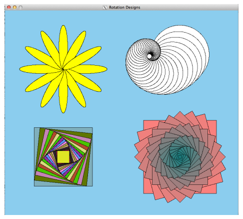
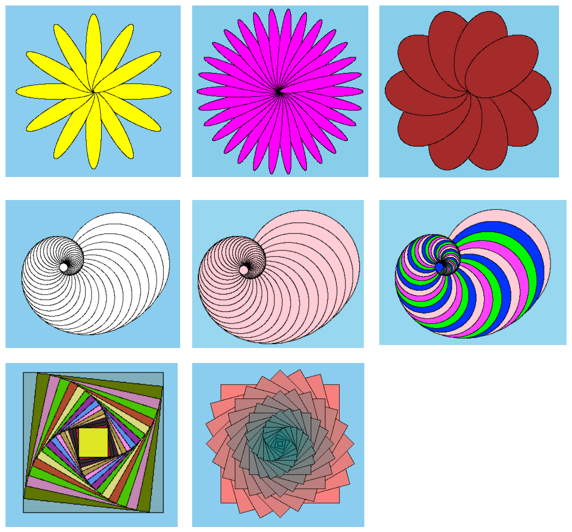
Graphics with Loops



CS111 Computer Programming

Department of Computer Science
Wellesley College

Motivation: How to create these pattern?

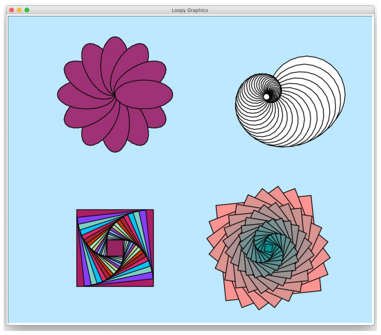


How to achieve repetition?

How to store color values so as to repeat them in the same order?

Graphics Examples with for Loops

We can use **for** loops in conjunction with the **range** function and the **turtle** module to create complex pictures with repeated subpatterns that are transformed by scaling, rotation, etc.

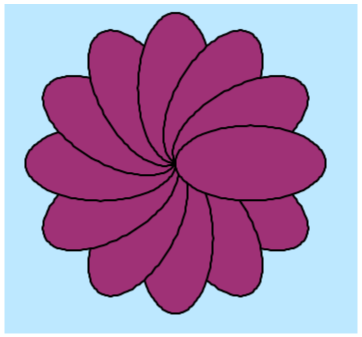


Each of these pictures is created by using a loop to create multiple copies of a simple shape (ellipse, circle, square) that differ in their rotation, size, and/or color.

A simple flower

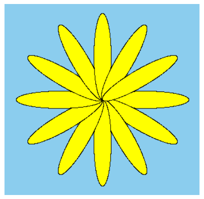
```
from turtle import *

for i in range(12):
    fillcolor('maroon4')
    #30 is 360/12
    rt(30)
    penup()
    fd(150/2)
    pendown()
    begin_fill()
    # the petals are 150 tall and 75 wide
    drawEllipse(150/4, 2)
    end_fill()
    penup()
    bk(150/2)
    pendown()
```

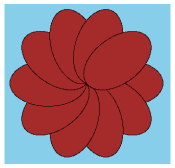


Abstracting over our flower with **makeFlower**

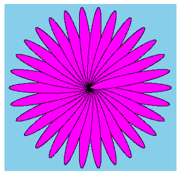
Define a function **makeFlower** that takes as arguments (1) the number of petals (2) the color of each petal (3) the width of each petal and (4) the height of each petal and draw the appropriate flower.



`makeFlower(12, 'yellow', 30, 150)`

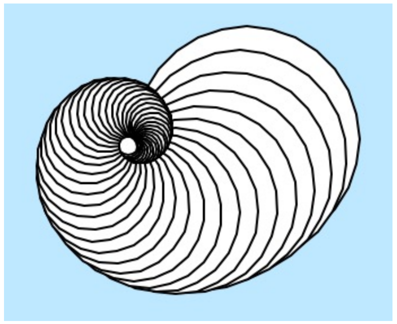


`makeFlower(10, 'brown', 90, 150)`



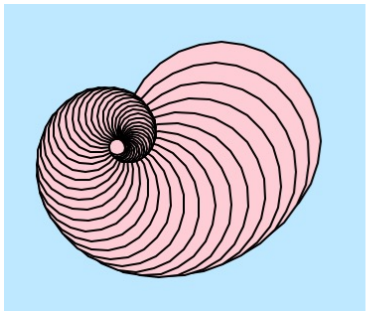
`makeFlower(30, 'magenta', 20, 150)`

A simple nautilus shell



```
for i in range(50):
    fillcolor('white')
    begin_fill()
    # make the circle a bit smaller in each iteration
    circle(100*(0.95**i))
    end_fill()
    rt(10)
```

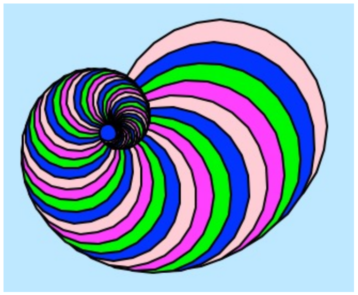
Parameterize it: **makeNautilus**



`makeNautilus(50, 100, 10, .95, 'pink')`

```
def makeNautilus(num, size, angle, shrink, color):
    for i in range(num):
        fillColor(color)
        begin_fill()
        circle(size*(shrink**i))
        end_fill()
        rt(angle)
```

Make it fancy: **makeColorfulNautilus**



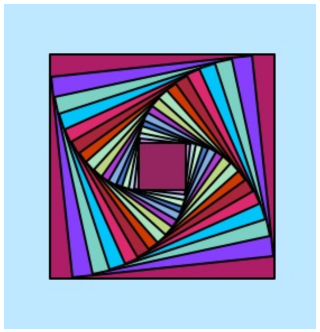
```
makeColorfulNautilus(50,100,
10, .95, ['pink','blue','green','magenta'])
```

```
def makeColorfulNautilus(num, size, angle, shrink, colorList):
    for i in range(num):
        fillColor(colorList[i % len(colorList)])
        begin_fill()
        circle(size*(shrink**i))
        end_fill()
        rt(angle)
```

The % operator makes sure that despite the value of num (and as a result, of i), the indices always are only between 0 and len of colorList.

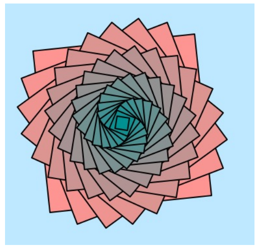
Rotated squares

The function **fillcolor** can get as arguments red, green, and blue values between 0 and 1. By using the function **random** will generate new colors in each iteration, thus you'll see a differently mix of colors every time you run the code.



```
for i in range(16):
    fillcolor(random(),random(),random())
    begin_fill()
    drawSquare(200*(0.9**i))
    end_fill()
    lt(6)
```

Rose-colored squares



```
for i in range(24,0,-1):
    fillcolor(i/24.0,0.5,0.5)
    begin_fill()
    drawSquare(10+10*i)
    end_fill()
    lt(15)
```

We must start drawing from the bigger squares, so the smaller squares are not covered but layered on top.