5. Introduction to Procedures

Topics:

The module simpleGraphics

Creating and Showing figures

Drawing Rectangles, Disks, and Stars

Optional arguments

Application Scripts

Procedures

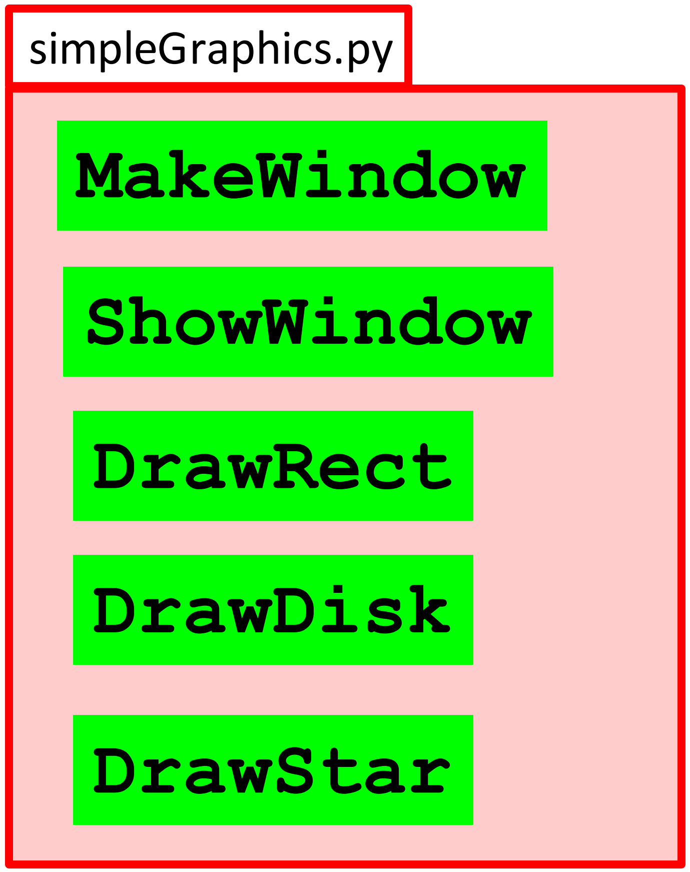
We continue our introduction to functions with a focus on procedures.

Procedures are functions that do not return a value.

Instead, they “do something.”

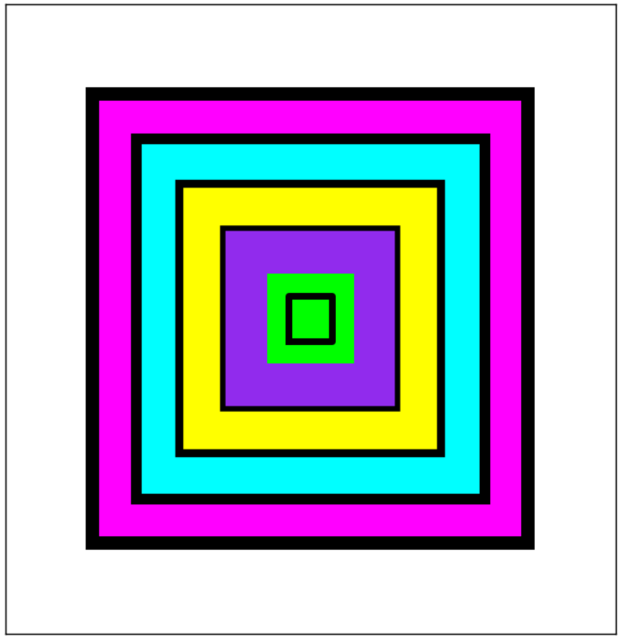
Graphics is a good place to illustrate the idea.

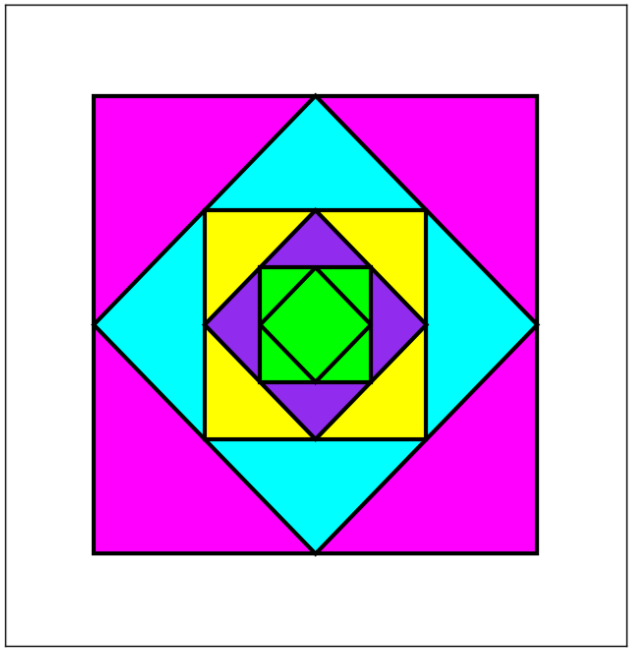
The Module simpleGraphics Has Five Procedures



We will use this module to make designs that involve rectangles, disks, and stars.

Examples that We Can do Right Now*





Looks like we will be able to draw tilted rectangles

* Right Now we only know about assignment statements and if-constructs.

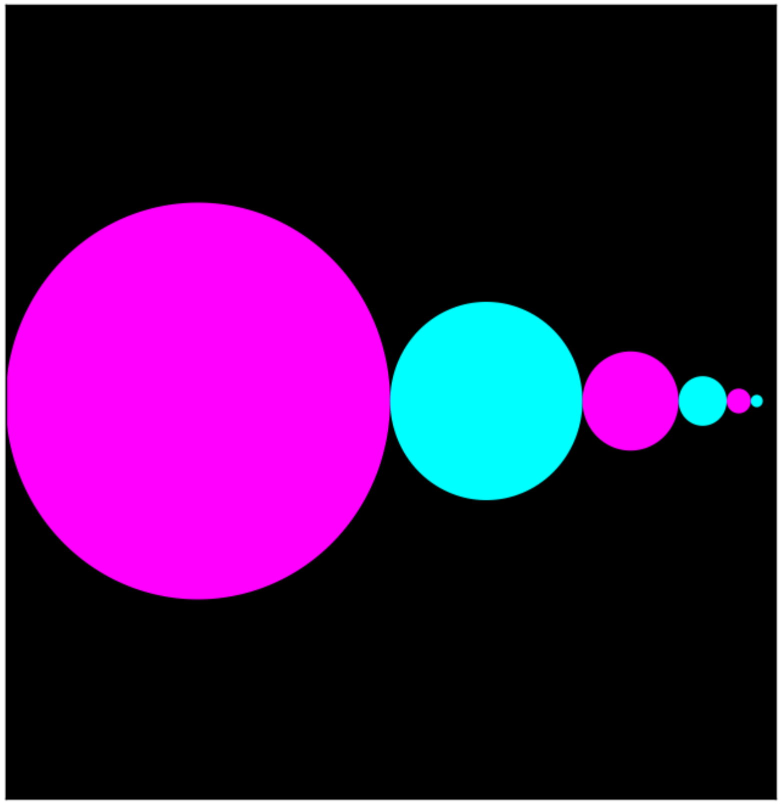
An Example We Can Do Right Now

How does color work?

What if we had 100 rows each with 100 stars ?

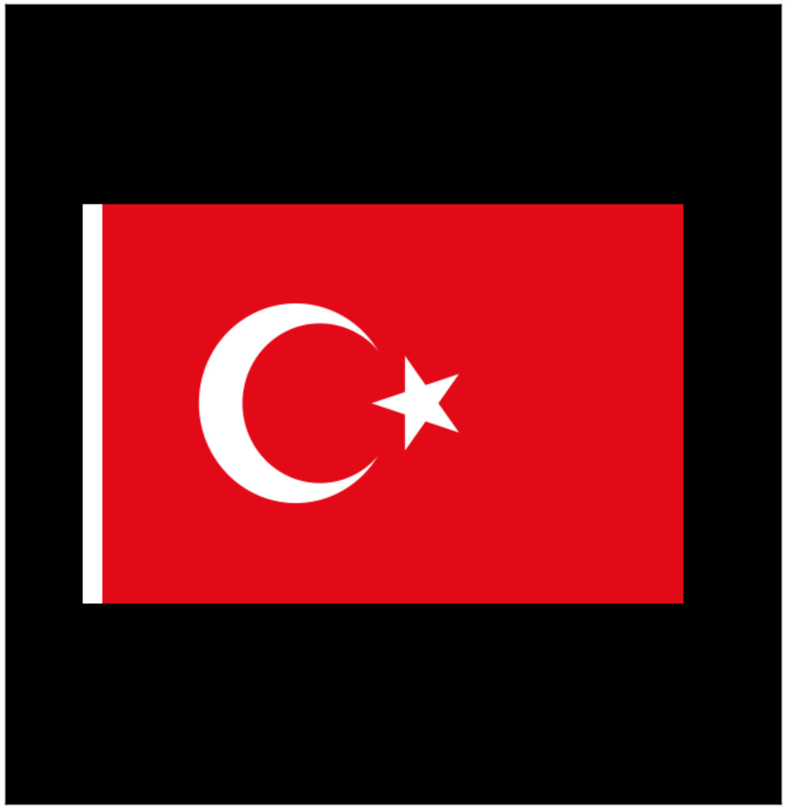
Anticipating loops.

An Example We Can Do Right Now



Xeno's Paradox: Will we ever reach the right edge?

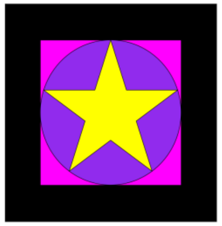
An Example We Can Do Right Now

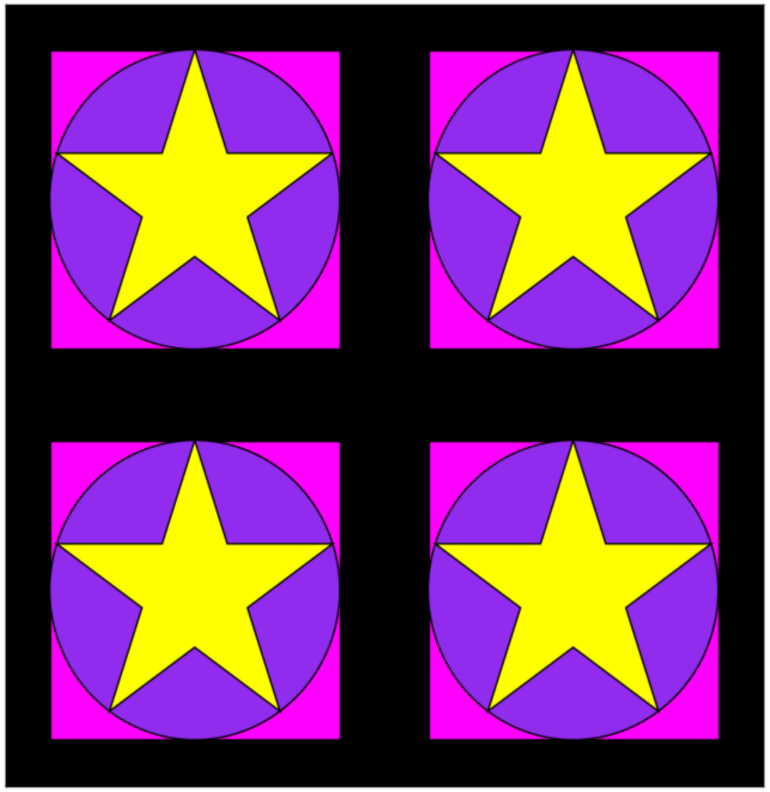


White Rectangle + Red Rectangle + White Disk + Red Disk + Tilted White Star

An Example We Can Do Right Now

Let's write a function to draw this:

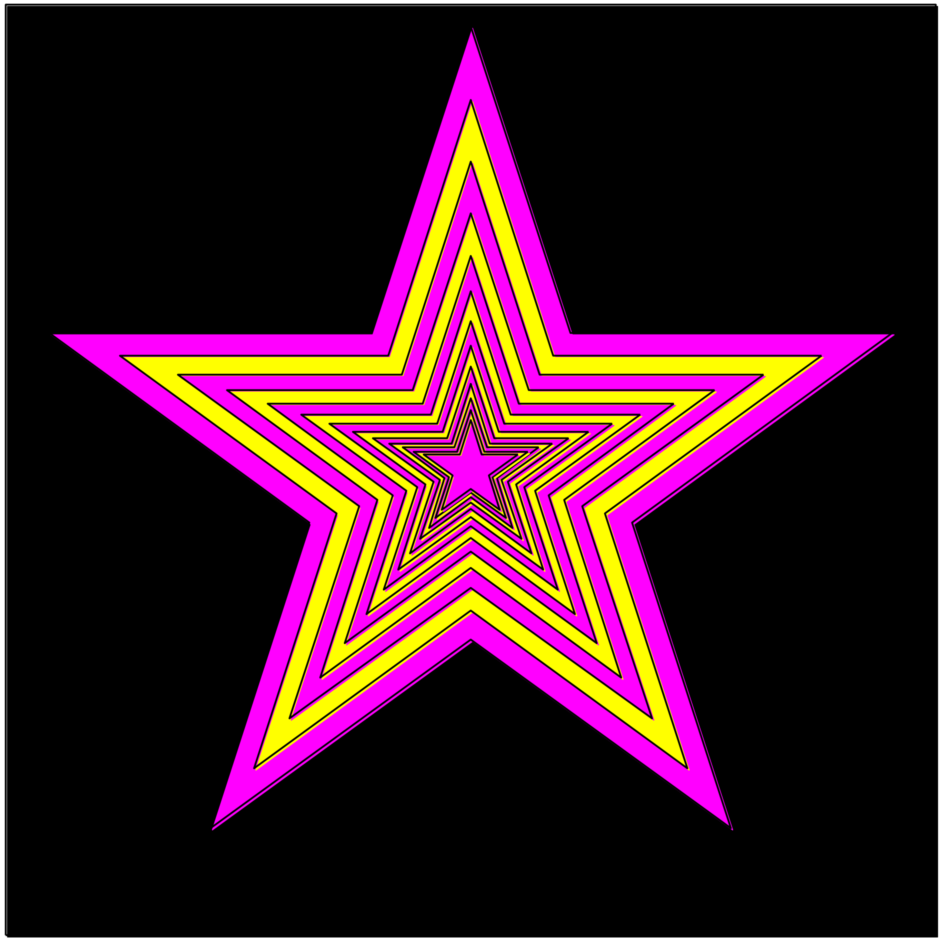




Then apply it four times.

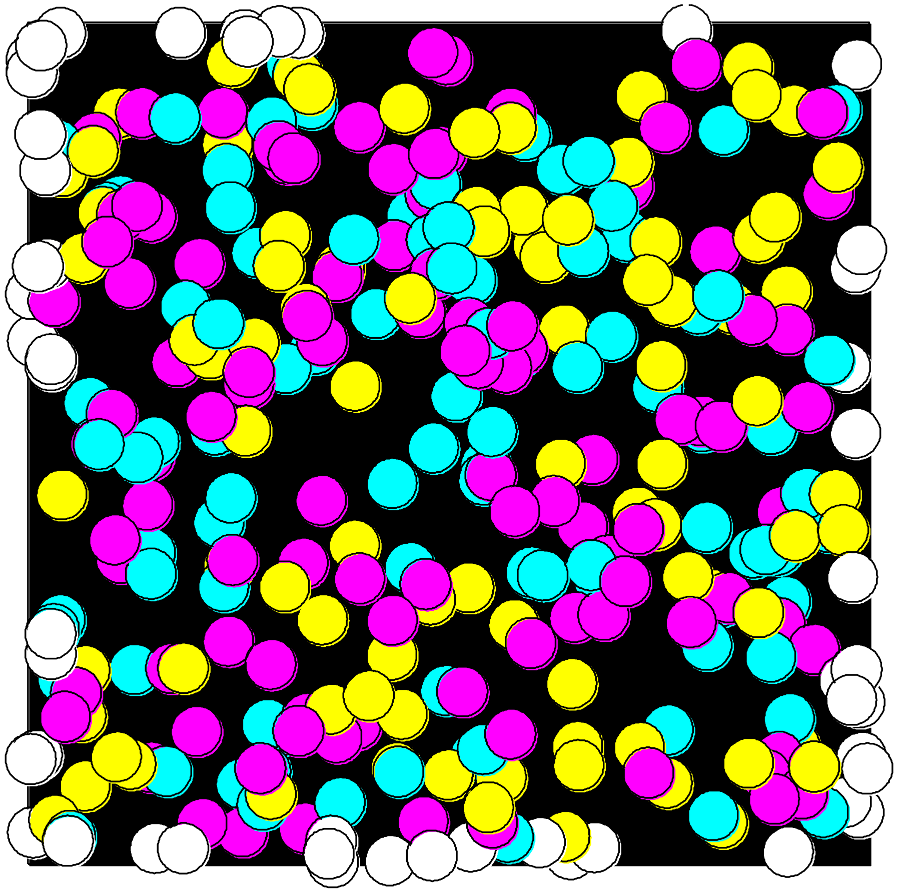
Functions calling other functions.

After We Learn About Iteration…



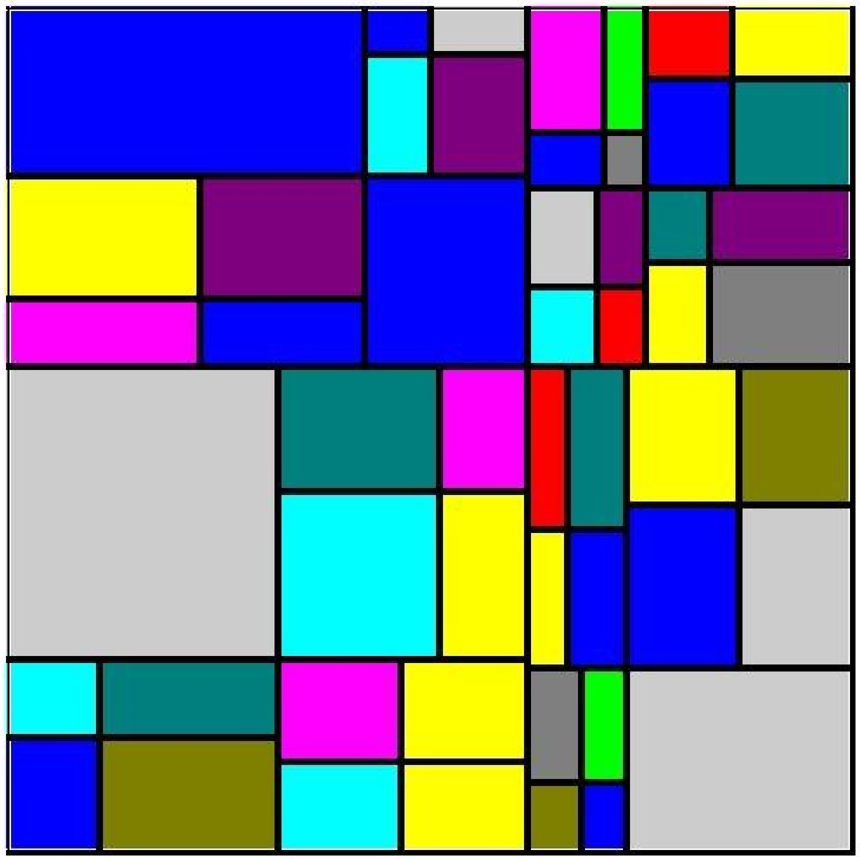
What if there were billions and billions of stars? Will need loops.

After We Learn About Iteration…



How long before the square is covered? Need loops.

After We Learn About Recursion…



Random Mondrian. Repeatedly cut a rectangle into 4 smaller rectangles.

Now lets show how to use the five procedures in **`simpleGraphics`**:

```
MakeWindow
ShowWindow
DrawRect
DrawDisk
DrawStar
```

First: Create a Figure Window

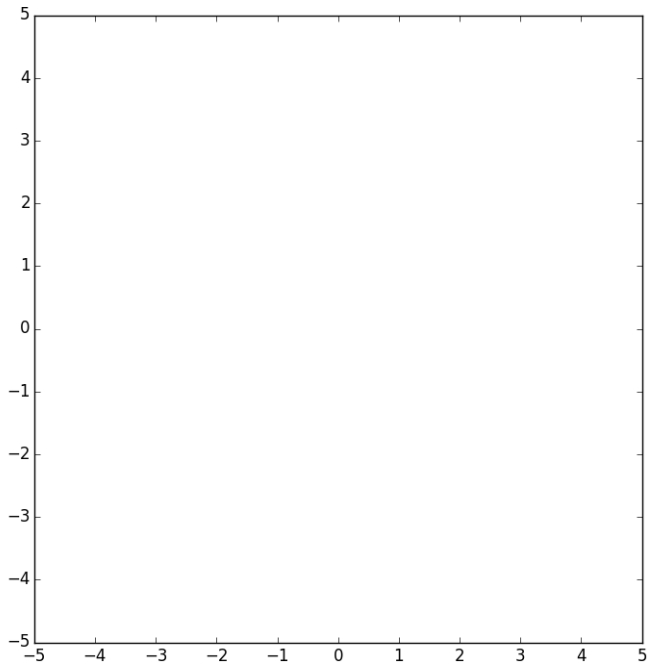
You cannot create any designs until you have a figure into which you can “drop” rectangles, disks, and stars.

MakeWindow

```
from simpleGraphics import *
n = 5
MakeWindow(n)
```

Here we have created
a figure with labeled axes
that is ready to display
things in the square defined
by

-5<=x<=+5, -5<=y<=5

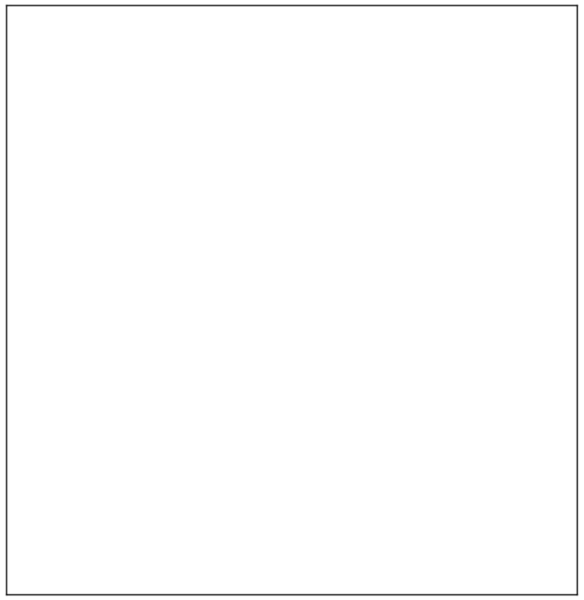


MakeWindow

```
from simpleGraphics import*
n = 5
MakeWindow(n,labels=False)
```

The "default" is to label the axes.

So this is what you must do to suppress the labeling.



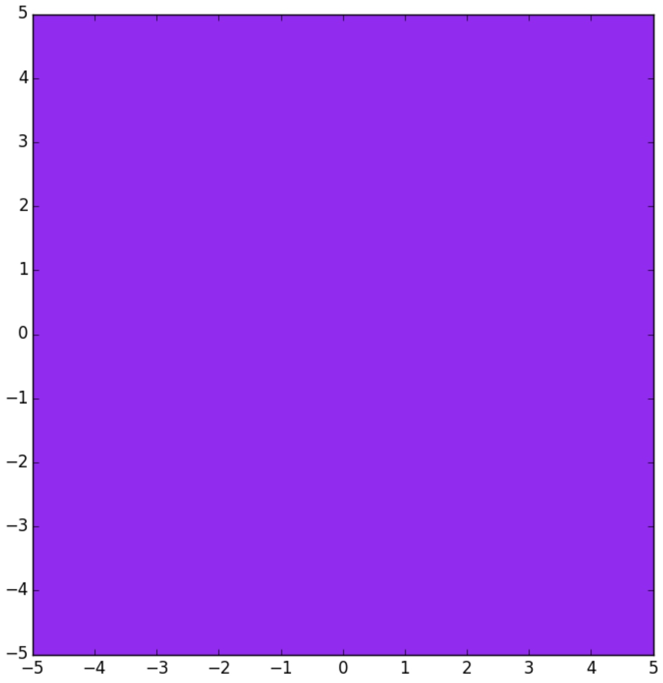
We are using import * to save space and because it is such a tiny module.

MakeWindow

```
from simpleGraphics import*
n = 5
MakeWindow(n,bgcolor=PURPLE)
```

The "default" is to "paint" the figure white.

So this is what you must do to set the background color to something different.



Color in `simpleGraphics`

The module has thirteen "built-in" colors.

If a `simpleGraphics` procedure wants a color, just "hand over" one of these:

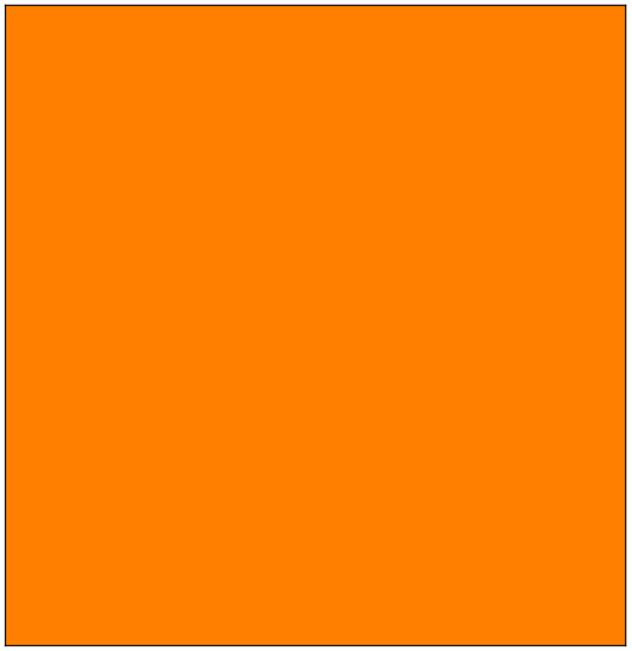
YELLOW	**PURPLE**	**CYAN**	**ORANGE**
RED	**BLUE**	**GREEN**	**MAGENTA**
PINK	**WHITE**	**BLACK**	**LIGHTGRAY**
	DARKGRAY		

There is more flexibility than this. More later.

MakeWindow

```
from simpleGraphics import*
n = 5
MakeWindow(n,labels=False,bgcolor=ORANGE)
```

You can turn off labeling and specify a color in the same call to **MakeWindow**.



Optional Arguments

The function **`MakeWindow`** has three arguments.

Two of the arguments are "optional".

When there are several optional arguments,
Their order is immaterial. Equivalent:

```
MakeWindow(n,labels=False,bgcolor=ORANGE)
MakeWindow(n,bgcolor=ORANGE,labels=False)
```

Note: You need the "assignment" for an optional argument.
This is illegal: **`MakeWindow(5,False,ORANGE)`**

Let's Draw a Rectangle with `DrawRect`

You must tell `DrawRect`

- the center of the rectangle.
- the horizontal dimension of the rectangle
- the vertical dimension of the rectangle

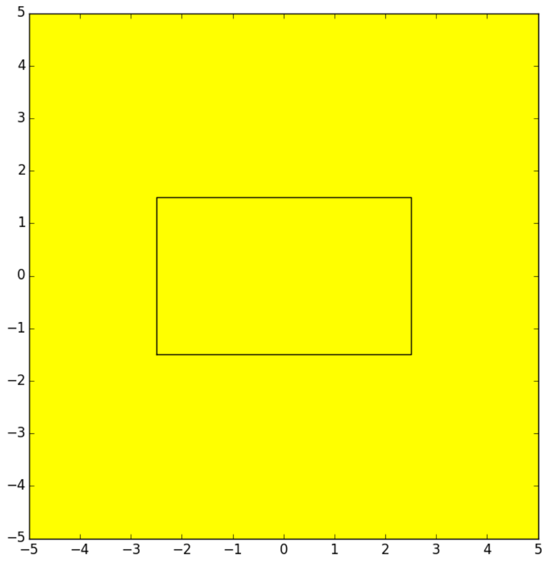
You have the option of telling `DrawRect`

- the fill color
- the width of the perimeter highlight
- the rotation angle

DrawRect

```
from simpleGraphics import*
MakeWindow(5,bgcolor=YELLOW)
x=0; y=0; L=5; W=3
DrawRect(x,y,L,W)
ShowWindow()
```

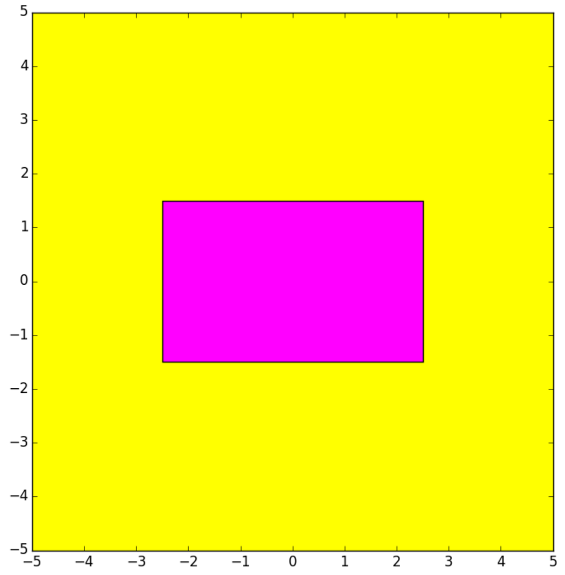
The default is a rectangle with no fill color. So all you get is the perimeter.



DrawRect

```
from simpleGraphics import*
MakeWindow(5,bgcolor=YELLOW)
x=0; y=0; L=5; W=3
DrawRect(x,y,L,W,color=MAGENTA)
ShowWindow()
```

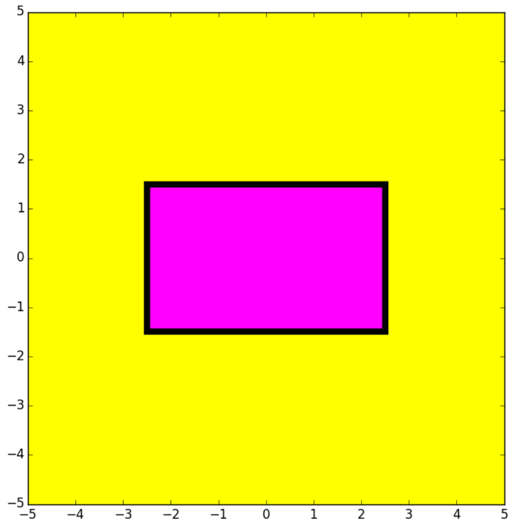
Use the optional color argument to specify a fill color.



DrawRect

```
from simpleGraphics import*
MakeWindow(5,bgcolor=YELLOW)
x=0; y=0; L=5; W=3
DrawRect(x,y,L,W,color=MAGENTA,stroke=6)
ShowWindow()
```

Use the optional **`stroke`** argument to specify the boldness of the perimeter highlight. The default is stroke = 1

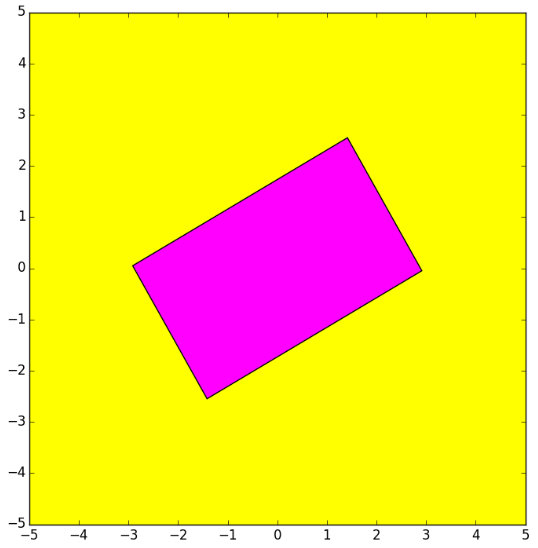


If you don't want any perimeter highlight, set **`stroke=0`**

DrawRect

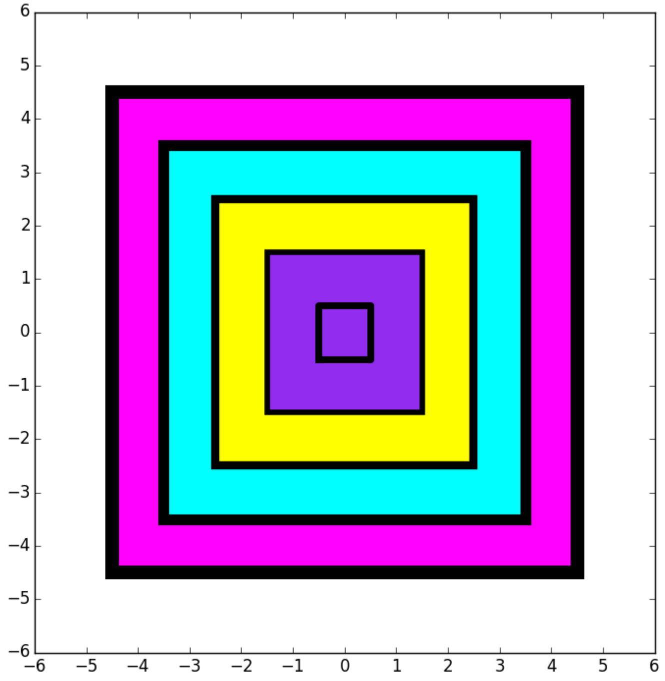
```
from simpleGraphics import*
MakeWindow(5,bgcolor=YELLOW)
x=0; y=0; L=5; W=3
DrawRect(x,y,L,W,color=MAGENTA,rotate=30)
ShowWindow()
```

Use the optional rotate argument to specify the counterclockwise rotation of the rectangle about its center. (Angle in degrees.)



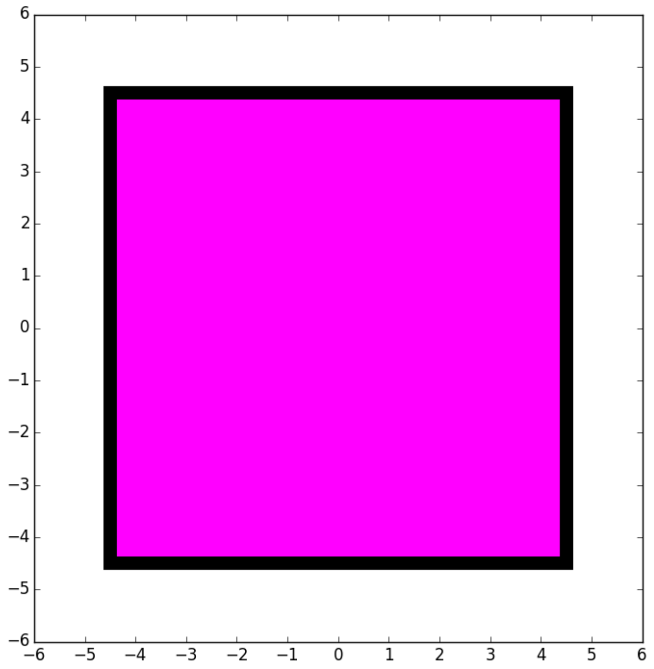
The default rotation angle is zero.

Let's Write a Script to Do This



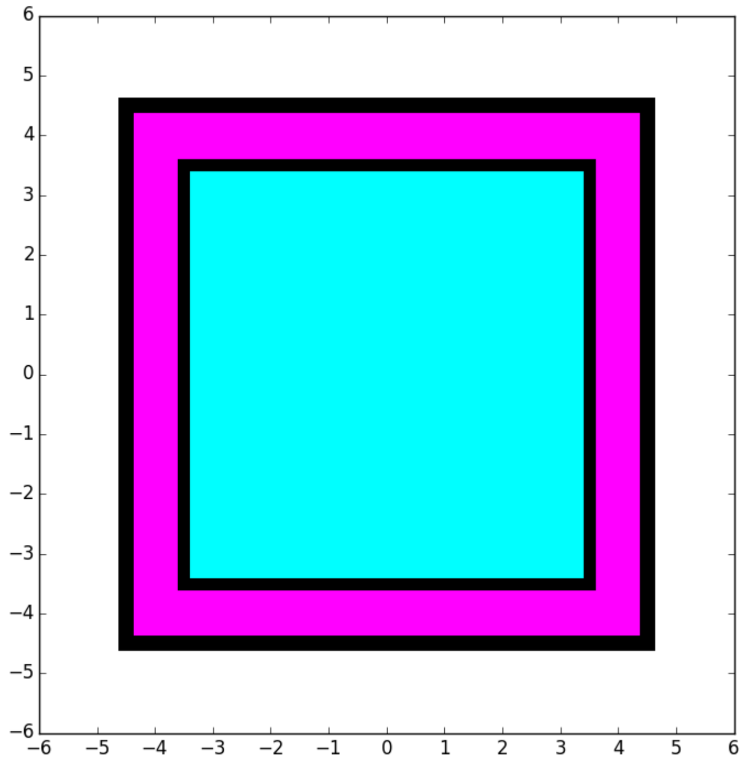
The squares are 9x9, 7x7, 5x5, 3x3, and 1x1.

Nested Squares



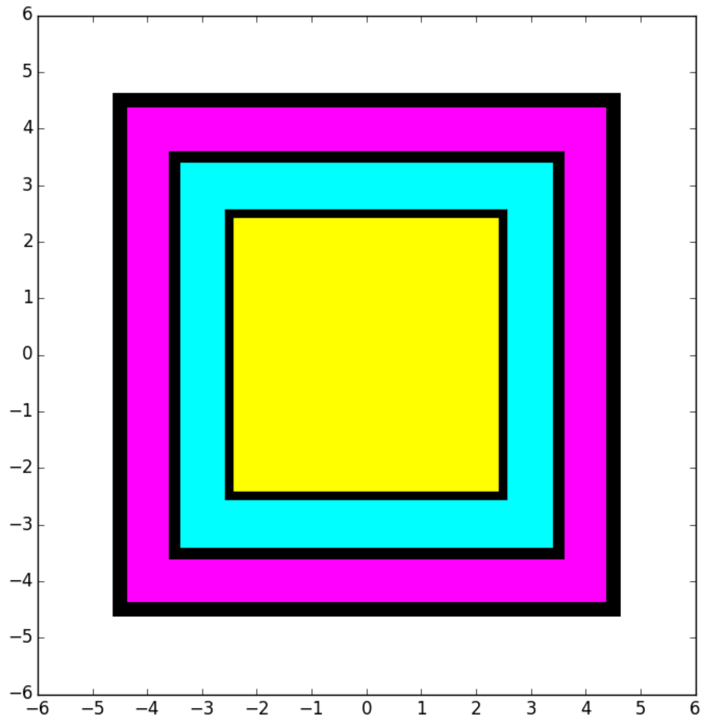
```
DrawRect(0,0,9,9,color=MAGENTA,stroke=10)
```

DrawRect



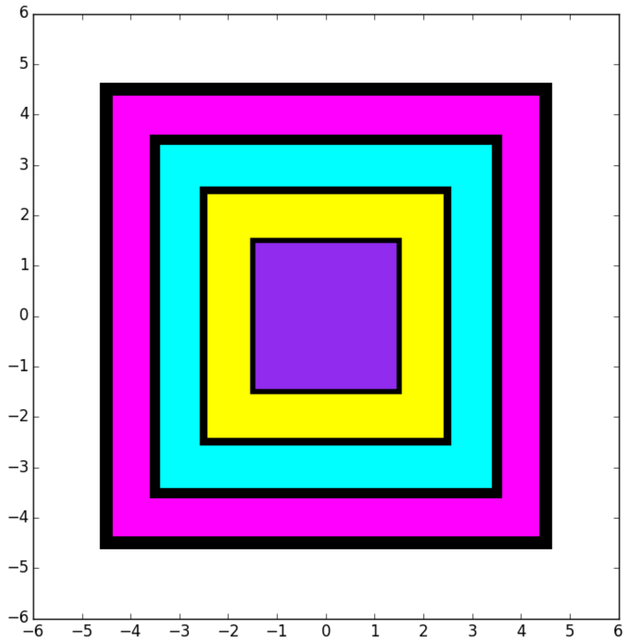
```
DrawRect(0,0,7,7,color=CYAN,stroke=8)
```

Nested Squares



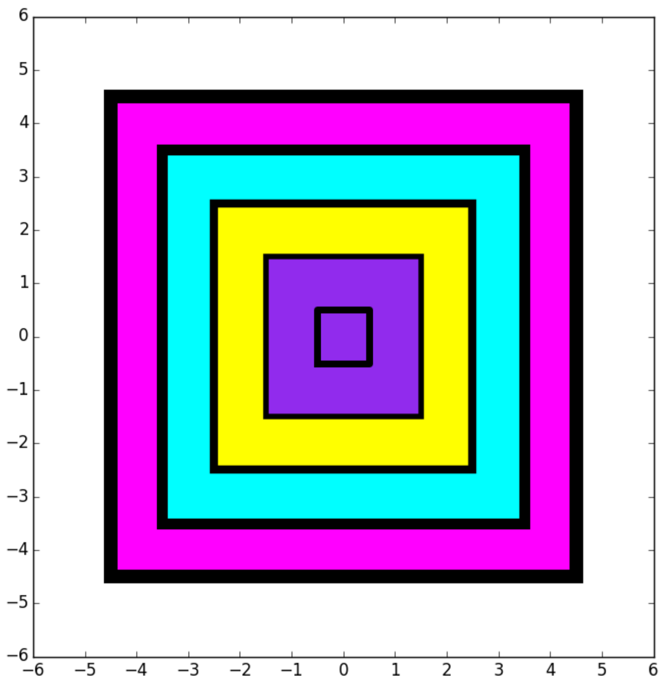
```
DrawRect(0,0,5,5,color=YELLOW,stroke=6)
```

DrawRect



```
DrawRect(0,0,3,3,color=PURPLE,stroke=4)
```

Nested Squares



```
DrawRect(0,0,1,1,stroke=5)
```

Nested Squares

```
MakeWindow(6, bgcolor=WHITE)

DrawRect(0,0,9,9,color=MAGENTA,stroke=10)

DrawRect(0,0,7,7,color=CYAN,stroke=8)

DrawRect(0,0,5,5,color=YELLOW,stroke=6)

DrawRect(0,0,3,3,color=PURPLE,stroke=4)

DrawRect(0,0,1,1,stroke=5)

ShowWindow()
```

Let's Draw a Disk with `DrawDisk`

You must tell **`DrawDisk`**

- the center of the disk.
- the radius of the disk

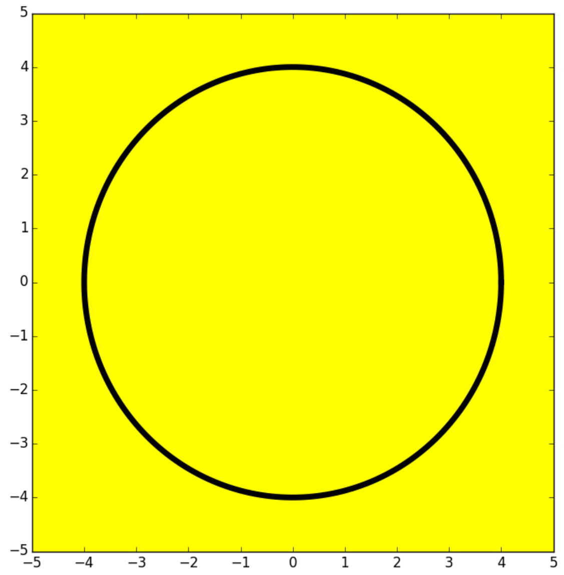
You have the option of telling **`DrawDisk`**

- the fill color
- the width of the perimeter highlight

DrawDisk

```
from simpleGraphics import*
MakeWindow(5,bgcolor=YELLOW)
x=0; y=0; r=4
DrawDisk(x,y,r)
ShowWindow()
```

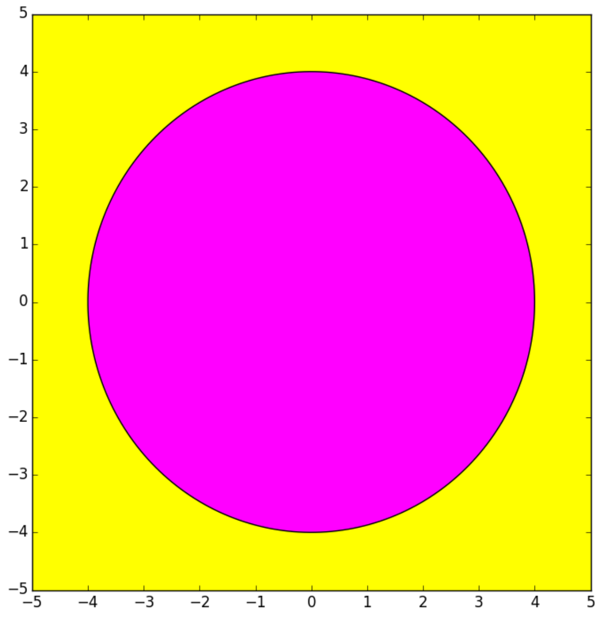
The default is a circle with no fill color. So all you get is the perimeter.



DrawDisk

```
from simpleGraphics import*
MakeWindow(5,bgcolor=YELLOW)
x=0; y=0; r=4
DrawDisk(x,y,r,color=MAGENTA)
ShowWindow()
```

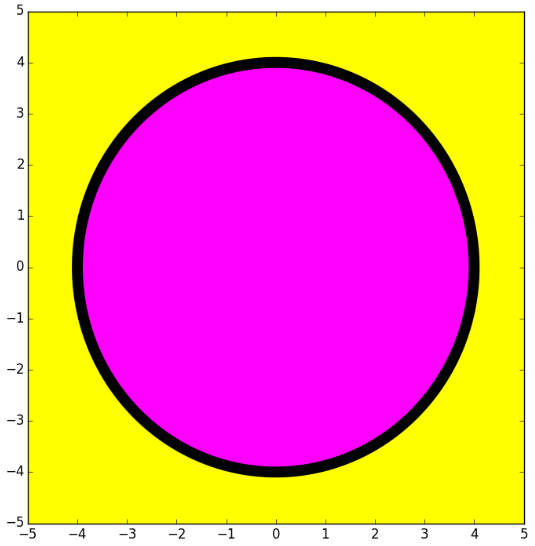
Use the optional color argument to specify a fill color.



DrawDisk

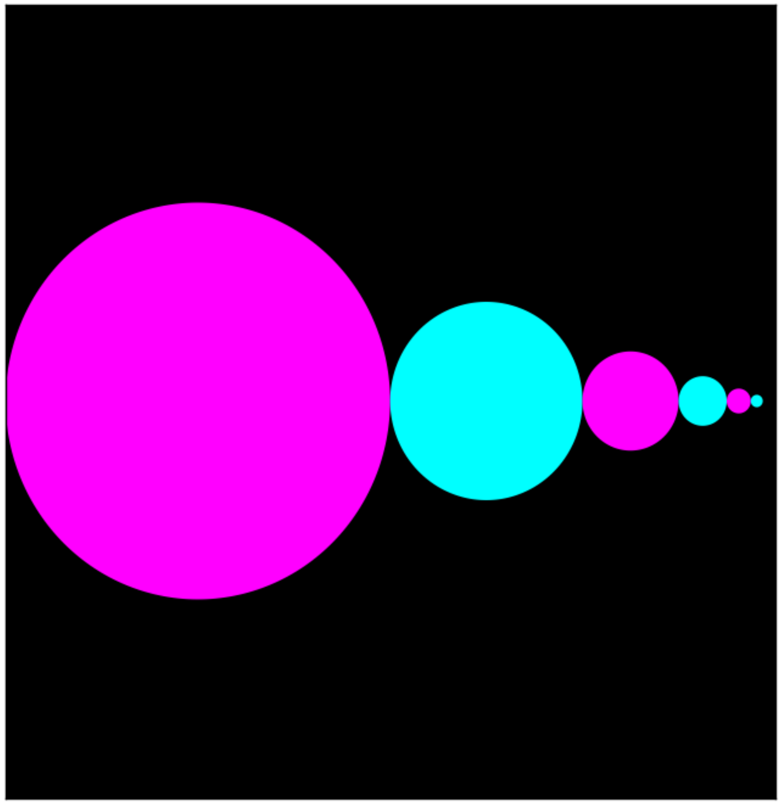
```
from simpleGraphics import*
MakeWindow(5,bgcolor=YELLOW)
x=0; y=0; r=4
DrawDisk(x,y,r,color=MAGENTA,stroke=6)
ShowWindow()
```

Use the optional stroke argument to specify the boldness of the perimeter highlight. The default is stroke = 1



If you don’t want any perimeter highlight, set stroke=0

Let's Draw This



Rules:

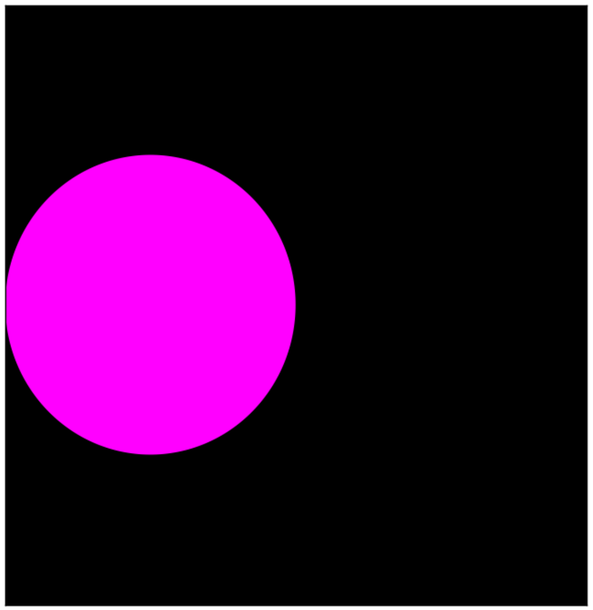
Big circle center at (-4,0) with radius 4.

Circles are tangent to each other. Centers on x-axis.

Each circle has half the radius of its left neighbor.

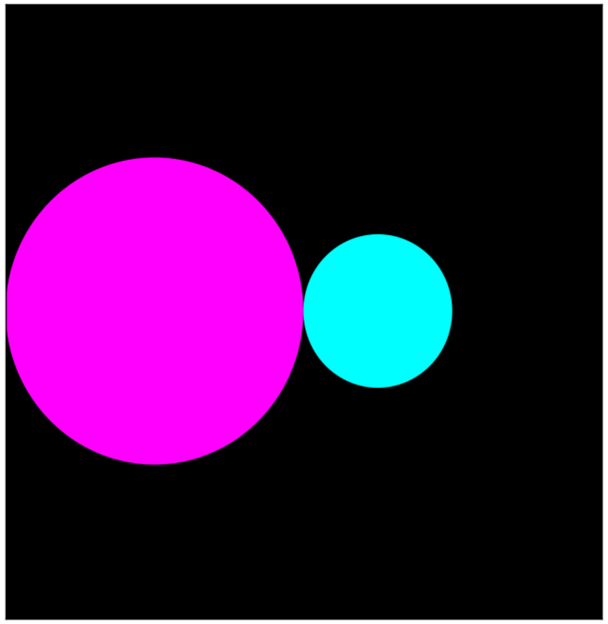
Draw the First Disk

```
x = -4
r = 4
DrawDisk(x,0,r,color=MAGENTA,stroke=0)
```



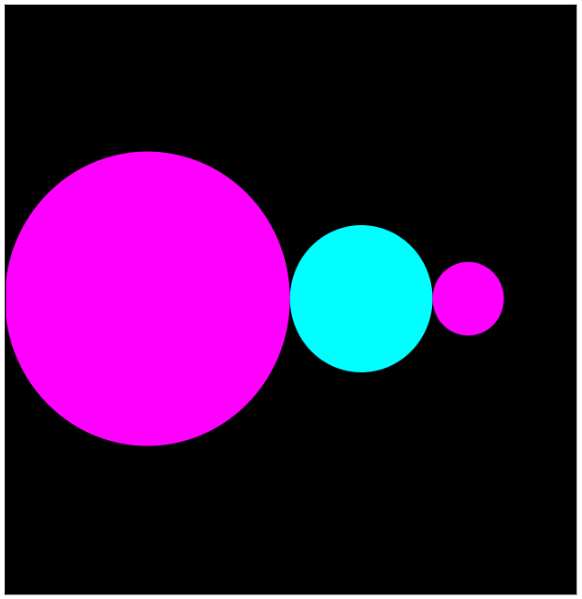
Draw the Second Disk

```
x = x + 1.5*r
r = r/2
DrawDisk(x,0,r,color=CYAN,stroke=0)
```



Draw the Third Disk

```
x = x + 1.5*r
r = r/2
DrawDisk(x,0,r,color=MAGENTA,stroke=0)
```



Overall

```
x = -4; r = 4
DrawDisk(x,0,r,color=MAGENTA,stroke=0)

x = x + 1.5*r; r = r/2
DrawDisk(x,0,r,color=CYAN,stroke=0)

x = x + 1.5*r; r = r/2
DrawDisk(x,0,r,color=MAGENTA,stroke=0)

x = x + 1.5*r; r = r/2
DrawDisk(x,0,r,color=CYAN,stroke=0)
```

Notice the repetition of the x and r updates. Simpler than figuring

Let's Draw a Star with `DrawStar`

You must tell `DrawStar`

- the center of the star.
- the radius of the star

You have the option of telling `DrawStar`

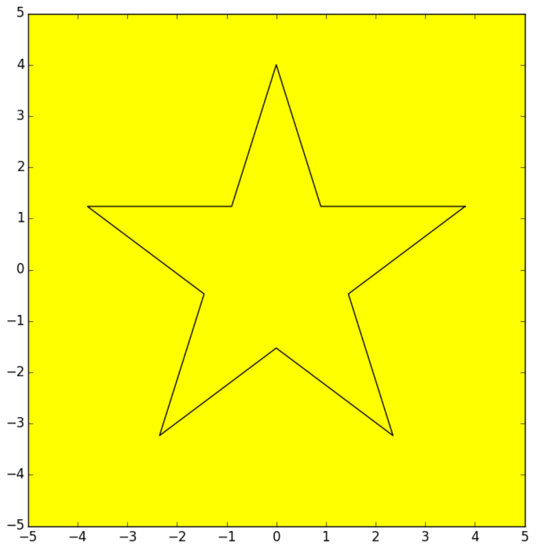
- the fill color
- the width of the perimeter highlight
- the rotation angle

DrawStar

```
from simpleGraphics import*
MakeWindow(5,bgcolor=YELLOW)
x=0; y=0; r=4
DrawStar(x,y,r)
ShowWindow()
```

The default is a star
with no fill color. So all you
get is the perimeter.

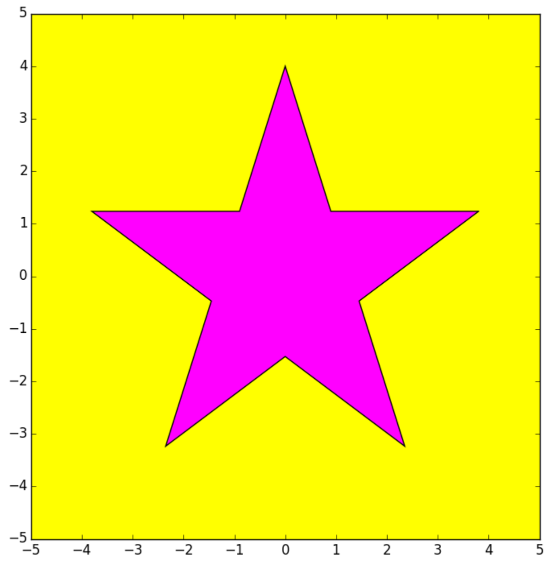
Note: the radius of a star is the
distance from its center to
any tip.



DrawStar

```
from simpleGraphics import*
MakeWindow(5,bgcolor=YELLOW)
x=0; y=0; r=4
DrawStar(x,y,r,color=MAGENTA)
ShowWindow()
```

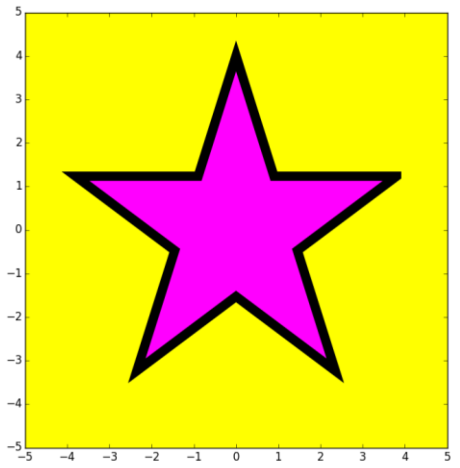
Use the optional color argument to specify a fill color.



DrawStar

```
from simpleGraphics import*
MakeWindow(5,bgcolor=YELLOW)
x=0; y=0; r=4
DrawStar(x,y,r,color=MAGENTA,stroke=6)
ShowWindow()
```

Use the optional stroke argument to specify the boldness of the perimeter highlight. The default is stroke = 1

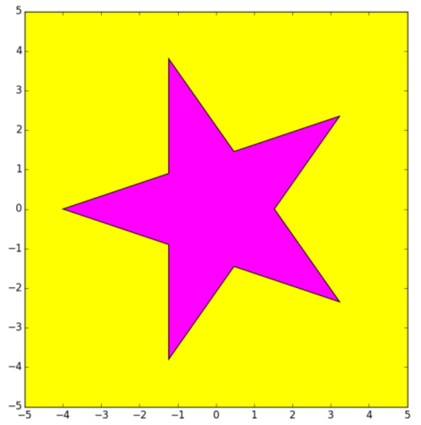


If you don't want any perimeter highlight, set stroke=0

DrawStar

```
from simpleGraphics import*
MakeWindow(5,bgcolor=YELLOW)
x=0; y=0; r=4
DrawStar(x,y,r,color=MAGENTA,rotate=18)
ShowWindow()
```

Use the optional rotate argument to specify the counterclockwise rotation of the rectangle about its center. (Angle in degrees.)



The default rotation angle is zero.

Let's Talk About Color

The rgb Representation

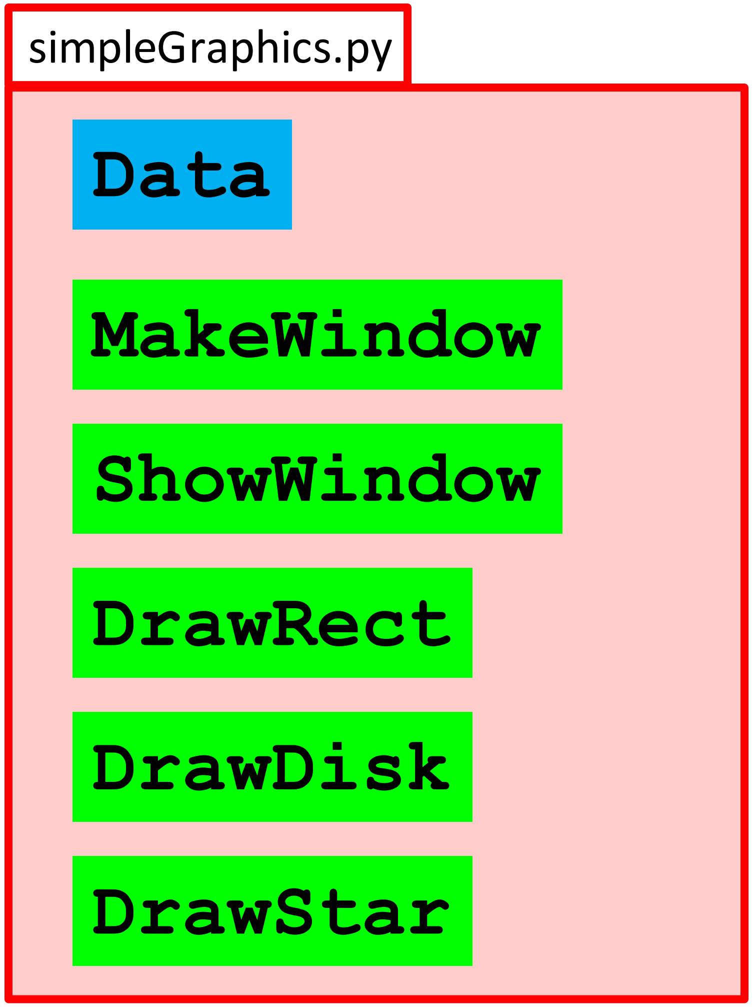
A color is a triple of numbers, each between zero and one.

The numbers represent the amount of red, green, and blue.

This is purple:

```
[ 0.57 , 0.17, 0.93]
```

The Module simpleGraphics Has Five Procedures and Data



In this case the data encodes the rgb values of thirteen colors

The simpleGraphics Colors

```
YELLOW    = [1.00,1.00,0.00]
CYAN      = [0.00,1.00,1.00]
MAGENTA   = [1.00,0.00,1.00]
RED       = [1.00,0.00,0.00]
GREEN     = [0.00,1.00,0.00]
BLUE      = [0.00,0.00,1.00]
WHITE     = [1.00,1.00,1.00]
BLACK     = [0.00,0.00,0.00]
PURPLE    = [0.57,0.17,0.93]
LIGHTGRAY = [0.33,0.33,0.33]
DARKGRAY  = [0.67,0.67,0.67]
ORANGE    = [1.00,0.50,0.00]
PINK      = [1.00,0.71,0.80]
```

These are called "Global Variables"

Convention: Global Variable Names should be upper case.

Access

```
from simpleGraphics import*
MakeWindow(5,bgcolor=YELLOW)
x=0; y=0; L=5; W=3
DrawRect(x,y,L,W,color=MAGENTA)
ShowWindow()
```

When a module is imported, it gives access to both its functions and its global variables.

Take a look at the Demos `ShowRect.py`, `ShowDisk.py`, and `ShowStar.py`

rgb Arrays

Things like `[0.74,1.00,0.34]` are called rgb arrays.

Rule: Square brackets, 3 numbers separated by commas, each number between 0 and 1.

First number = red value
Second number = green value
Third number = blue value

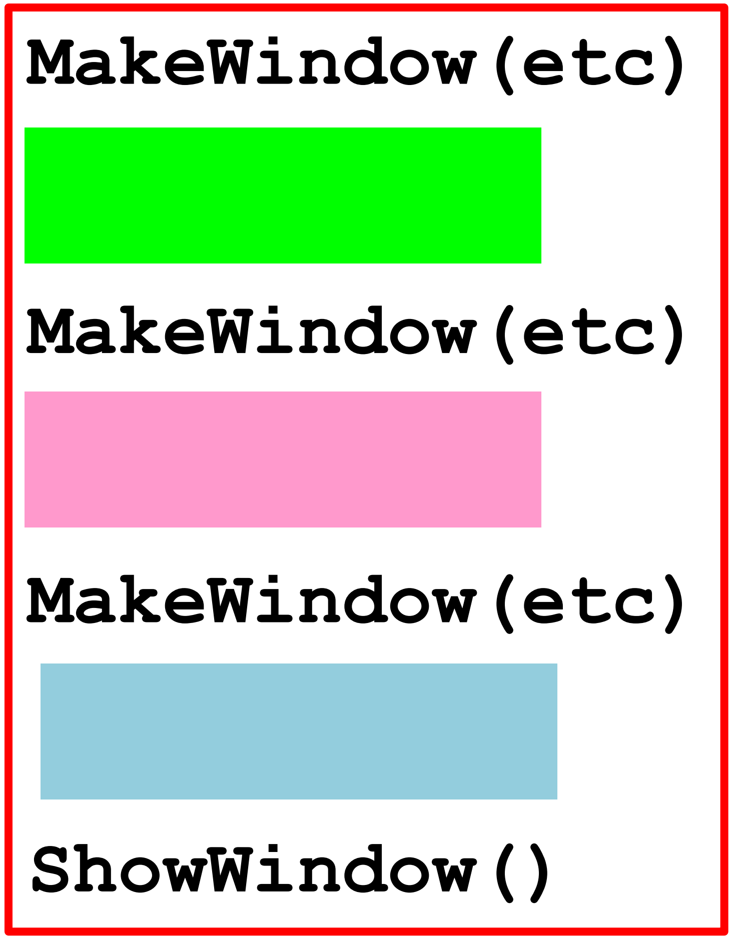
Using rgb Arrays

Instead of using the predfined colors you can make up and use your own fill color, e.g.

```
c = [0.74,1.00,0.34]
DrawDisk(0,0,1,color=c)
```

Google "rgb values" to look at huge tables of colors and rgb values.

A Note on Managing Figures



```
MakeWindow(etc)

MakeWindow(etc)

MakeWindow(etc)

ShowWindow()
```

Three figure windows will be produced.

The green code defines what is in the first window.

The pink and blue code set up the second and third windows.

The **ShowWindow** says. "Show all the windows."

Take a look at the Demos `ShowRect.py` and `ShowStar.py`

A Final Example

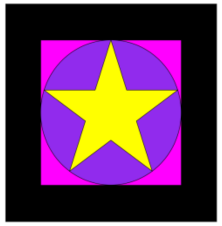
Shows two things.

1. You can write a module that uses other modules that YOU have written.

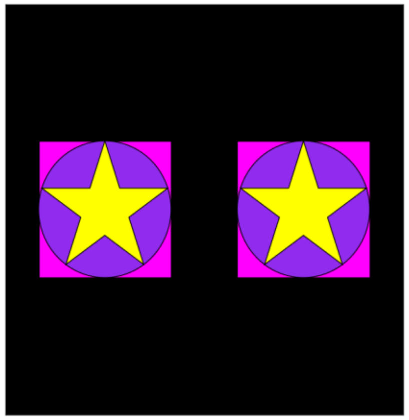
2. You can have a module that has both function definitions and a script.

A Final Example

We write a procedure to draw this



and a script that calls it twice:



and we put them both in the SAME module....

A Final Example

Tile.py

```
from simpleGraphics import *

def DrawTile(x,y,r,c1,c2,c3):
    DrawRect(x,y,2*r,2*r,color=c1)
    DrawDisk(x,y,r,color=c2)
    DrawStar(x,y,r,color=c3)

if __name__ == '__main__':

    MakeWindow(6,bgcolor=BLACK,labels=False)
    DrawTile(3,0,2,MAGENTA,PURPLE,YELLOW)
    DrawTile(-3,0,2,MAGENTA,PURPLE,YELLOW)
    ShowWindow()
```

See the Demo `Tile.py`   In command mode, enter `python Tile.py`

A Final Example

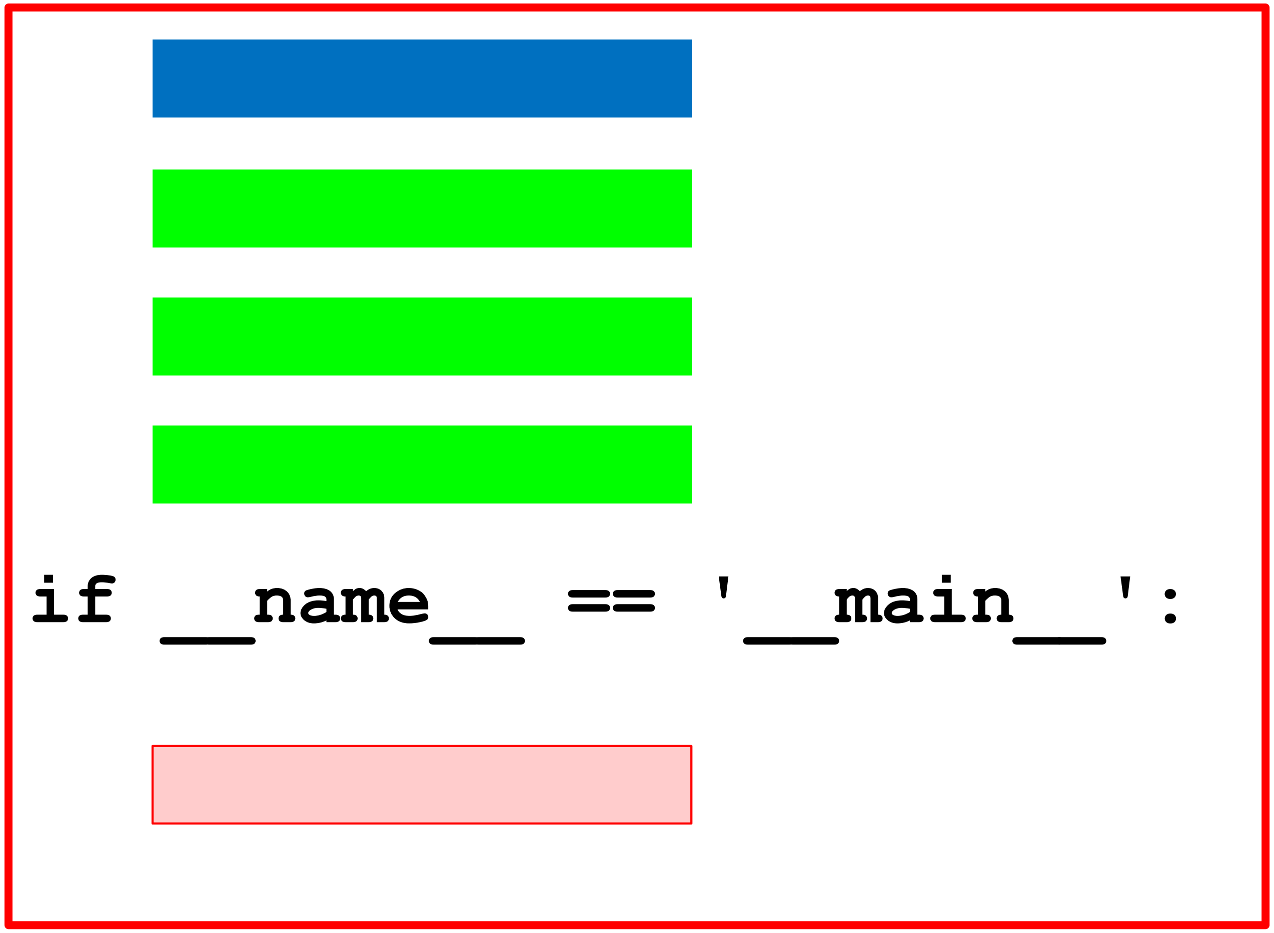
Tile.py

```
from simpleGraphics import *

def DrawTile(x,y,r,c1,c2,c3):
    DrawRect(x,y,2*r,2*r,color=c1)
    DrawDisk(x,y,r,color=c2)
    DrawStar(x,y,r,color=c3)

if __name__ == '__main__':

    MakeWindow(6,bgcolor=BLACK,labels=False)
    DrawTile(3,0,2,MAGENTA,PURPLE,YELLOW)
    DrawTile(-3,0,2,MAGENTA,PURPLE,YELLOW)
    ShowWindow()
```

This is called an “Application Script”

See the Demo `Tile.py` In command mode, enter `python Tile.py`

So a Module Can Look Like This



```
if __name__ == '__main__':
```

Data

Function Definitions

Gotta have

Application Script

Those are "double underscores" in the if statement.

Summary

1. Procedures “look like” functions without the “return.” They “do stuff” but do not return values
2. Graphics procedures were used to illustrate the idea.
3. Color can be encoded with three numbers that indicate the amount of red, green, and blue.
4. A single module can house data, functions, and a script at the same time

Terminology

procedure

A function that has no explicit return statements that yield a value. A function call on a procedure always evaluates to None.

Reference: http://www.cs.cornell.edu/Courses/cs1110/2015sp/materials/definitions.php

Terminology

script

A program that contains a segment of code like this: **if** __name__ == "__main__":

Scripts can be run outside of the interactive mode. To run a script, type python *<application name>* at the OS command shell. When a script is run, it will execute all of the code indented under the if-statement above.

Reference: http://www.cs.cornell.edu/Courses/cs1110/2015sp/materials/definitions.php