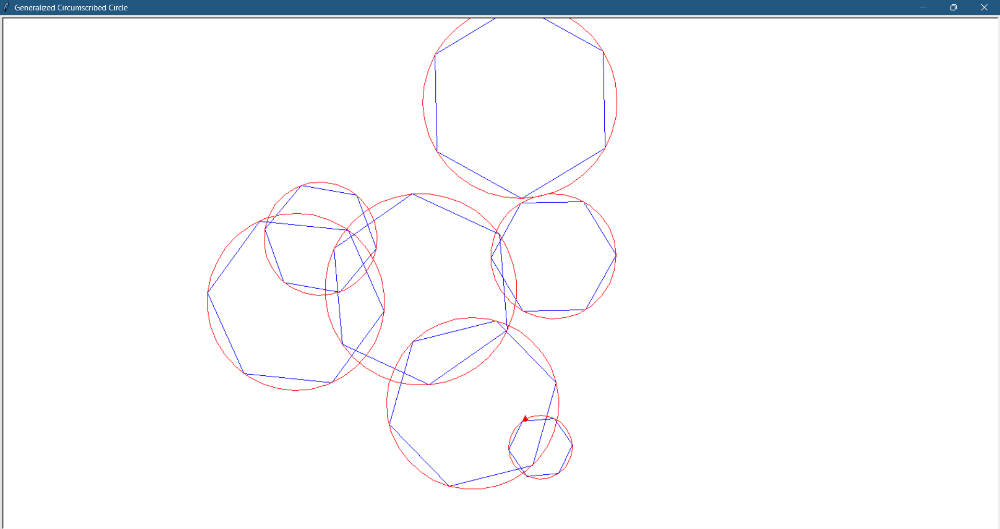




Circumscribed Triangles:



Circumscribed Squares:

Circumscribed Hexagons:

Observation: The polygon becomes closer to a circle and the area difference of the two figures gets closer to 0. This can be seen much better if we draw different polygons with circumscribed circles.

