

Exposure CS 2021 **for CS1**

Chapter 8 Section 7-9 Slides

**Creating Custom Colors and Repetition
with Random Numbers and Graphics**

**PowerPoint Presentation
created by:
Mr. John L. M. Schram
and Mr. Leon Schram
Authors of Exposure
Computer Science**



Section 8.7

creating

custom colors

```
1 # CustomColors01.py
2 # This program displays the Texas flag using the
3 # built-in colors for red, white and blue. While
4 # this certainly looks like the Texas flag, it does
5 # not use the official shades of red and blue that
6 # are used in the Official Texas Flag.
7
8
9 from Graphics import *
10
11 beginGrafX(1300,700)
12
13 setColor("blue")
14 fillRectangle(0,0,350,700)
15 setColor("red")
16 fillRectangle(350,350,1300,700)
17 setColor("white")
18 fillStar(175,350,130,5)
19
20 endGrafX()
```

```
1 # CustomCo
2 # This pro
3 # built-in
4 # this cer
5 # not use
6 # are used
```

```
7
8
9 from Graph
```

```
10
11 beginGrfx(
```

```
12
13 setColor("blue")
```

```
14 fillRectangle(0,0,350,700)
```

```
15 setColor("red")
```

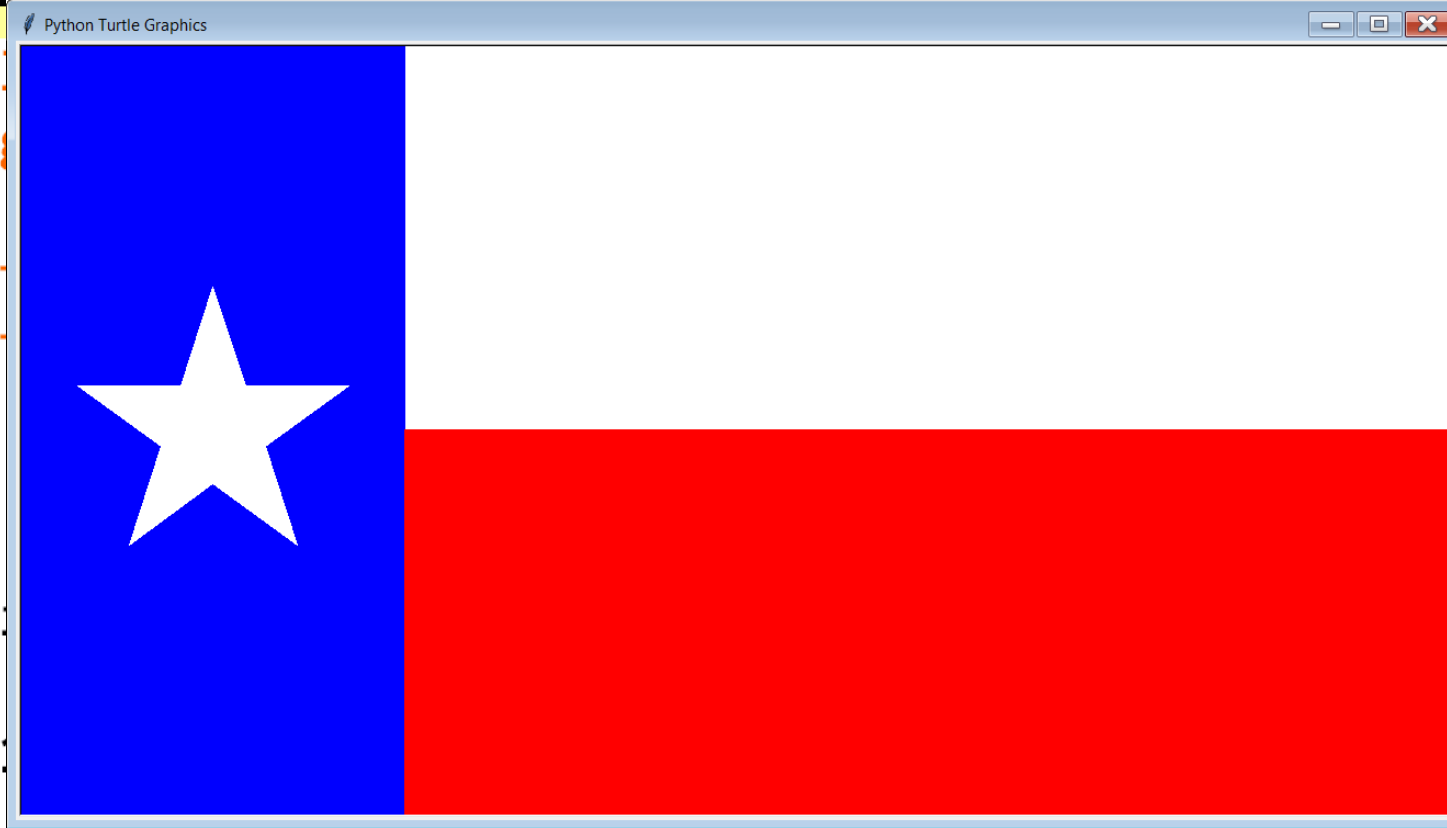
```
16 fillRectangle(350,350,1300,700)
```

```
17 setColor("white")
```

```
18 fillStar(175,350,130,5)
```

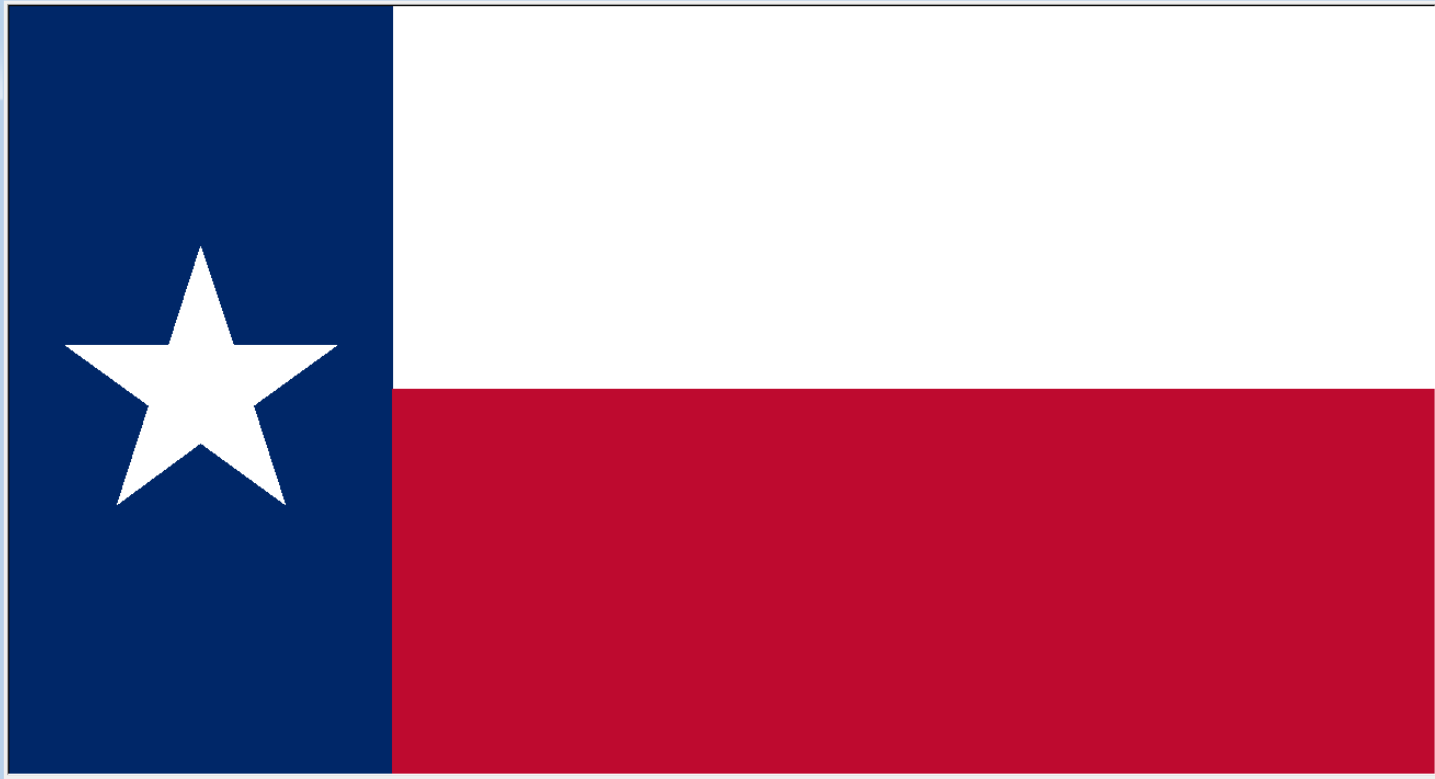
```
19
```

```
20 endGrfx()
```

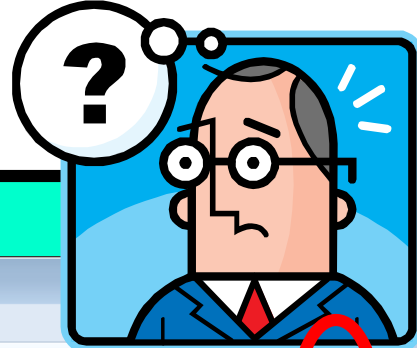


```
1 # CustomColors02.py
2 # This program demonstrates the <setColor> procedure
3 # being used to "create" custom colors. The program
4 # will draw the Official Texas Flag with the EXACT
5 # official shades of red and blue required.
6
7
8 from Graphics import *
9
10 beginGrfx(1300,700)
11
12 setColor(0,39,104)      # Texas Flag blue
13 fillRectangle(0,0,350,700)
14 setColor(190,10,47)     # Texas Flag red
15 fillRectangle(350,350,1300,700)
16 setColor(255,255,255)   # Three 255s = "white"
17 fillStar(175,350,130,5)
18
19 endGrfx()
```

```
1 # CustomCo
2 # This pro
3 # being us
4 # will dra
5 # official
6
7
8 from Graph
9
10 beginGrfx(
11
12 setColor(0,39,104)      # Texas Flag blue
13 fillRectangle(0,0,350,700)
14 setColor(190,10,47)    # Texas Flag red
15 fillRectangle(350,350,1300,700)
16 setColor(255,255,255)  # Three 255s = "white"
17 fillStar(175,350,130,5)
18
19 endGrfx()
```



Where do you get the 3 color #s

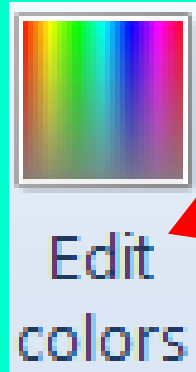


I started by downloading an image of the Official Texas Flag.
I then loaded it in MS Paint.

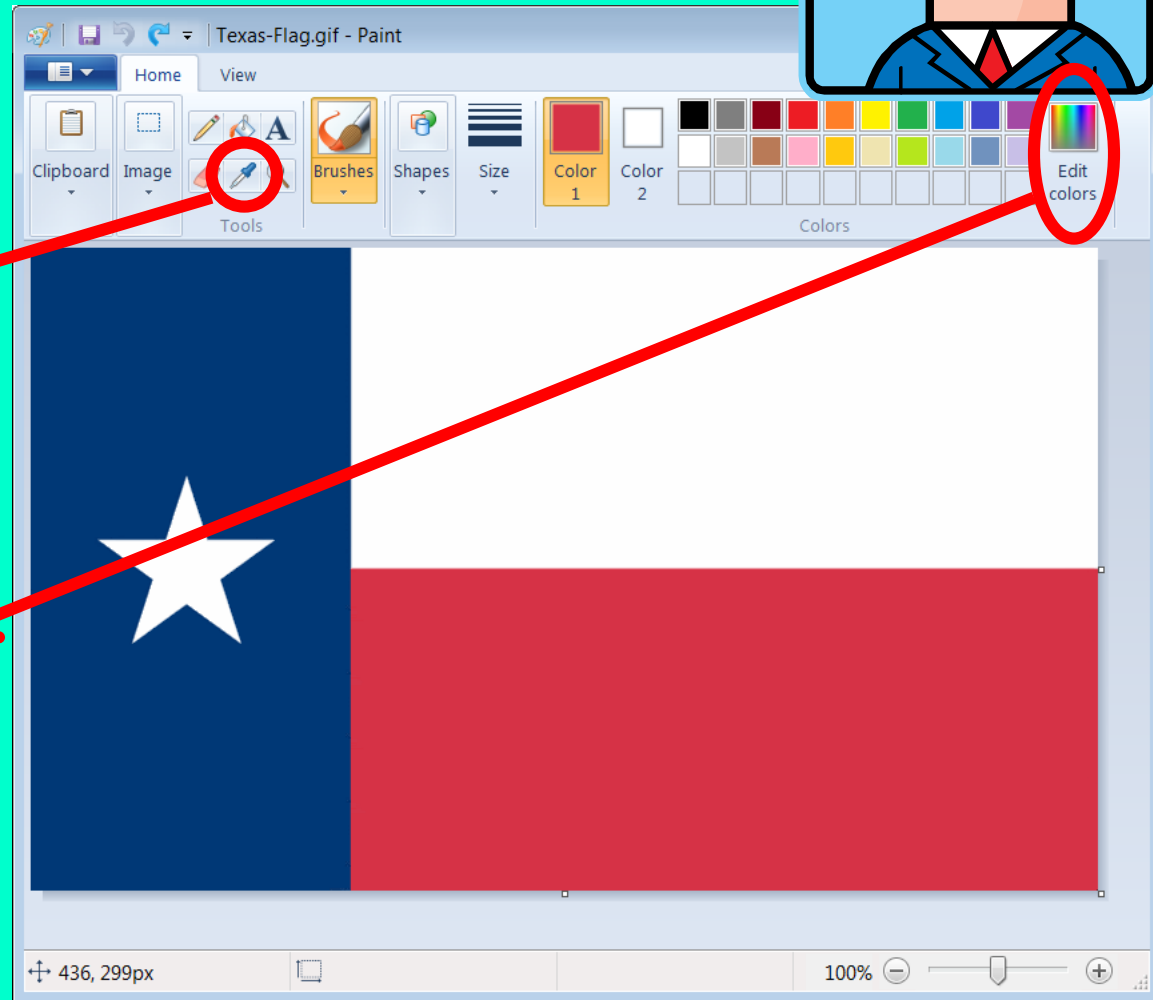
Using the eye-dropper tool,
I then clicked on
the desired color.



I then clicked
[Edit Colors] and
made note of the **red**,
green & **blue** values.



Red: 0
Green: 39
Blue: 104



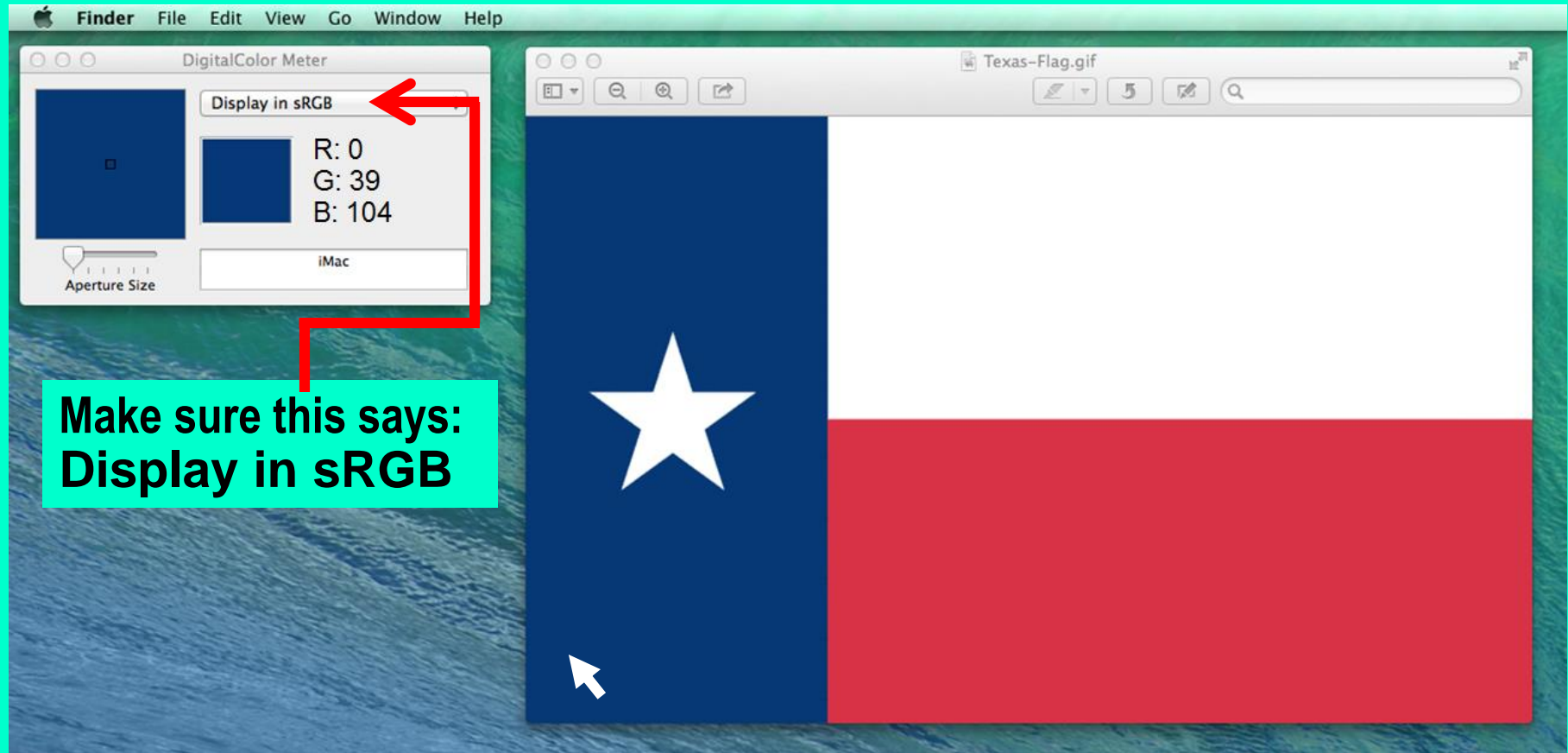
By using the same 3 numbers in my program, I get the exact same color in
the output of my Python program:

```
setColor(0,39,104)
```


You can use DigitalColor Meter on a Mac

Double-click the image; which will open it up in a Preview window.

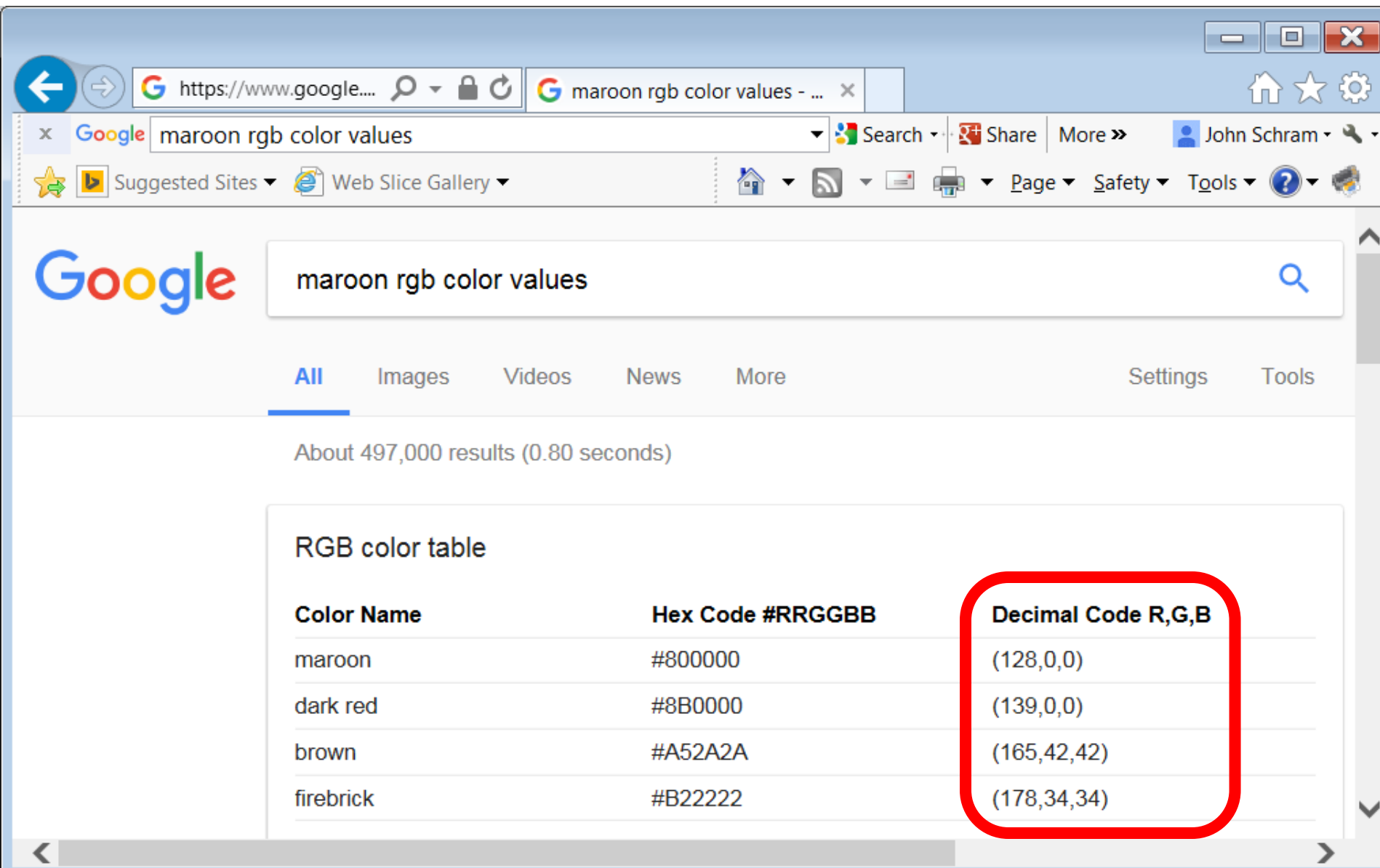
Open DigitalColor Meter. (You can do a search for it.)



Make sure this says:
Display in sRGB

Move your mouse pointer inside the preview window over the desired color. The DigitalColor Meter will show you the **red**, green and **blue** values.

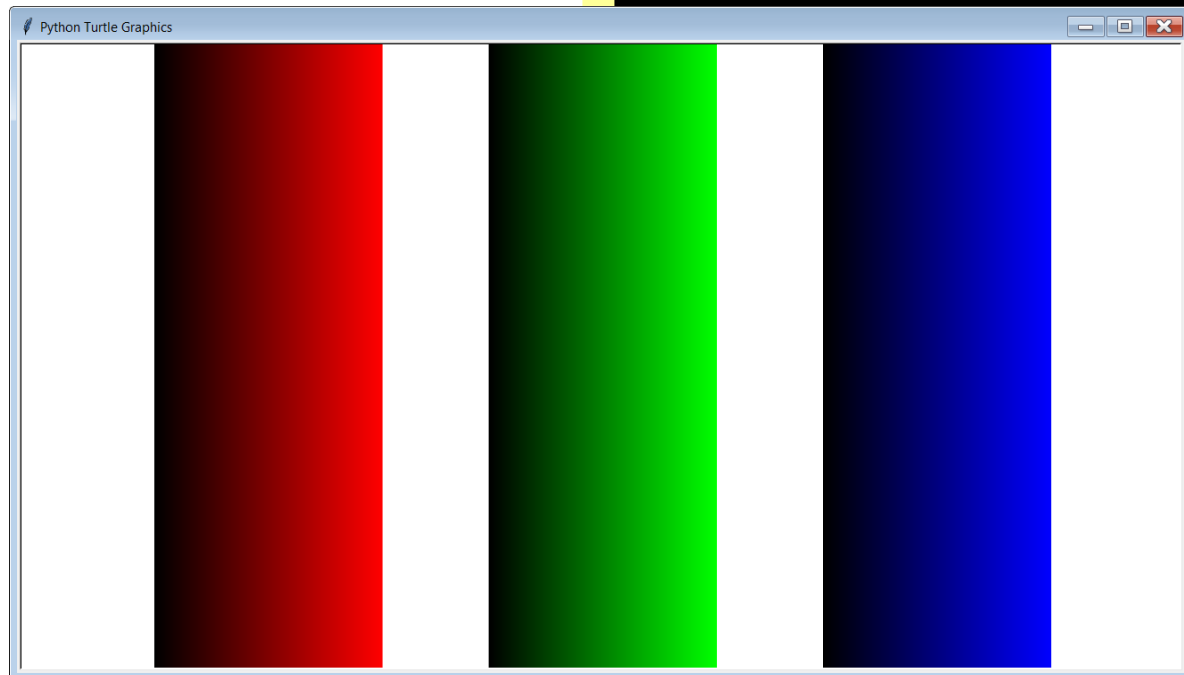
You can also just do a **Google** search for most RGB color values.



A screenshot of a Google search results page. The search query is "maroon rgb color values". The results show a table titled "RGB color table" with three columns: "Color Name", "Hex Code #RRGGBB", and "Decimal Code R,G,B". The table lists four colors: maroon, dark red, brown, and firebrick. The "Decimal Code R,G,B" column is highlighted with a red rounded rectangle.

| Color Name | Hex Code #RRGGBB | Decimal Code R,G,B |
|------------|------------------|--------------------|
| maroon | #800000 | (128,0,0) |
| dark red | #8B0000 | (139,0,0) |
| brown | #A52A2A | (165,42,42) |
| firebrick | #B22222 | (178,34,34) |

```
1 # CustomColors03.py
2 # This program uses the <setColor> procedure
3 # to show 256 shades of red, green and blue.
4 # This creates a "shading" effect which
5 # results in an illusion of depth.
6
7
8 from Graphics import *
9
10 beginGrfx(1300,700)
11
12 x = 150;
13 for red in range(256):
14     setColor(red,0,0)
15     drawLine(x,0,x,700)
16     x += 1
17
18 x = 525;
19 for green in range(256):
20     setColor(0,green,0)
21     drawLine(x,0,x,700)
22     x += 1
23
24 x = 900;
25 for blue in range(256):
26     setColor(0,0,blue)
27     drawLine(x,0,x,700)
28     x += 1
29
30 endGrfx()
```





Section 8.8

creating

Random

Numbers

```
1 # RandomNumbers01.py
2 # This program uses the <randint> function from
3 # the <random> library 5 times to create 5 random
4 # integers between 1 and 100.
5
6 # NOTE: When you execute the program a second time,
7 #         you will get a different set of numbers
8 #         because they are "random".
9
10
11 # Required to have access the <randint> command
12 from random import randint
13
14 print()
15 print(randint(1,100))
16 print(randint(1,100))
17 print(randint(1,100))
18 print(randint(1,100))
19 print(randint(1,100))
```

**Since we are working with
RANDOM numbers,
your outputs
will be different than mine –
and also different from the
other students in the class.**

```
10
11 # Required to have access the <randir
12 from random import randint
13
14 print()
15 print(randint(1,100))
16 print(randint(1,100))
17 print(randint(1,100))
18 print(randint(1,100))
19 print(randint(1,100))
```

----jGRASP

47
66
17
6
97

----jGRASP:

----jGRASP

26
12
20
57
52

----jGRASP:

```
1 # RandomNumbers02.py
2 # This program is more efficient and flexible than the
3 # previous program. It is more efficient because the
4 # 5 <print> commands are now in a <for> loop.
5 # It is more flexible because the user can specify
6 # the range of random numbers
7
8
9 from random import randint
10
11 print()
12 min = eval(input("Enter the smallest number. --> "))
13 max = eval(input("Enter the largest number. --> "))
14 print()
15
16 for k in range(5):
17     print(randint(min,max))
18
```



```
----jGRASP exec: python RandomNumbers02.py
```

```
Enter the smallest number.  --> 10
```

```
Enter the largest number.  --> 99
```

```
85
```

```
16
```

```
18
```

```
50
```

```
88
```

```
----jGRASP: operation complete.
```

```
----jGRASP exec: python RandomNumbers02.py
```

```
Enter the smallest number.  --> 1000
```

```
Enter the largest number.  --> 9999
```

```
4356
```

```
7949
```

```
9183
```

```
6040
```

```
6199
```

```
----jGRASP: operation complete.
```

Simulations

Random numbers are used extensively in computer simulations.

The next 2 programs deal with a simple simulation of rolling 2 dice 1,000,000 times.



This same concept is also used in video games, military software statistics and cryptography.



```
1 # RandomNumbers03.py
2 # This program INCORRECTLY simulates rolling
3 # dice 1,000,000 times.
4
5
6 from random import randint
7
8 sevens = 0
9 elevens = 0
10 snakeEyes = 0
11 doubles = 0
12
13 for roll in range(1000000):
14     dice = randint(2,12)
15     #
16     if dice == 2:
17         snakeEyes += 1
18     #
19     if dice == 7:
20         sevens += 1
21     #
22     if dice == 11:
23         elevens += 1
24
25 print()
26 print("# of Sevens:      ",sevens)
27 print("# of Elevens:      ",elevens)
28 print("# of Snake Eyes:",snakeEyes)
29 print("# of Doubles:      ",doubles)
```

```
1 # RandomNumbers03.py
2 # This program INCORRECTLY simulates rolling
3 # dice 1,000,000 times.
```

```
4
5
6 from random import randint
7
8 sevens = 0
9 elevens = 0
10 snakeEyes = 0
11 doubles = 0
12
13 for roll in range(1000000):
14     dice = randint(2,12)
15     #
16     if dice == 2:
17         snakeEyes += 1
18     #
19     if dice == 7:
20         sevens += 1
21     #
22     if dice == 11:
23         elevens += 1
24
25 print()
26 print("# of Sevens:      ",sevens)
27 print("# of Elevens:      ",elevens)
28 print("# of Snake Eyes:",snakeEyes)
29 print("# of Doubles:      ",doubles)
```

```
----jGRASP exec: python


# of Sevens:      91003
# of Elevens:      90583
# of Snake Eyes:  91183
# of Doubles:      0

----jGRASP: operation c
```



What else is wrong?



Aside from the *double* issue the simulation seems OK. It may seem logical to think that the number of 2s, 7s, and 11s rolled would be about the same.  However, this is not true. Look at the chart below:

| Dice | 1 | 2 | 3 | 4 | 5 | 6 |
|------|---|---|---|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |

```
1 # RandomNumbers04.py
2 # This program PROPERLY simulates rolling
3 # dice 1,000,000 times.
4
5
6 from random import randint
7
8 sevens = 0
9 elevens = 0
10 snakeEyes = 0
11 doubles = 0
12
13 for roll in range(1000000):
14     die1 = randint(1,6)
15     die2 = randint(1,6)
16     diceTotal = die1 + die2
17     #
18     if diceTotal == 2:
19         snakeEyes += 1
20     if diceTotal == 7:
21         sevens += 1
22     if diceTotal == 11:
23         elevens += 1
24     if die1 == die2:
25         doubles += 1
26
27 print()
28 print("# of Sevens:      ",sevens)
29 print("# of Elevens:     ",elevens)
30 print("# of Snake Eyes:",snakeEyes)
31 print("# of Doubles:      ",doubles)
```

```

1 # RandomNumbers04.py
2 # This program PROPERLY simu
3 # dice 1,000,000 times.
4
5
6 from random import randint
7
8 sevens = 0
9 elevens = 0
10 snakeEyes = 0
11 doubles = 0
12
13 for roll in range(1000000):
14     die1 = randint(1,6)
15     die2 = randint(1,6)
16     diceTotal = die1 + die2
17     #
18     if diceTotal == 2:
19         snakeEyes += 1
20     if diceTotal == 7:
21         sevens += 1
22     if diceTotal == 11:
23         elevens += 1
24     if die1 == die2:
25         doubles += 1
26
27 print()
28 print("# of Sevens: ",sevens)
29 print("# of Elevens: ",elevens)
30 print("# of Snake Eyes:",snakeEyes)
31 print("# of Doubles: ",doubles)

```

```

----jGRASP exec: python

```

```

# of Sevens:      166874
# of Elevens:     55185
# of Snake Eyes:  27817
# of Doubles:     166205

```

```

----jGRASP: operation c

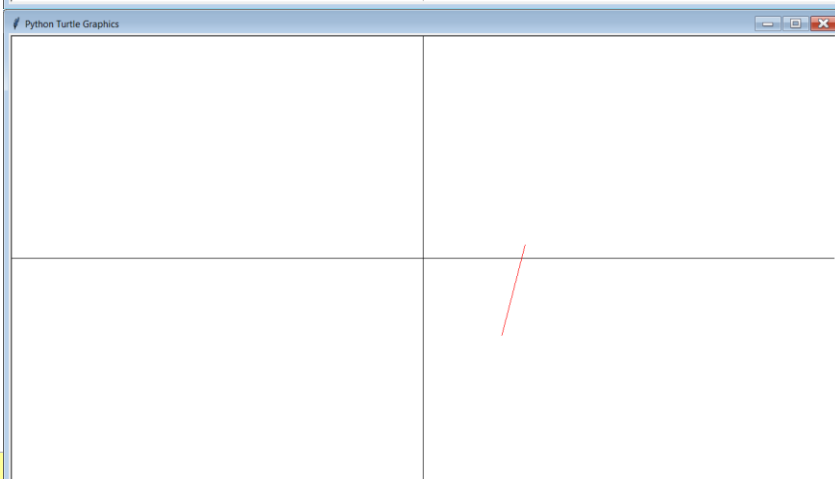
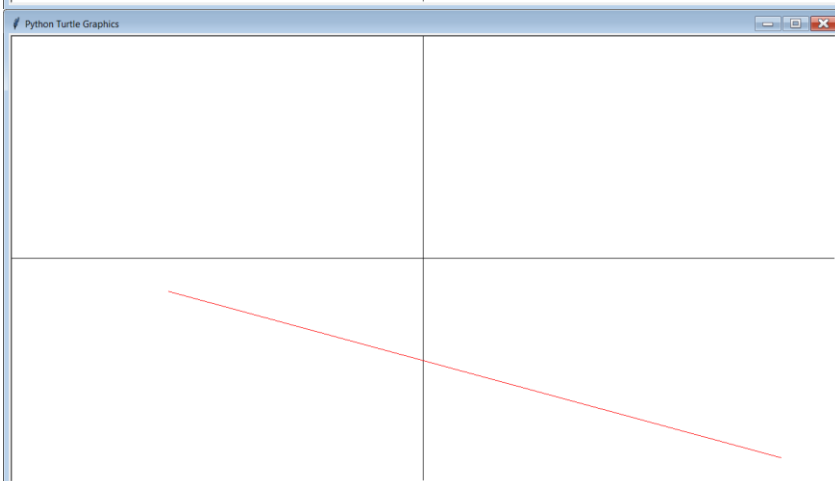
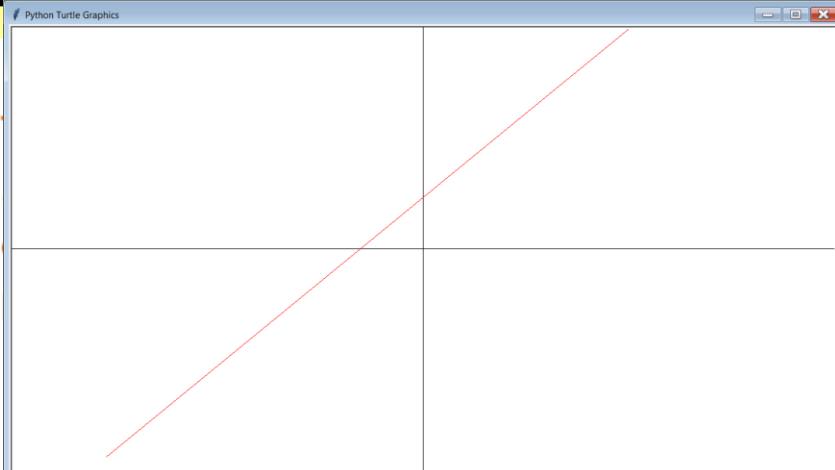
```


Section 8.9

Using Random #s with Graphics

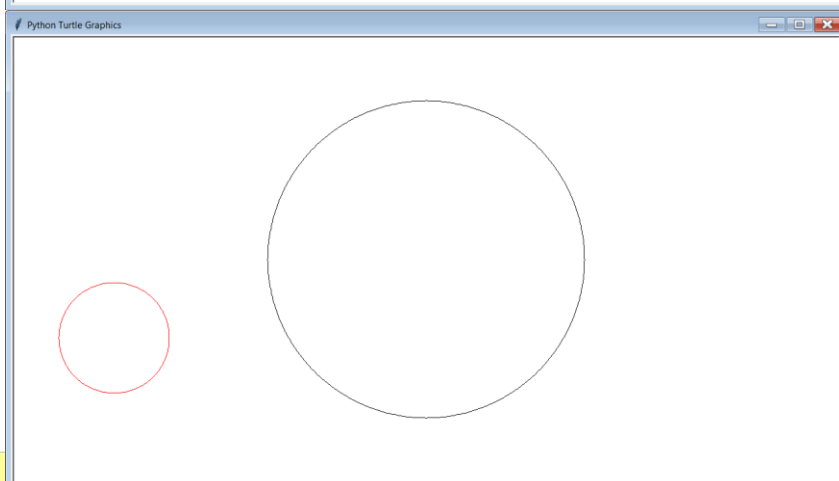
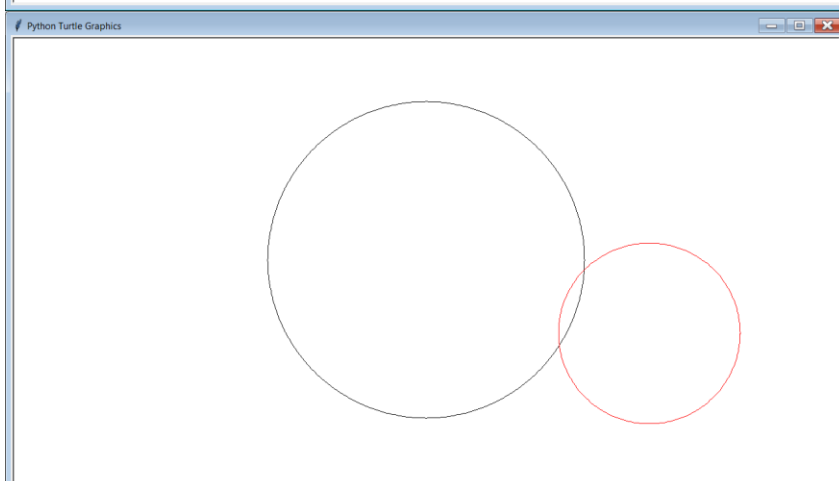
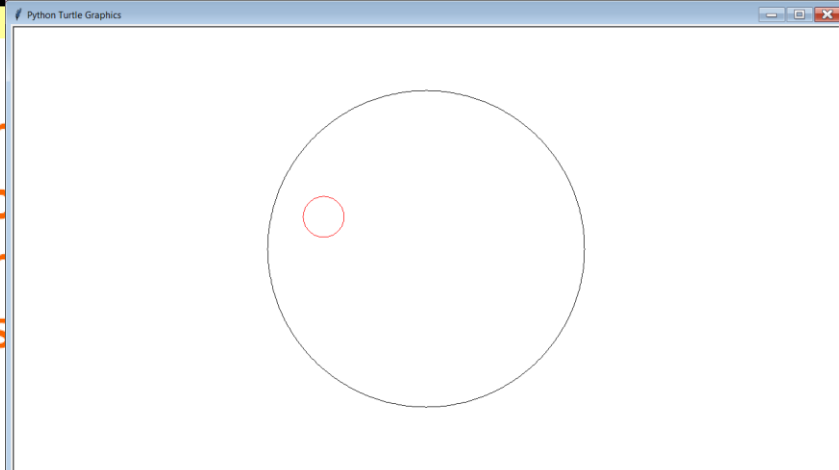
```
1 # RandomGraphics01.py
2 # This program first displays two black lines in a
3 # fixed location. This is followed by generating
4 # four random values, which are used to draw a third,
5 # red line in a random location.
6
7
8 from Graphics import *
9 from random import randint
10
11 beginGrafX(1300,700)
12
13 drawLine(650,0,650,700)
14 drawLine(0,350,1300,350)
15
16 setColor("red")
17 x1 = randint(0,1300)
18 y1 = randint(0,700)
19 x2 = randint(0,1300)
20 y2 = randint(0,700)
21 drawLine(x1,y1,x2,y2)
22
23 endGrafX()
```

```
1 # RandomGraphics01.py
2 # This program first displays
3 # fixed location. This is fol
4 # four random values, which ar
5 # red line in a random locatio
6
7
8 from Graphics import *
9 from random import randint
10
11 beginGrfx(1300,700)
12
13 drawLine(650,0,650,700)
14 drawLine(0,350,1300,350)
15
16 setColor("red")
17 x1 = randint(0,1300)
18 y1 = randint(0,700)
19 x2 = randint(0,1300)
20 y2 = randint(0,700)
21 drawLine(x1,y1,x2,y2)
22
23 endGrfx()
```



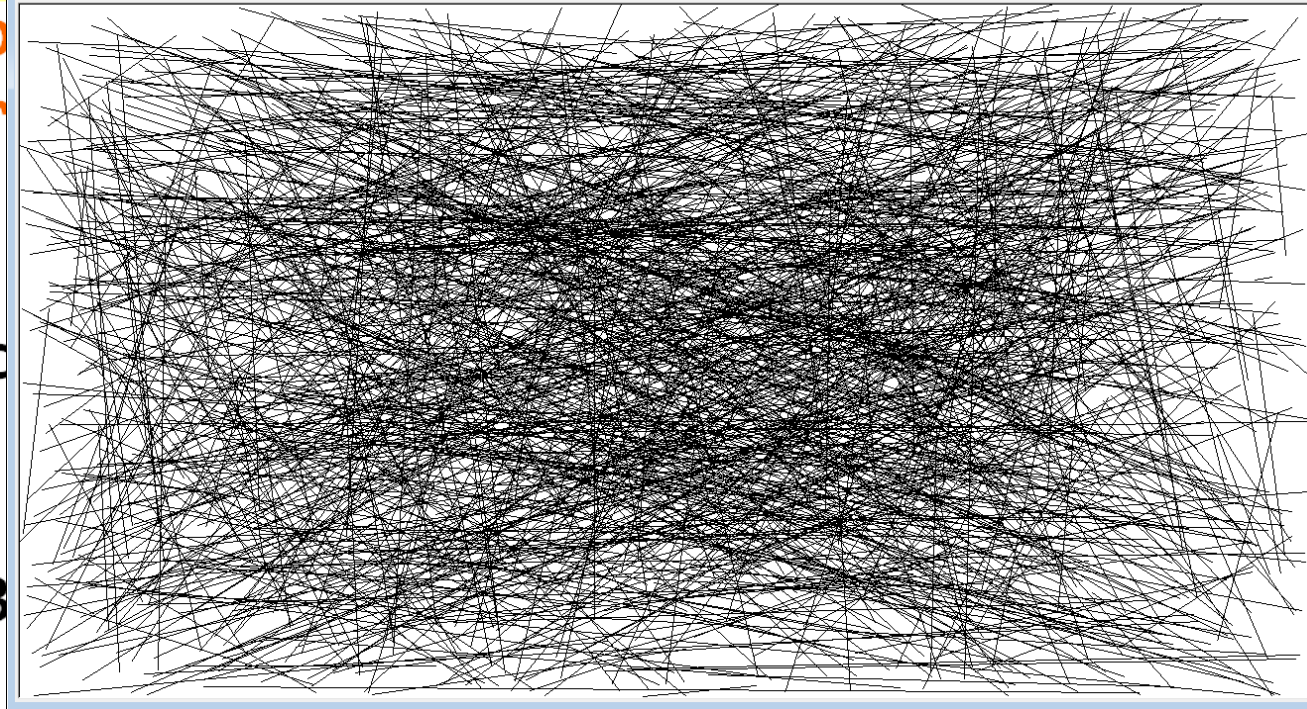
```
1 # RandomGraphics02.py
2 # This program displays a circle in a fixed location
3 # with a fixed size. 3 random values are generated.
4 # Two are used for the center location of the circle
5 # and a third is used for its radius.
6
7
8 from Graphics import *
9 from random import randint
10
11 beginGrafX(1300,700)
12
13 drawCircle(650,350,250)
14
15 setColor("red")
16 x = randint(150,1150)
17 y = randint(150,550)
18 r = randint(10,150)
19 drawCircle(x,y,r)
20
21 endGrafX()
```

```
1 # RandomGraphics02.py
2 # This program displays a circle
3 # with a fixed size. 3 random
4 # Two are used for the center
5 # and a third is used for its
6
7
8 from Graphics import *
9 from random import randint
10
11 beginGrfx(1300,700)
12
13 drawCircle(650,350,250)
14
15 setColor("red")
16 x = randint(150,1150)
17 y = randint(150,550)
18 r = randint(10,150)
19 drawCircle(x,y,r)
20
21 endGrfx()
```



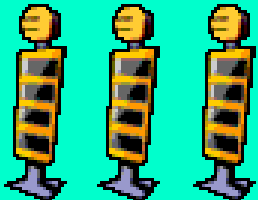
```
1 # RandomGraphics03.py
2 # This program displays 1000 random lines.
3
4
5 from Graphics import *
6 from random import randint
7
8 beginGrfx(1300,700)
9
10 for k in range(1000):
11     x1 = randint(0,1300)
12     y1 = randint(0,700)
13     x2 = randint(0,1300)
14     y2 = randint(0,700)
15     drawLine(x1,y1,x2,y2)
16
17 endGrfx()
```

```
1 # RandomGrap
2 # This progr
3
4
5 from Graphic
6 from random
7
8 beginGrfx(13
9
10 for k in range(1000):
11     x1 = randint(0,1300)
12     y1 = randint(0,700)
13     x2 = randint(0,1300)
14     y2 = randint(0,700)
15     drawLine(x1,y1,x2,y2)
16
17 endGrfx()
```



Changing Random Number Ranges to Affect the Graphics Program Output

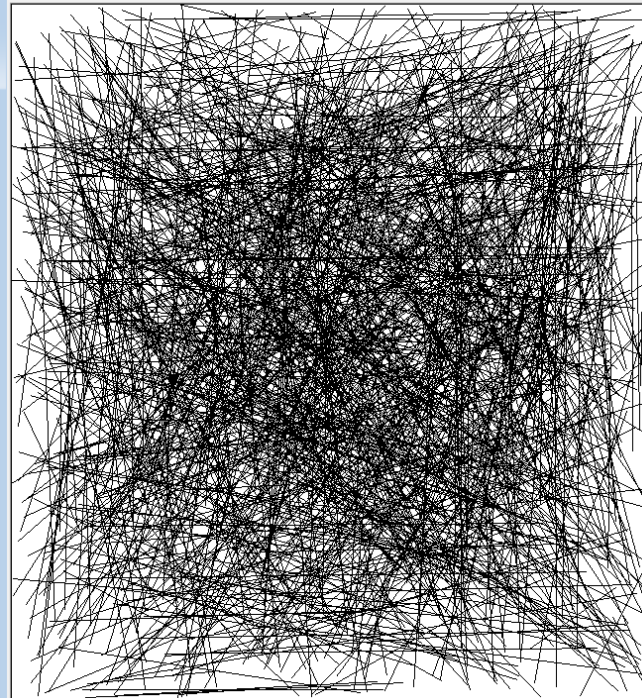
The next several slides will still show program **RandomGraphics03.py**, but the output of the program is different. You need to figure out how to change the each program to make it produce the output shown.



NOTE:
This skill is essential
in doing **Lab 8C!**

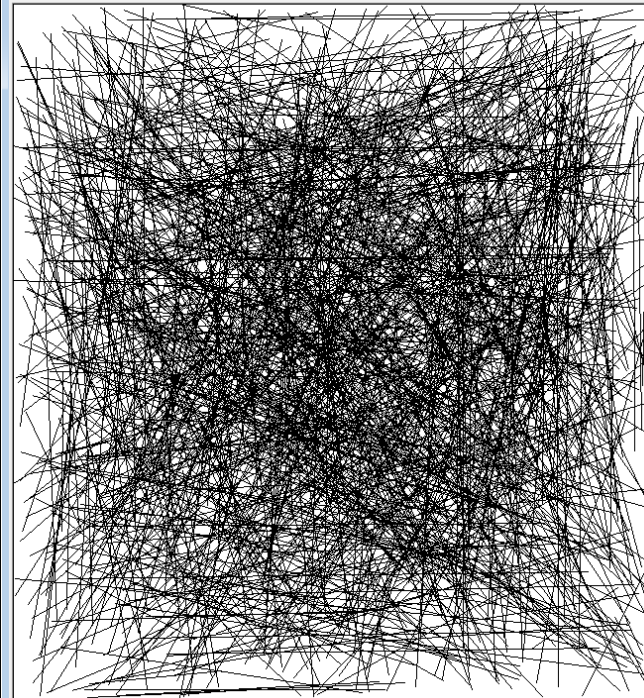


```
1 # RandomGrap
2 # This progr
3
4
5 from Graphic
6 from random
7
8 beginGrfx(13
9
10 for k in range(1000):
11     x1 = randint(0,1300)
12     y1 = randint(0,700)
13     x2 = randint(0,1300)
14     y2 = randint(0,700)
15     drawLine(x1,y1,x2,y2)
16
17 endGrfx()
```



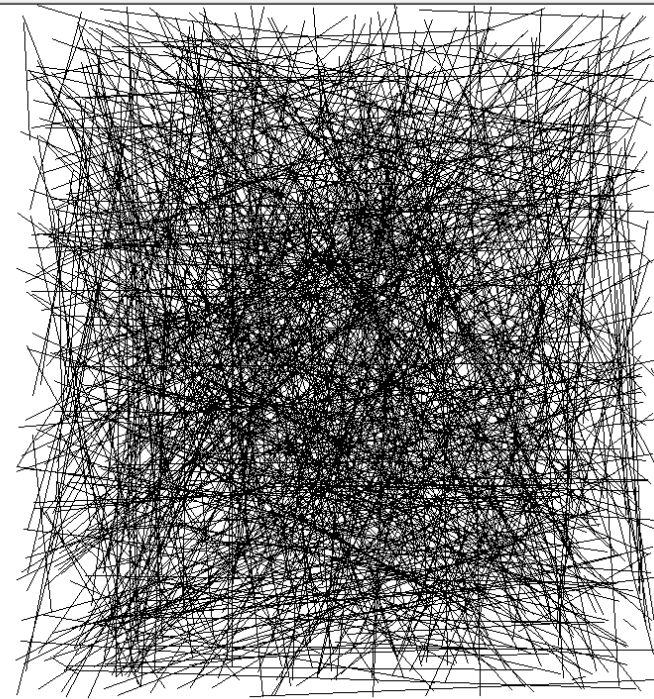
**Make the random lines
only show up on the
left $\frac{1}{2}$ of the screen.**

```
1 # RandomGrap
2 # This progr
3
4
5 from Graphic
6 from random
7
8 beginGrfx(13
9
10 for k in range(1000):
11     x1 = randint(0,650)
12     y1 = randint(0,700)
13     x2 = randint(0,650)
14     y2 = randint(0,700)
15     drawLine(x1,y1,x2,y2)
16
17 endGrfx()
```



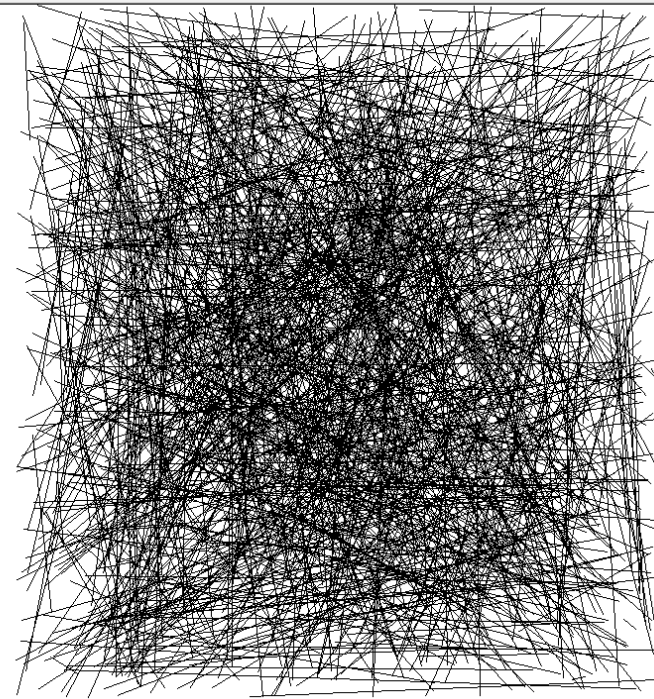
Solution

```
1 # RandomGrap
2 # This progr
3
4
5 from Graphic
6 from random
7
8 beginGrfx(13
9
10 for k in range(1000):
11     x1 = randint(0,1300)
12     y1 = randint(0,700)
13     x2 = randint(0,1300)
14     y2 = randint(0,700)
15     drawLine(x1,y1,x2,y2)
16
17 endGrfx()
```



**Make the random lines
only show up on the
right ½ of the screen.**


```
1 # RandomGrap
2 # This progr
3
4
5 from Graphic
6 from random
7
8 beginGrfx(13
9
10 for k in range(1000):
11     x1 = randint(650,1300)
12     y1 = randint(0,700)
13     x2 = randint(650,1300)
14     y2 = randint(0,700)
15     drawLine(x1,y1,x2,y2)
16
17 endGrfx()
```



Solution

```
1 # RandomGrap
2 # This progr
3
4
5 from Graphic
6 from random
7
8 beginGrfx(13
9
10 for k in range(1000):
11     x1 = randint(0,1300)
12     y1 = randint(0,700)
13     x2 = randint(0,1300)
14     y2 = randint(0,700)
15     drawLine(x1,y1,x2,y2)
16
17 endGrfx()
```



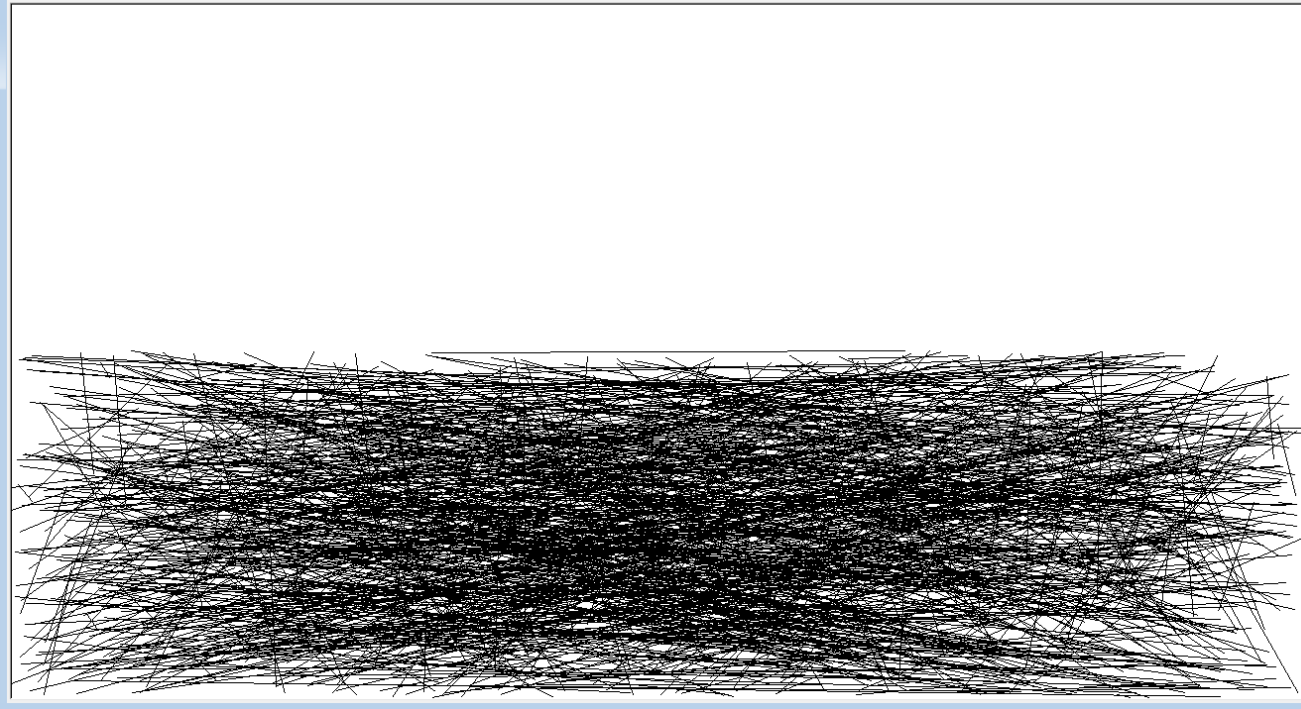
**Make the random lines
only show up on the
top ½ of the screen.**

```
1 # RandomGrap
2 # This progr
3
4
5 from Graphic
6 from random
7
8 beginGrfx(13
9
10 for k in range(1000):
11     x1 = randint(0,1300)
12     y1 = randint(0,350)
13     x2 = randint(0,1300)
14     y2 = randint(0,350)
15     drawLine(x1,y1,x2,y2)
16
17 endGrfx()
```



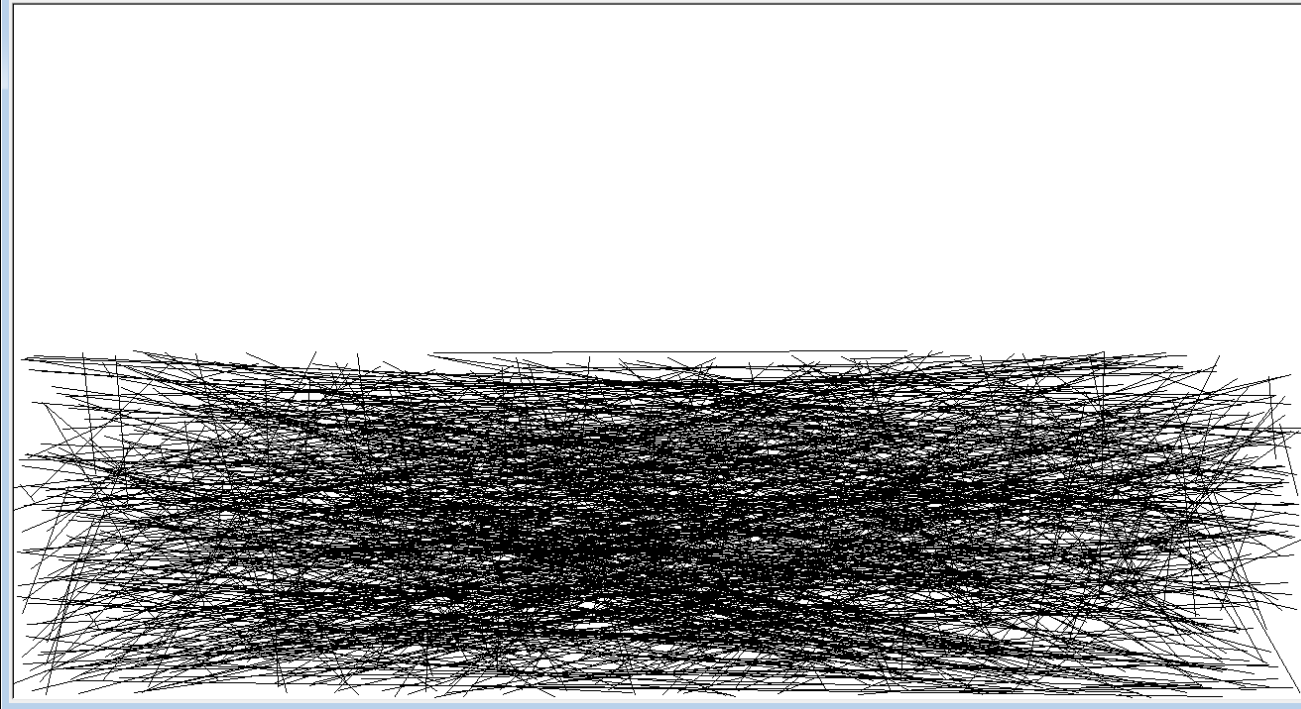
Solution


```
1 # RandomGrap
2 # This progr
3
4
5 from Graphic
6 from random
7
8 beginGrfx(1300
9
10 for k in range(1000):
11     x1 = randint(0,1300)
12     y1 = randint(0,700)
13     x2 = randint(0,1300)
14     y2 = randint(0,700)
15     drawLine(x1,y1,x2,y2)
16
17 endGrfx()
```



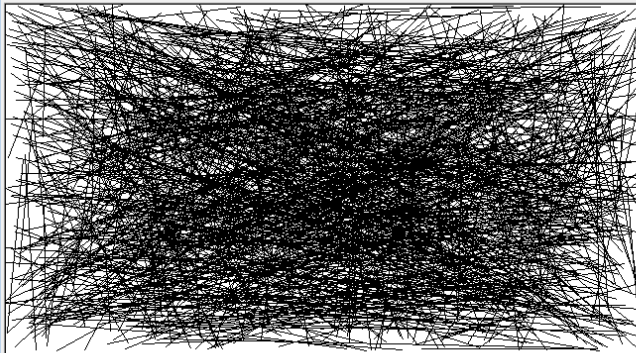
**Make the random lines
only show up on the
bottom 1/2 of the screen.**

```
1 # RandomGrap
2 # This progr
3
4
5 from Graphic
6 from random
7
8 beginGrfx(1300
9
10 for k in range(1000):
11     x1 = randint(0,1300)
12     y1 = randint(350,700)
13     x2 = randint(0,1300)
14     y2 = randint(350,700)
15     drawLine(x1,y1,x2,y2)
16
17 endGrfx()
```



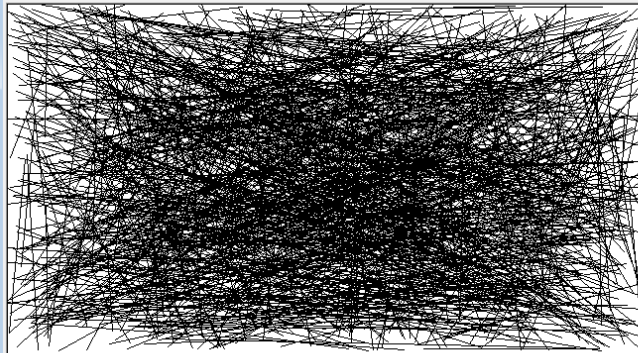
Solution

```
1 # RandomGrap
2 # This progr
3
4
5 from Graphic
6 from random
7
8 beginGrfx(13
9
10 for k in range(1000):
11     x1 = randint(0,1300)
12     y1 = randint(0,700)
13     x2 = randint(0,1300)
14     y2 = randint(0,700)
15     drawLine(x1,y1,x2,y2)
16
17 endGrfx()
```



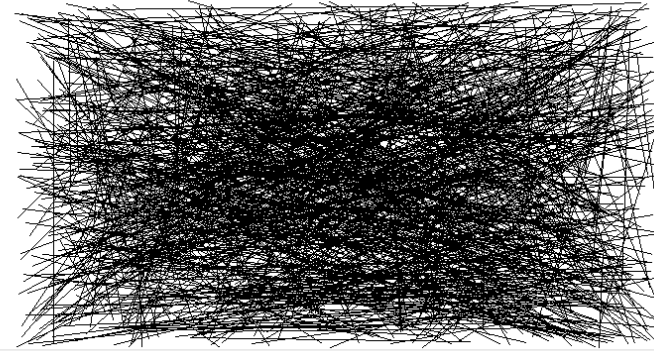
**Make the random lines
only show up in the
top-left 1/4 of the screen.**

```
1 # RandomGrap
2 # This progr
3
4
5 from Graphic
6 from random
7
8 beginGrfx(13
9
10 for k in range(1000):
11     x1 = randint(0,650)
12     y1 = randint(0,350)
13     x2 = randint(0,650)
14     y2 = randint(0,350)
15     drawLine(x1,y1,x2,y2)
16
17 endGrfx()
```



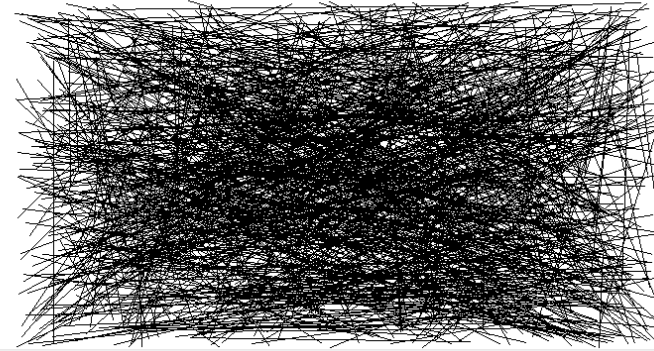
Solution

```
1 # RandomGrap
2 # This progr
3
4
5 from Graphic
6 from random
7
8 beginGrfx(13
9
10 for k in range(1000):
11     x1 = randint(0,1300)
12     y1 = randint(0,700)
13     x2 = randint(0,1300)
14     y2 = randint(0,700)
15     drawLine(x1,y1,x2,y2)
16
17 endGrfx()
```



**Make the random lines
only show up in the bottom-
right 1/4 of the screen.**


```
1 # RandomGrap
2 # This progr
3
4
5 from Graphic
6 from random
7
8 beginGrfx(13
9
10 for k in range(1000):
11     x1 = randint(650,1300)
12     y1 = randint(350,700)
13     x2 = randint(650,1300)
14     y2 = randint(350,700)
15     drawLine(x1,y1,x2,y2)
16
17 endGrfx()
```

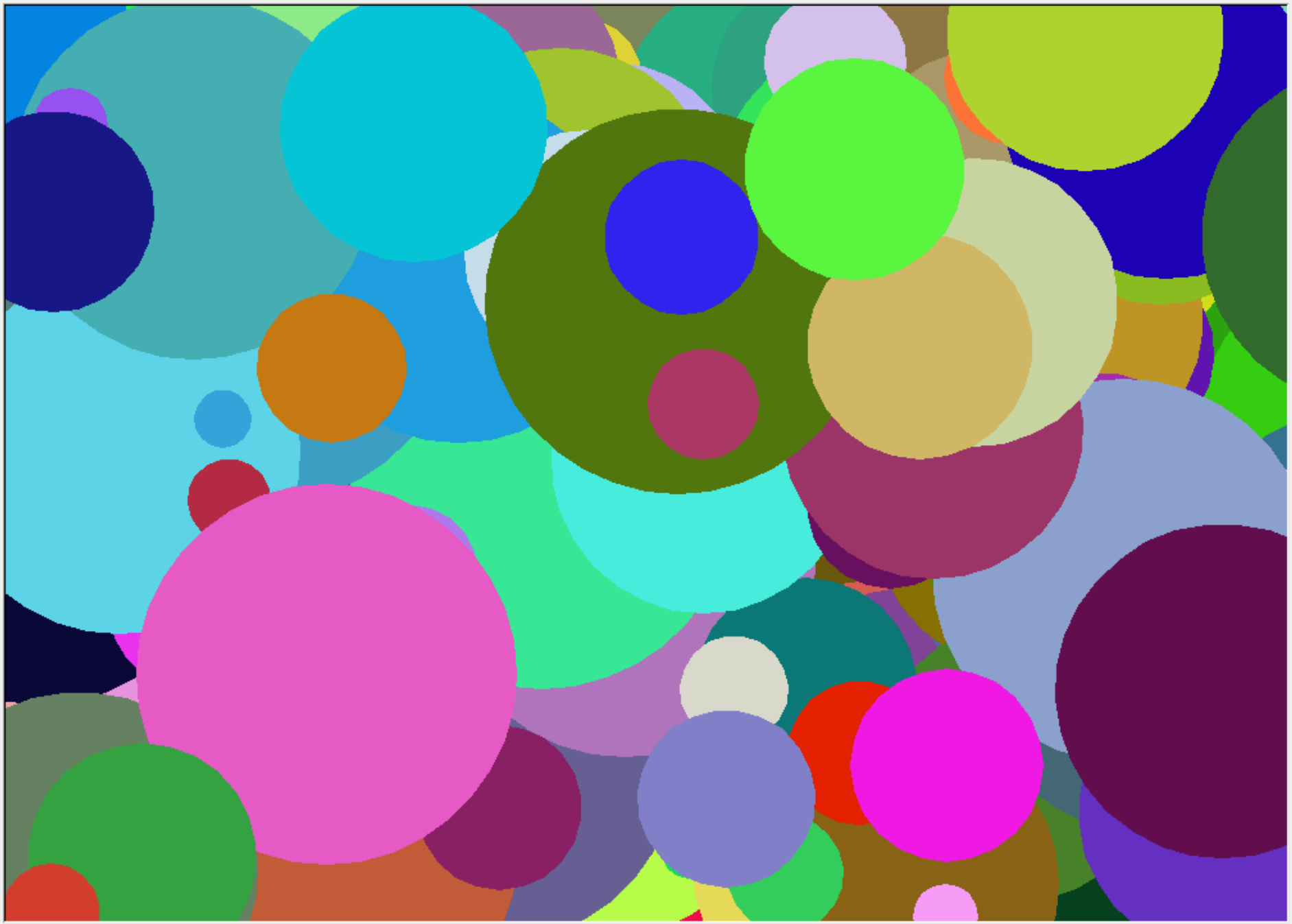


Solution

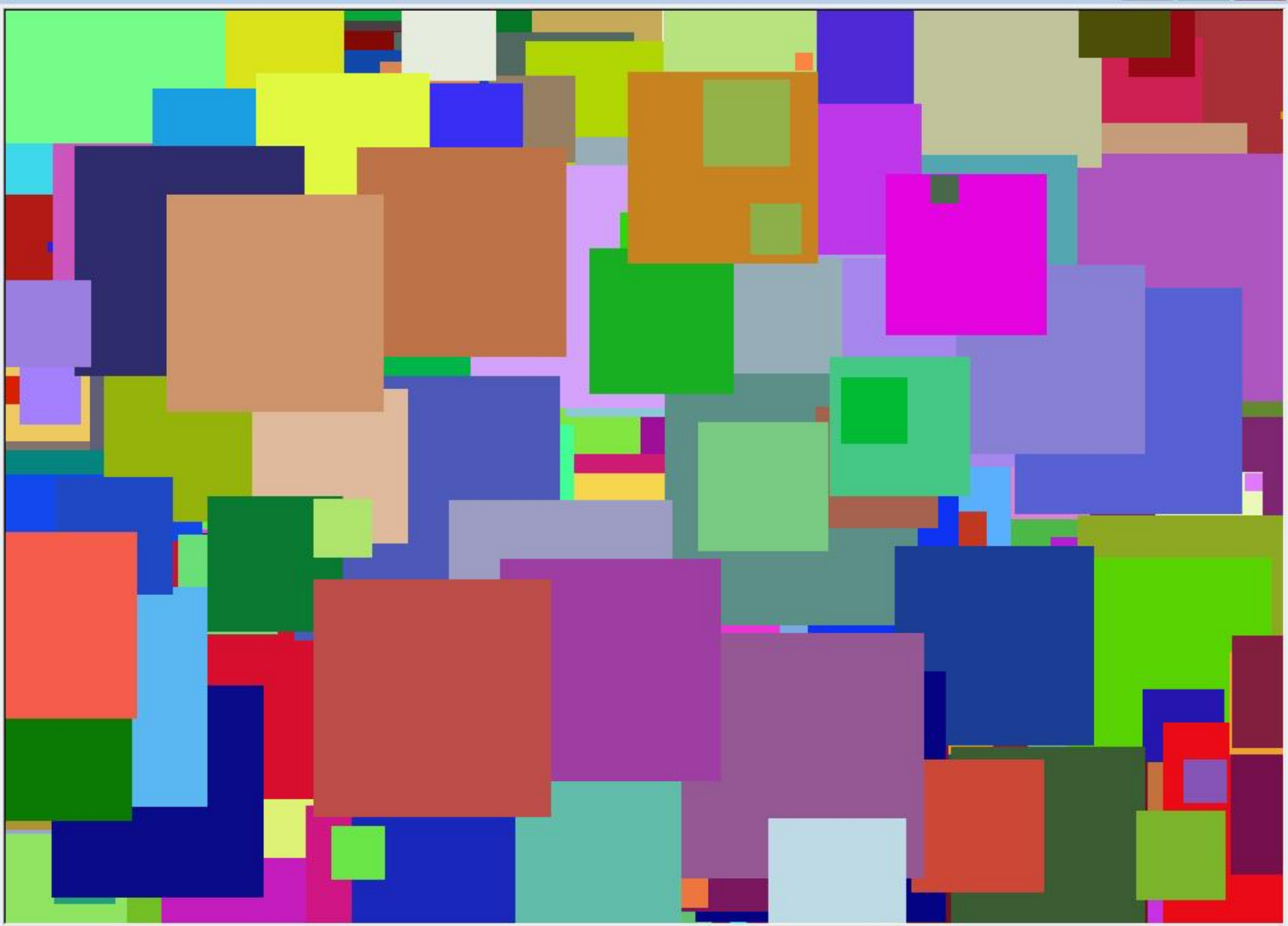
```
1 # RandomGraphics04.py
2 # This program displays 500 randomly colored
3 # solid circles with a fixed radius of 75.
4
5
6 from Graphics import *
7 from random import randint
8
9 beginGrfx(1300,700)
10
11 for k in range(500):
12     x = randint(0,1300)
13     y = randint(0,700)
14     red = randint(0,255)
15     green = randint(0,255)
16     blue = randint(0,255)
17     setColor(red,green,blue)
18     fillCircle(x,y,75)
19
20 endGrfx()
```




```
1 # RandomGraphics05.py
2 # This program displays 500 randomly colored
3 # solid circles with random radii.
4
5
6 from Graphics import *
7 from random import randint
8
9 beginGrfx(1300,700)
10
11 for k in range(500):
12     x = randint(0,1300)
13     y = randint(0,700)
14     red = randint(0,255)
15     green = randint(0,255)
16     blue = randint(0,255)
17     radius = randint(1,150)
18     setColor(red,green,blue)
19     fillCircle(x,y,radius)
20
21 endGrfx()
```



```
1 # RandomGraphics06.py
2 # This program displays 500 randomly colored squares.
3 # Creating the random colors is also simplified
4 # with the <setRandomColor> procedure.
5
6
7 from Graphics import *
8 from random import randint
9
10 beginGrfx(1300,700)
11
12 for k in range(500):
13     x = randint(0,1300)
14     y = randint(0,700)
15     radius = randint(1,150)
16     sides = 4
17     setRandomColor()
18     fillRegularPolygon(x,y,radius,sides)
19
20 endGrfx()
```

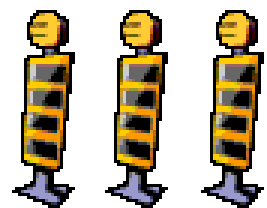


```
1 # RandomGraphics07.py
2 # This program displays 500 randomly colored
3 # polygons with a random number of sides.
4
5
6 from Graphics import *
7 from random import randint
8
9 beginGrfx(1300,700)
10
11 for k in range(500):
12     x = randint(0,1300)
13     y = randint(0,700)
14     radius = randint(1,150)
15     sides = randint(3,10)
16     setRandomColor()
17     fillRegularPolygon(x,y,radius,sides)
18
19 endGrfx()
```

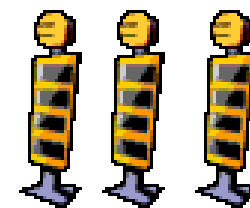


```
1 # RandomGraphics08.py
2 # This program demonstrates that even the
3 # width of the lines can be random.
4
5
6 from Graphics import *
7 from random import randint
8
9 beginGrfx(1300,700)
10
11 for k in range(500):
12     x = randint(0,1300)
13     y = randint(0,700)
14     radius = randint(1,150)
15     numLines = randint(3,10)
16     setRandomColor()
17     w = randint(1,30)
18     width(w)
19     drawBurst(x,y,radius,numLines)
20
21 endGrfx()
```



Lab 8C



What you saw in the past 6 program examples relates directly to what you will be doing in Lab 8C.

