Exposure CS 2022 for CS1

Chapter 8 Section 7-9 Slides

Creating Custom Colors and Repetition with Random Numbers and Graphics

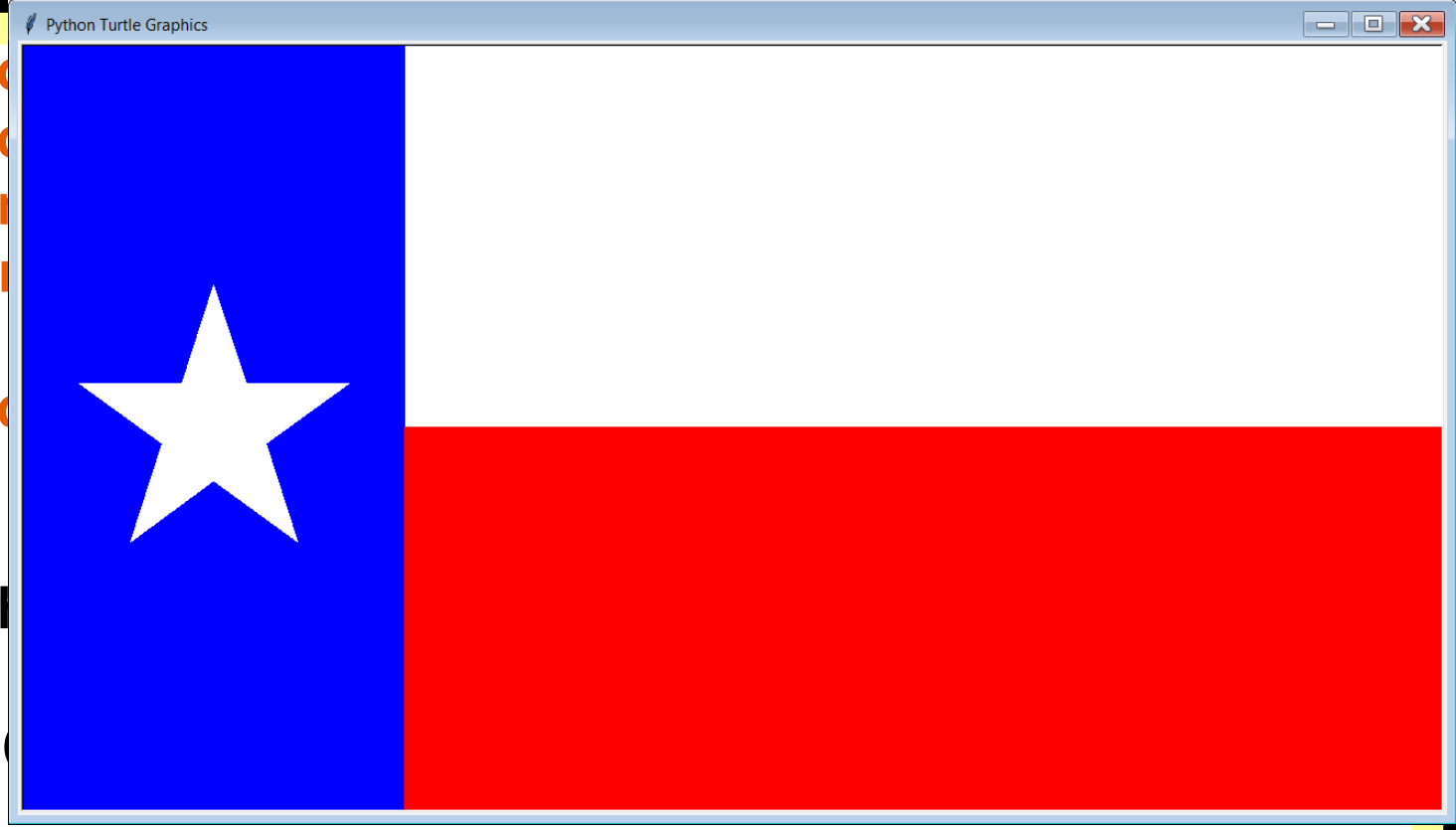
PowerPoint Presentation
created by:
Mr. John L. M. Schram
and Mr. Leon Schram
Authors of Exposure
Computer Science

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 beginGrfx(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 endGrfx()
```

```
 1 # CustomC
 2 # This pr
 3 # built-i
 4 # this ce
 5 # not use
 6 # are use
 7
 8
 9 from Grap
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()
```

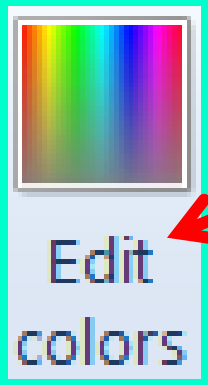

```
1  # CustomColo
2  # This progr
3  # being usec
4  # will draw
5  # official s
6
7
8  from Graphic
9
10 beginGrfx(13
11
12 setColor(0,39,104)        # Texas Flag blue
13 fillRectangle(0,0,350,700)
14 setColor(190,10,47)       # Texas Flag red
15 fillRectangle(350,350,1300,700)
16 setColor(255,255,255)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

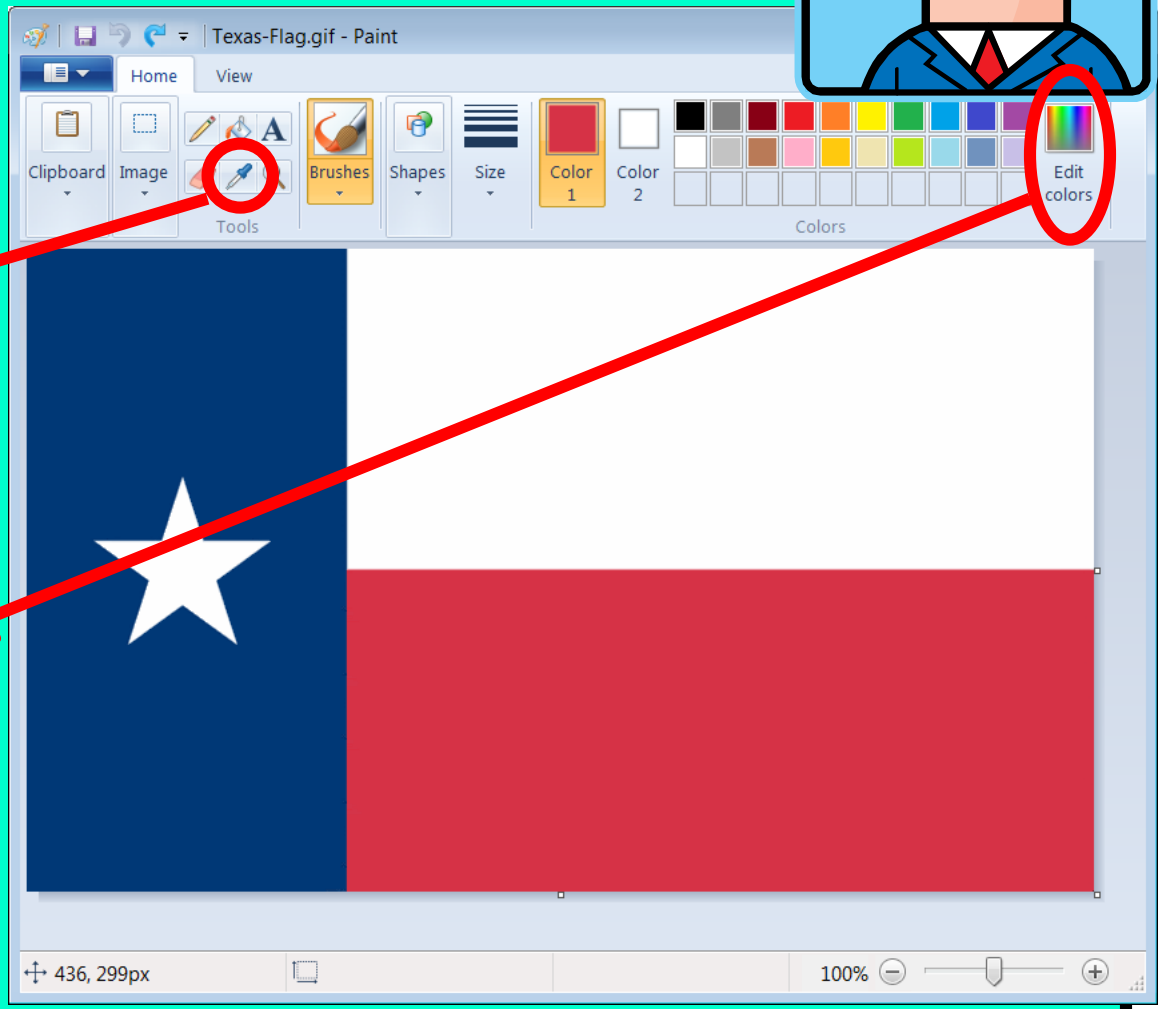
Where do you get the 3 color #s

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.





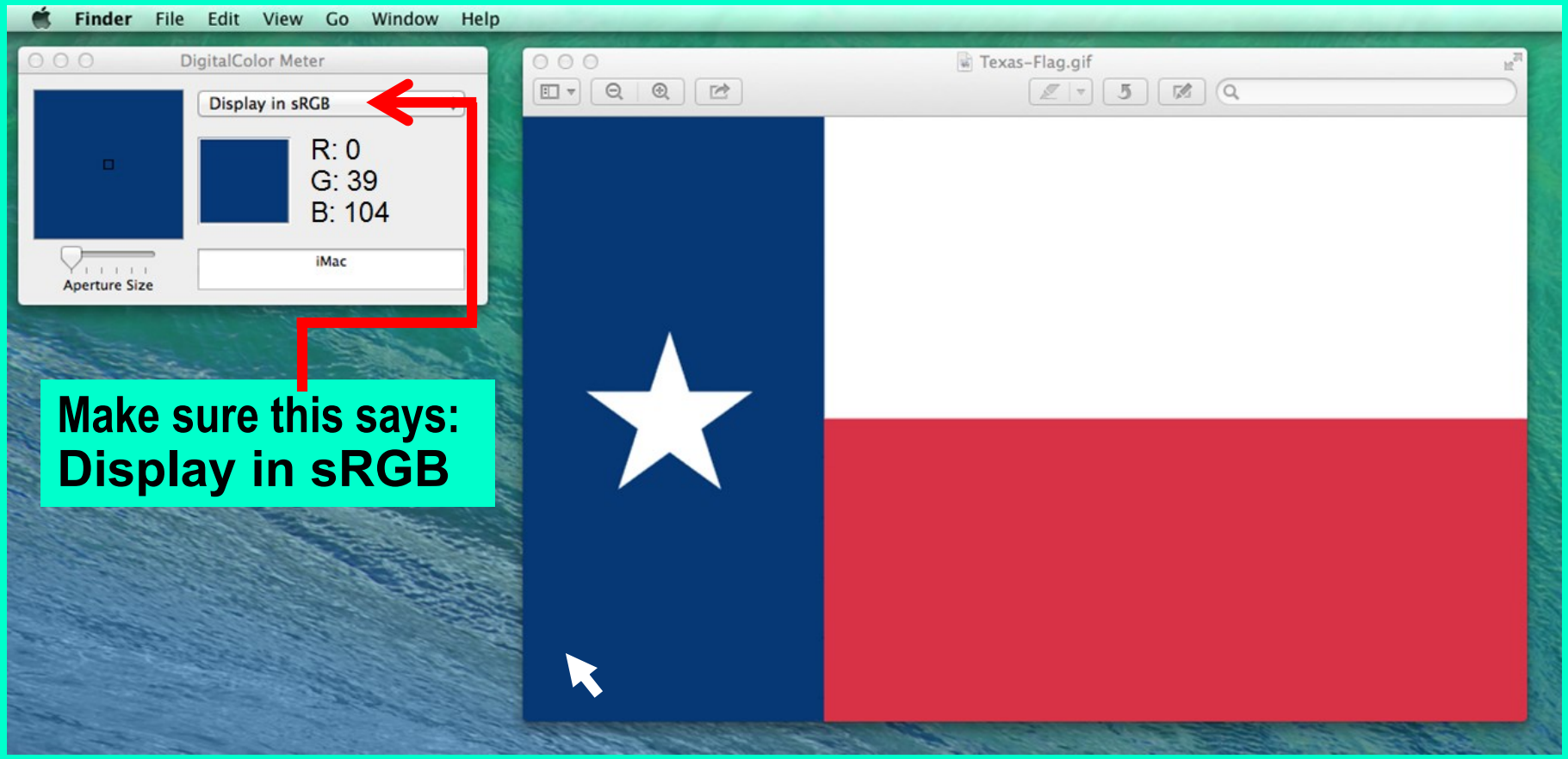
By using the same 3 numbers in my program, I get the exact same color in the output of my Python program:

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

You can use DigitalColor Meter on a Mac

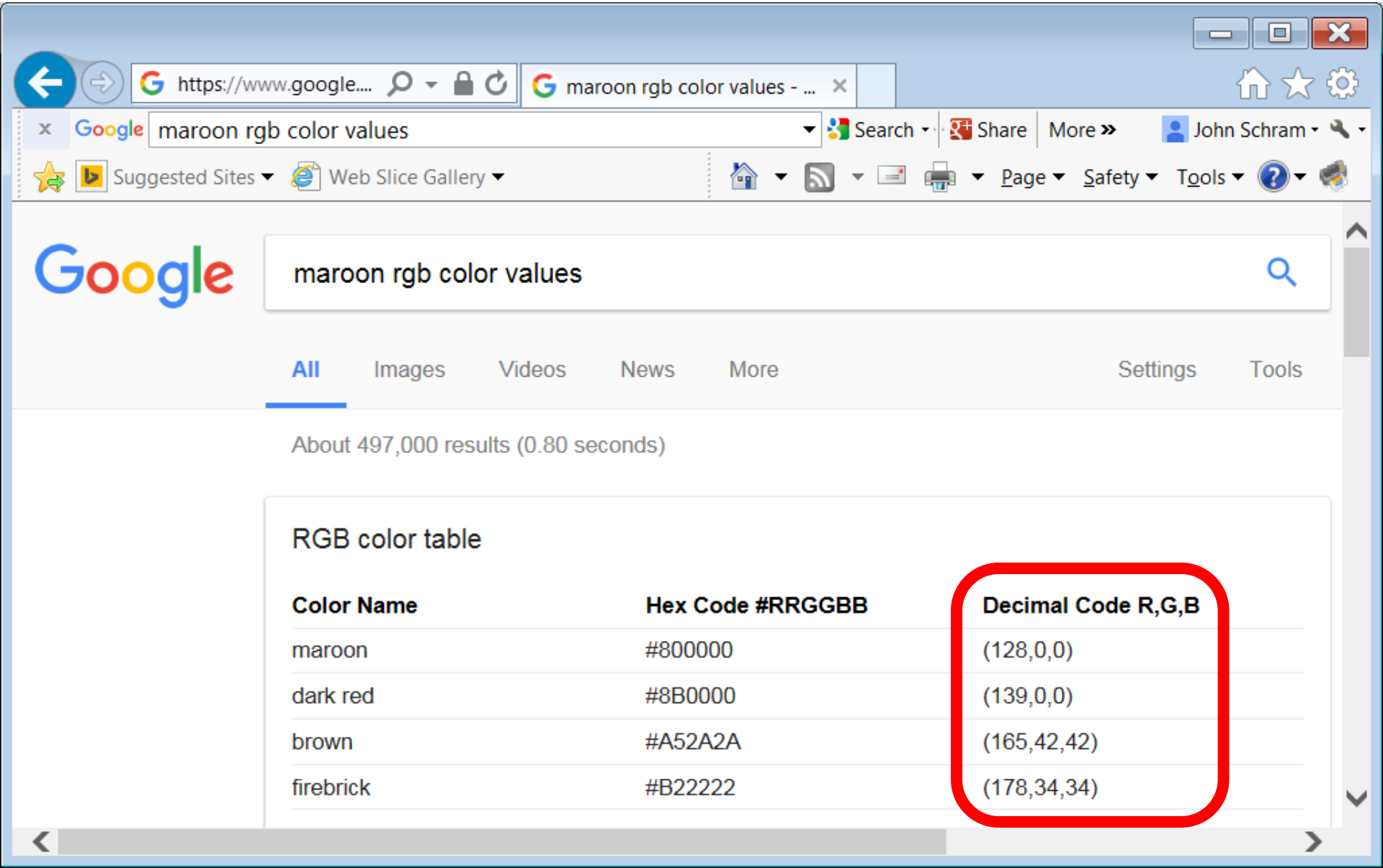
Double-click the image; which will open it up in a Preview window.

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

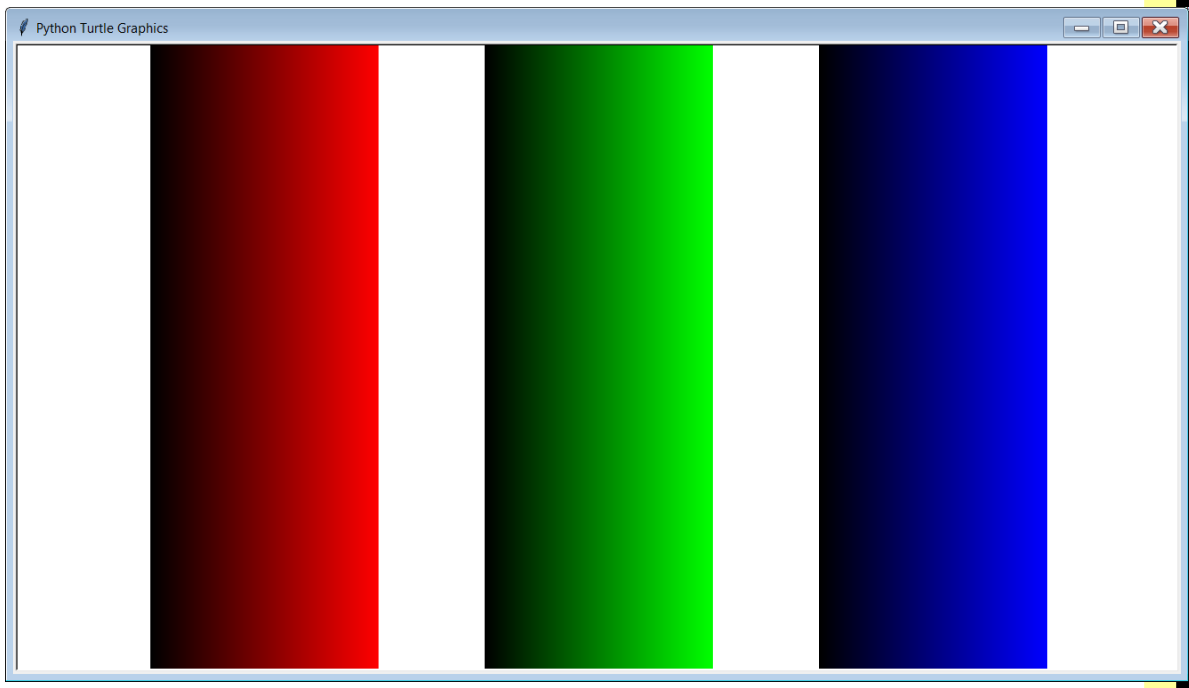


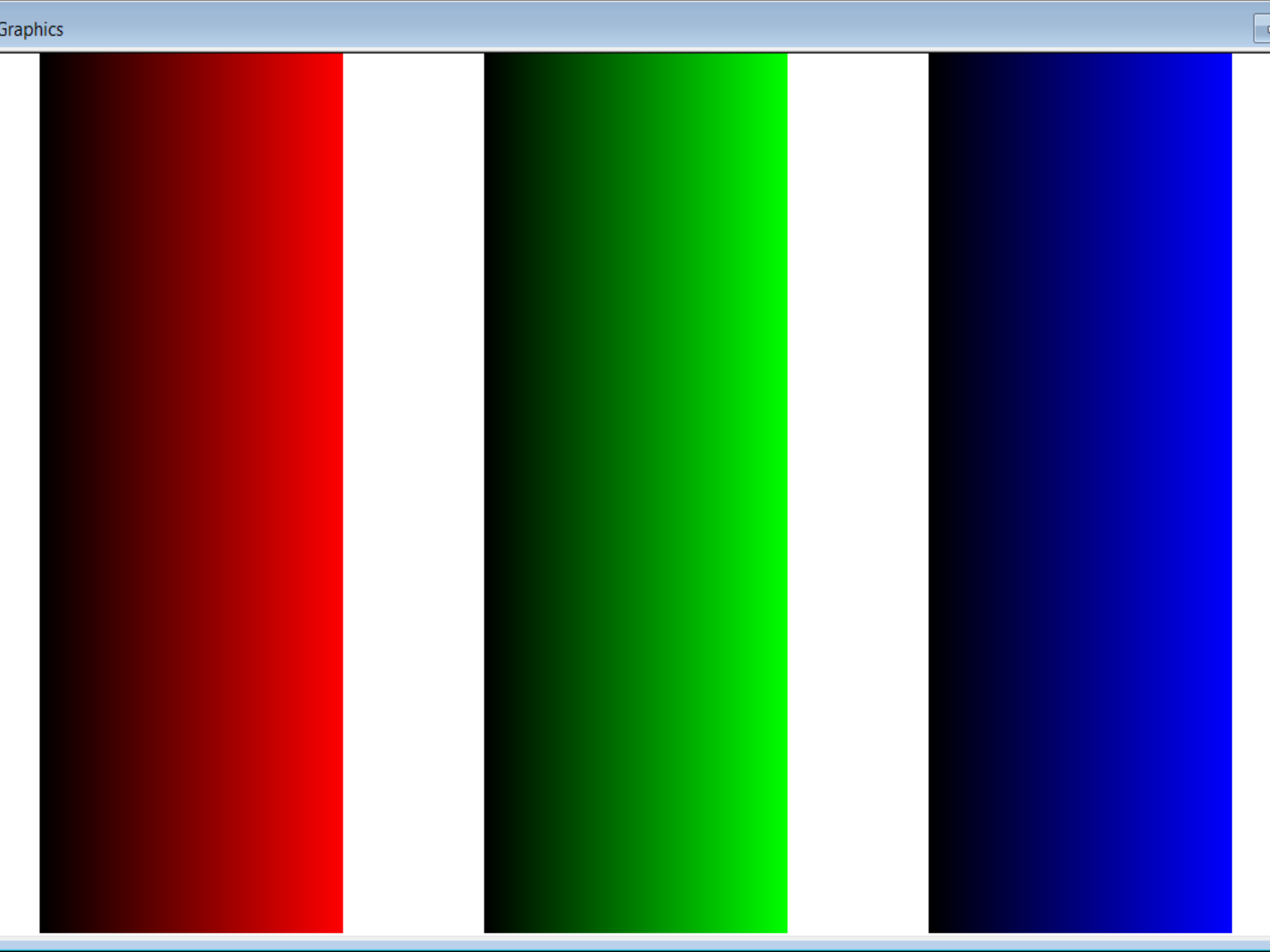
Move your mouse pointer inside the preview window over the desired color.
The DigitalColor Meter will show you the red, green and blue values.

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



```
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  from Graphics import *
8
9  beginGrfx(1300,700)
10
11 x = 150;
12 for red in range(256):
13     setColor(red,0,0)
14     drawLine(x,0,x,700)
15     x += 1
16
17 x = 525;
18 for green in range(256)
19     setColor(0,green,0)
20     drawLine(x,0,x,700)
21     x += 1
22
23 x = 900;
24 for blue in range(256):
25     setColor(0,0,blue)
26     drawLine(x,0,x,700)
27     x += 1
28
29 endGrfx()
```




Graphics

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 to 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 to the <ra
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

22
53
46
99
3

----jGRASP:
```

```
----jGRASP

58
42
30
28
42

----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

89
45
34
40
10

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

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

Enter the smallest number.  -->  1000
Enter the largest number.   -->  9999

6388
1993
8690
3726
2333

----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 cryptology.

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

from random import randint

sevens = 0
elevens = 0
snakeEyes = 0
doubles = 0

for roll in range(1000000):
    dice = randint(2,12)
    if dice == 2:
        snakeEyes += 1
    if dice == 7:
        sevens += 1
    if dice == 11:
        elevens += 1

print()
print("# of Sevens:     ",sevens)
print("# of Elevens:    ",elevens)
print("# of Snake Eyes:",snakeEyes)
print("# of Doubles:    ",doubles)
```

```
----jGRASP exec: python

# of Sevens:      91467
# of Elevens:     90853
# of Snake Eyes: 90756
# 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

```python
# RandomNumbers04.py
# This program PROPERLY simulates rolling dice 1,000,000 times.

from random import randint

sevens = 0
elevens = 0
snakeEyes = 0
doubles = 0

for roll in range(1000000):
    die1 = randint(1,6)
    die2 = randint(1,6)
    diceTotal = die1 + die2
    #
    if diceTotal == 2:
        snakeEyes += 1
    if diceTotal == 7:
        sevens += 1
    if diceTotal == 11:
        elevens += 1
    if die1 == die2:
        doubles += 1

print()
print("# of Sevens:    ",sevens)
print("# of Elevens:   ",elevens)
print("# of Snake Eyes:",snakeEyes)
print("# of Doubles:   ",doubles)
```

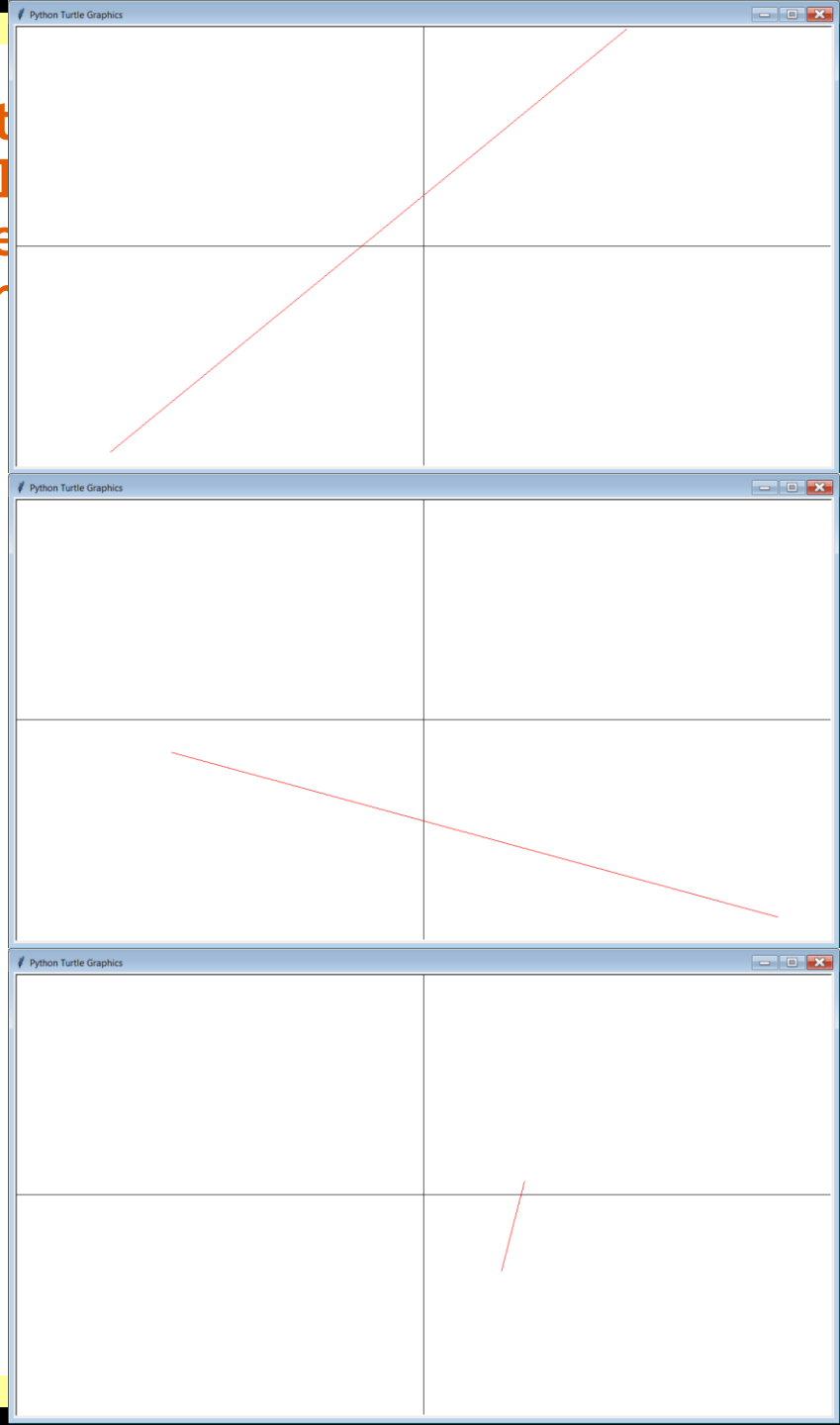
```
----jGRASP exec: python

# of Sevens:      166191
# of Elevens:     55302
# of Snake Eyes:  27693
# of Doubles:     166532

----jGRASP: operation c
```

Section 8.9

Using Random #s with Graphics

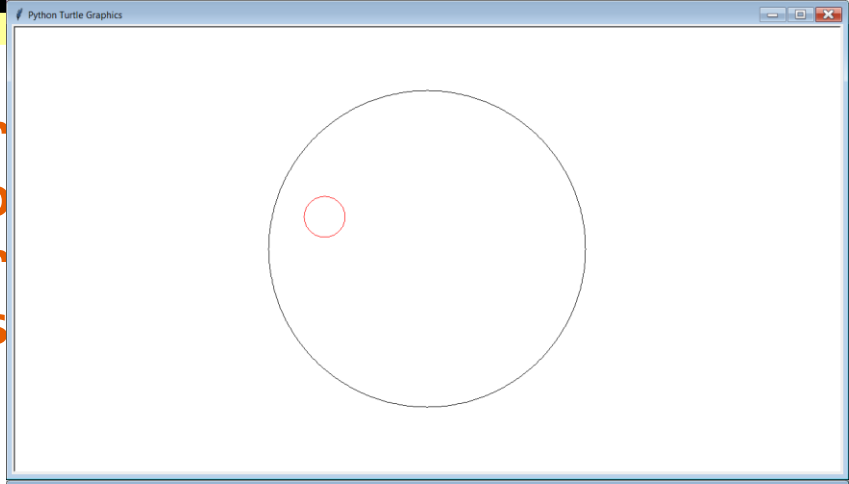
```
 1 # RandomGraphics01.py
 2 # This program first displays two black lines in a
 3 # fixed location.  This is followed by generating
 4 # four random values, which are used to draw a third,
 5 # red line in a random location.
 6
 7
 8 from Graphics import *
 9 from random import randint
10
11 beginGrfx(1300,700)
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()
```

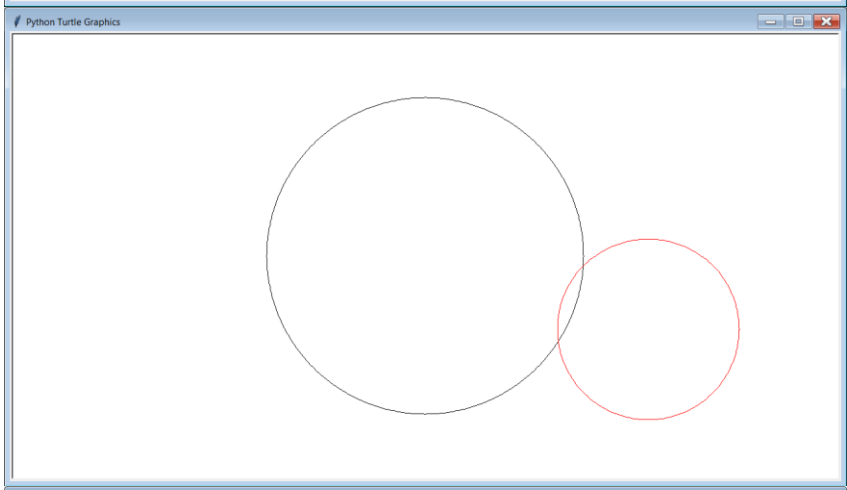
```
 1 # RandomGraphics01.py
 2 # This program first displays t
 3 # fixed location.  This is foll
 4 # four random values, which are
 5 # red line in a random locatior
 6
 7
 8 from Graphics import *
 9 from random import randint
10
11 beginGrfx(1300,700)
12
13 drawLine(650,0,650,700)
14 drawLine(0,350,1300,350)
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()
```

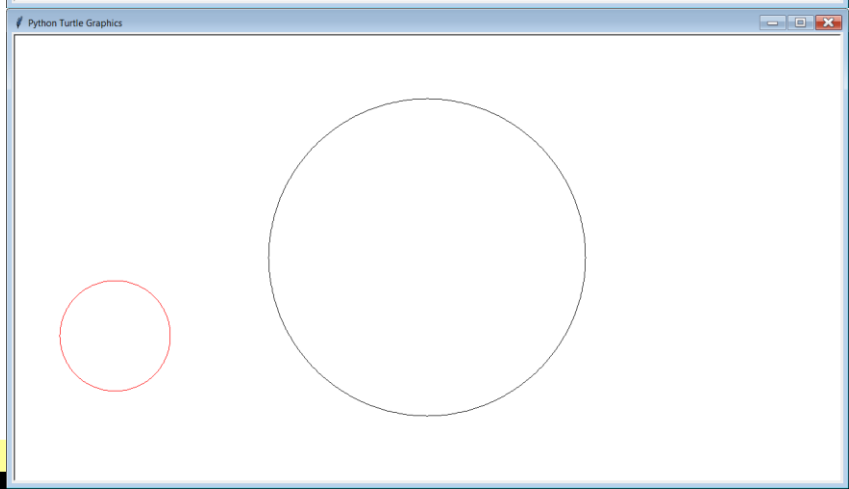


```
 1 # RandomGraphics02.py
 2 # This program displays a circle in a fixed location
 3 # with a fixed size.  3 random values are generated.
 4 # Two are used for the center location of the circle
 5 # and a third is used for its radius.
 6
 7
 8 from Graphics import *
 9 from random import randint
10
11 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 # RandomGraphics02.py
 2 # This program displays a cir
 3 # with a fixed size.  3 rando
 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()
```

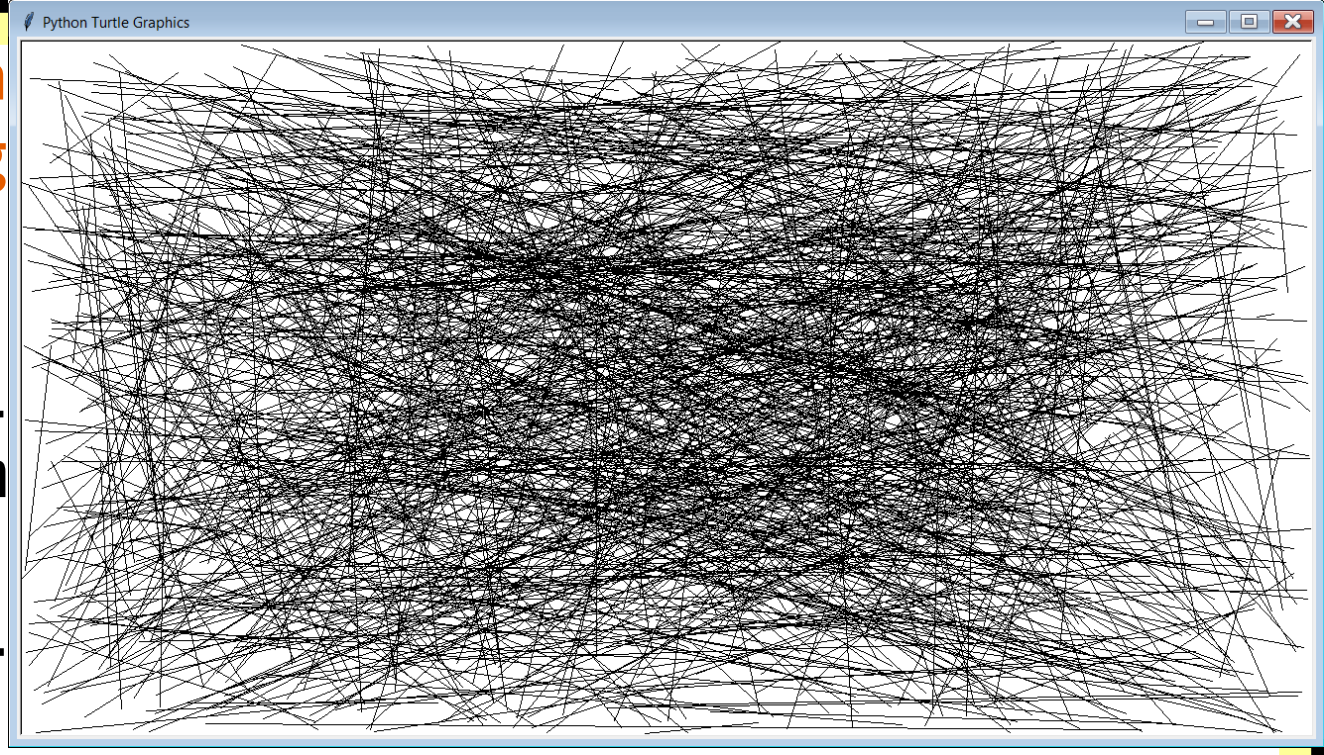






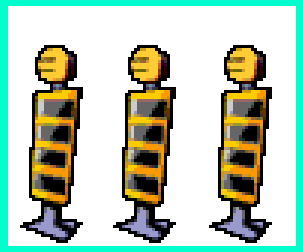
```
 1 # RandomGraphics03.py
 2 # This program displays 1000 random lines.
 3
 4
 5 from Graphics import *
 6 from random import randint
 7
 8 beginGrfx(1300,700)
 9
10 for k in range(1000):
11     x1 = randint(0,1300)
12     y1 = randint(0,700)
13     x2 = randint(0,1300)
14     y2 = randint(0,700)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

```
1  # RandomGra
2  # This prog
3
4
5  from Graphi
6  from random
7
8  beginGrfx(1
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()
```



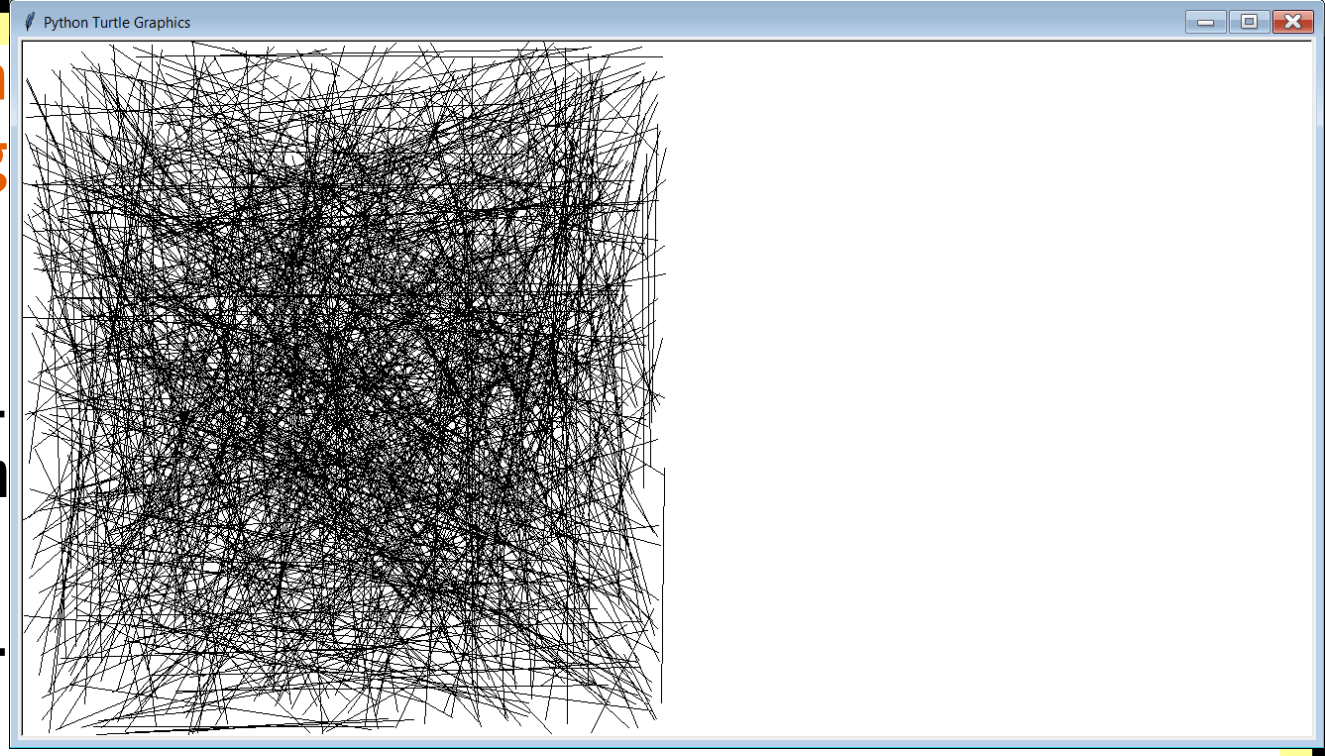
Changing Random Number Ranges to Affect the Graphics Program Output

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.



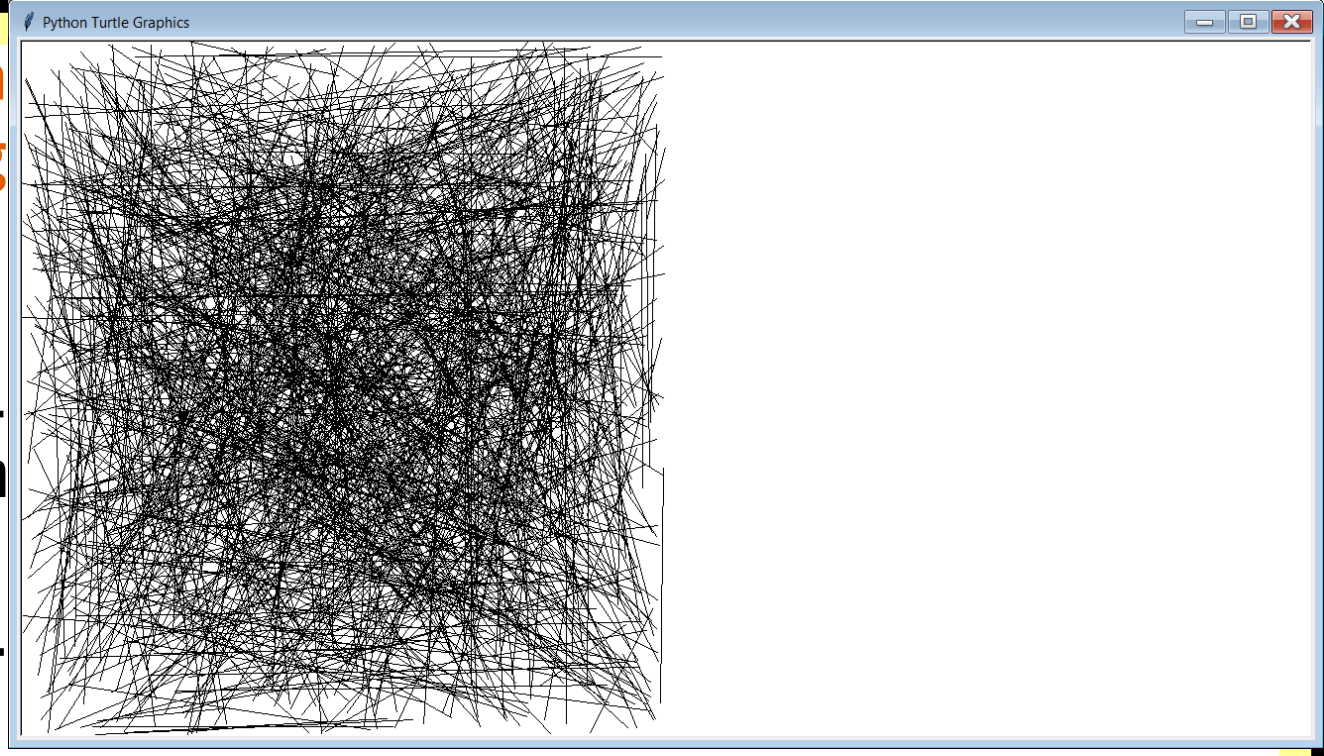
NOTE:
This skill is essential in doing Lab 8C!

```
1  # RandomGra
2  # This prog
3
4
5  from Graphi
6  from random
7
8  beginGrfx(1
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()
```



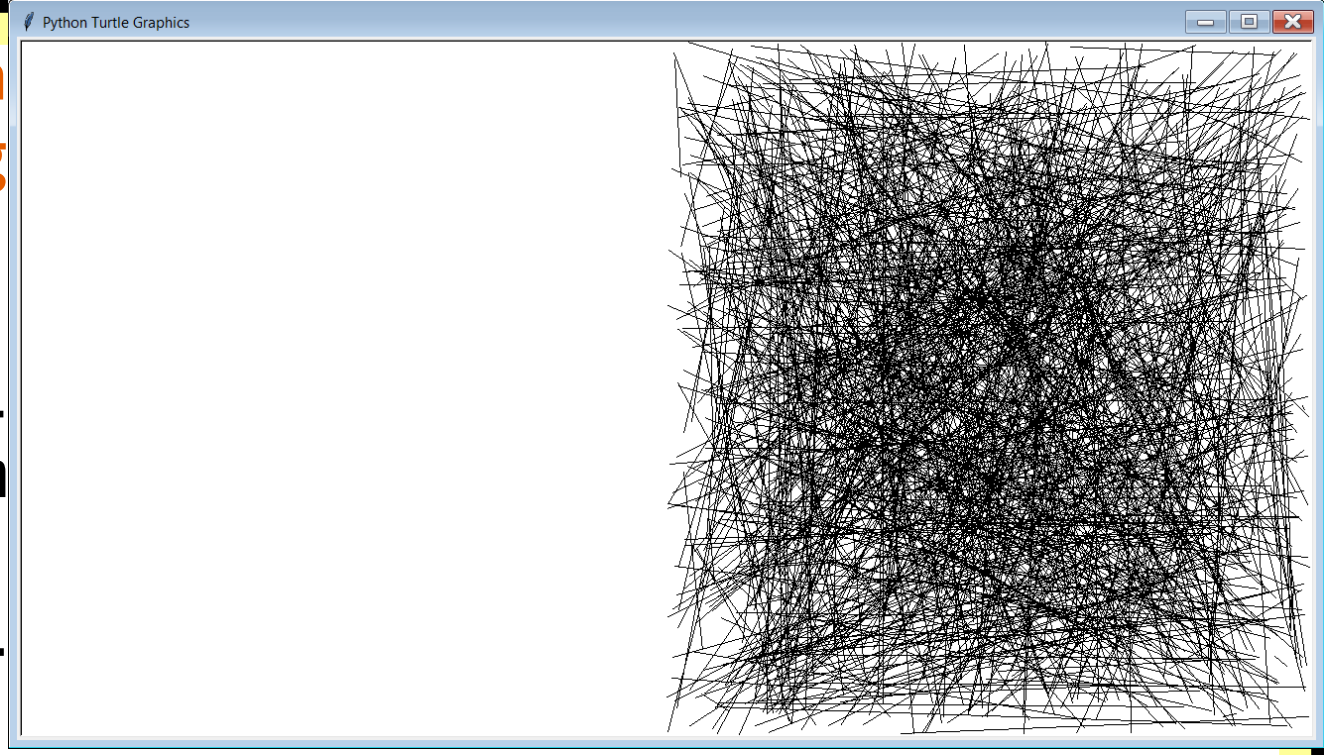
Make the random lines only show up on the left ½ of the screen.

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



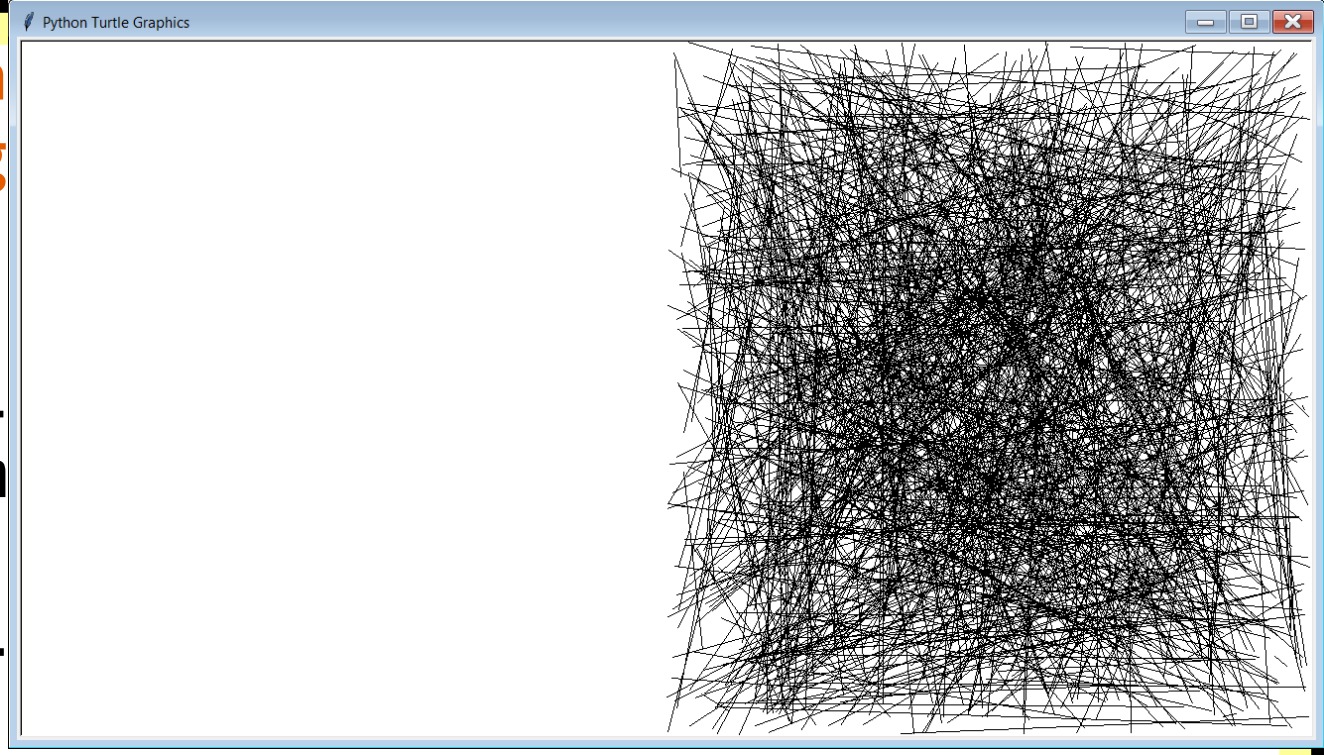
Solution

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



Solution

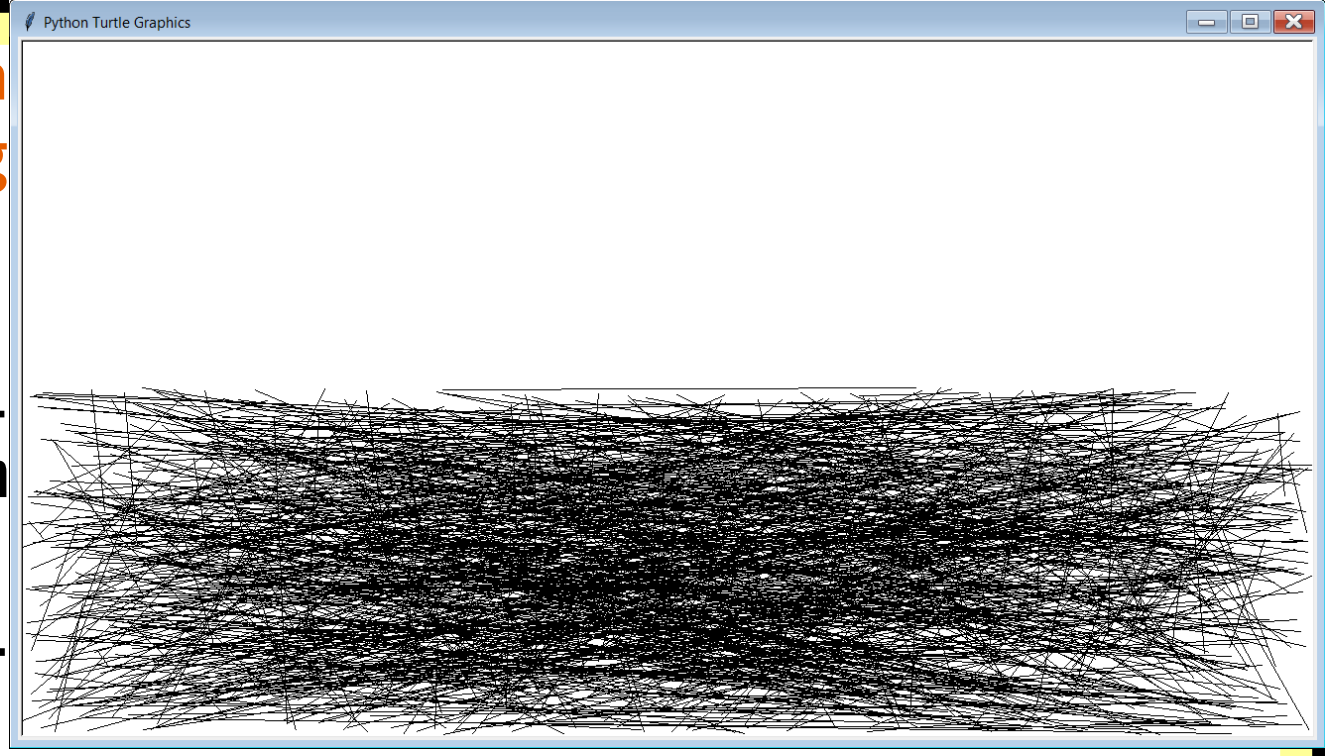
```
1  # RandomGra
2  # This prog
3
4
5  from Graphi
6  from random
7
8  beginGrfx(1
9
10 for k in range(1000):
11     x1 = randint(0,1300)
12     y1 = randint(0,700)
13     x2 = randint(0,1300)
14     y2 = randint(0,700)
15     drawLine(x1,y1,x2,y2)
16
17 endGrfx()
```

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

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

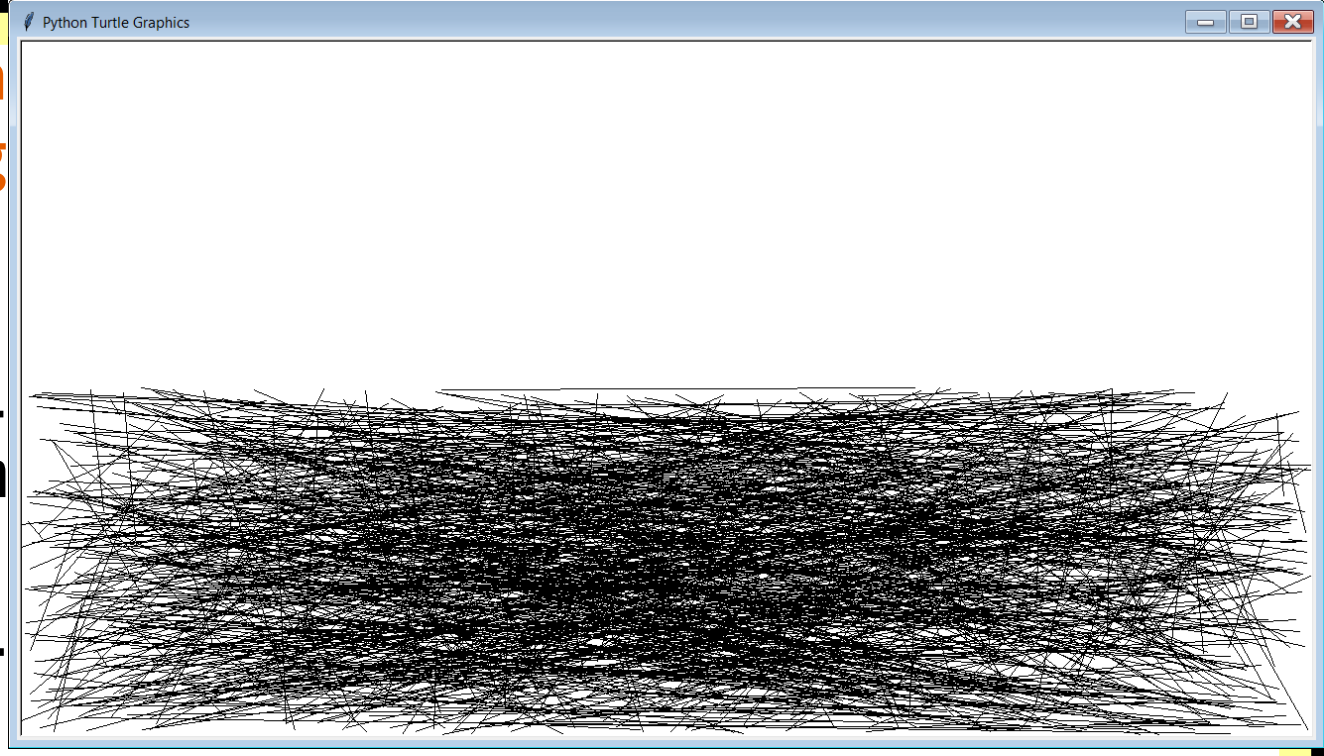
Solution

```
1  # RandomGra
2  # This prog
3
4
5  from Graphi
6  from random
7
8  beginGrfx(1
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()
```



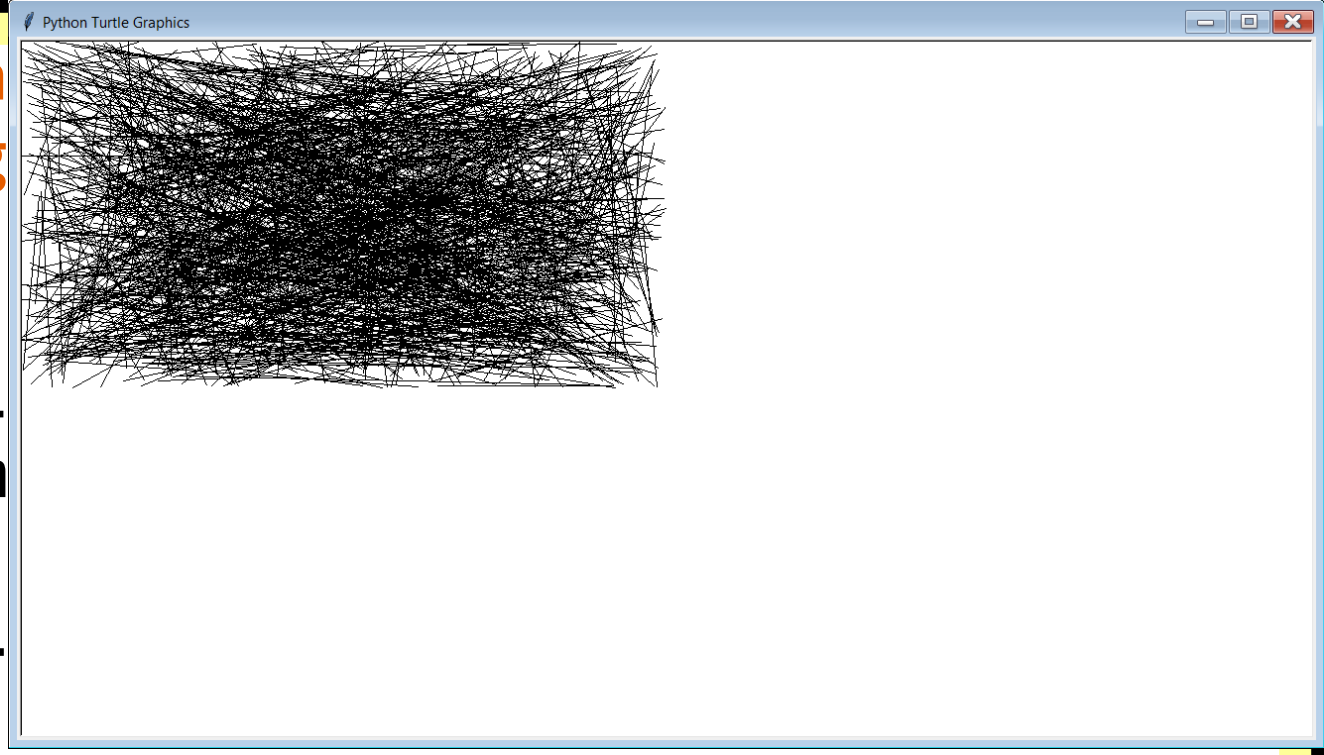
Make the random lines only show up on the bottom ½ of the screen.

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



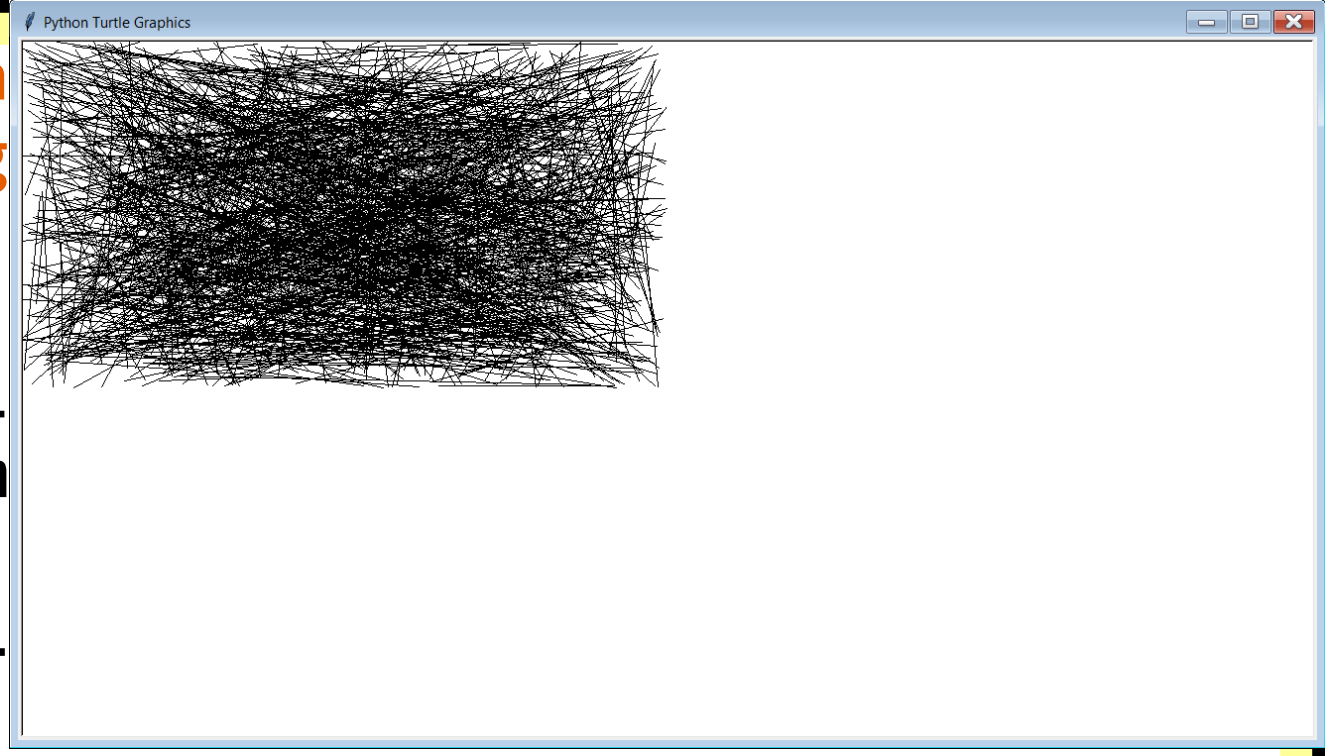
Solution

```
1  # RandomGra
2  # This prog
3
4
5  from Graphi
6  from random
7
8  beginGrfx(1
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()
```



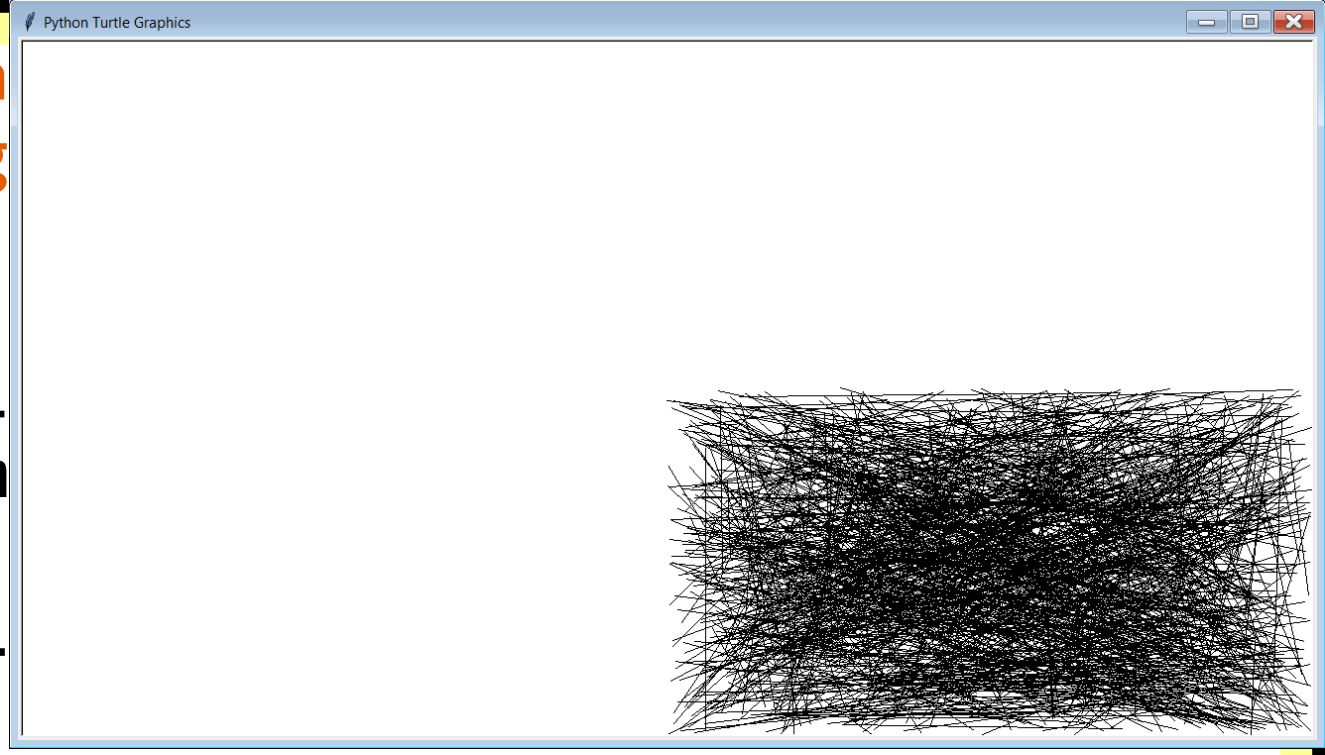
Make the random lines only show up in the top-left ¼ of the screen.

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



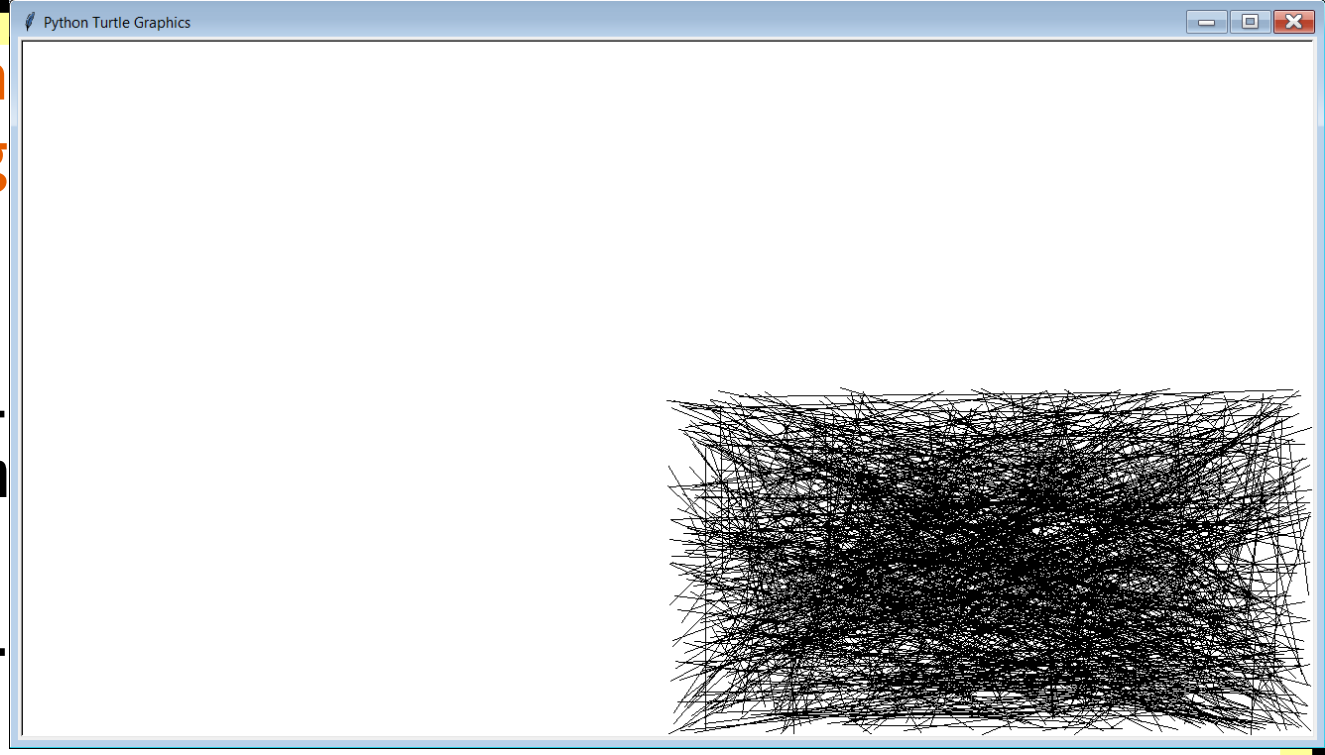
Solution

```
1  # RandomGra
2  # This prog
3
4
5  from Graphi
6  from random
7
8  beginGrfx(1
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()
```



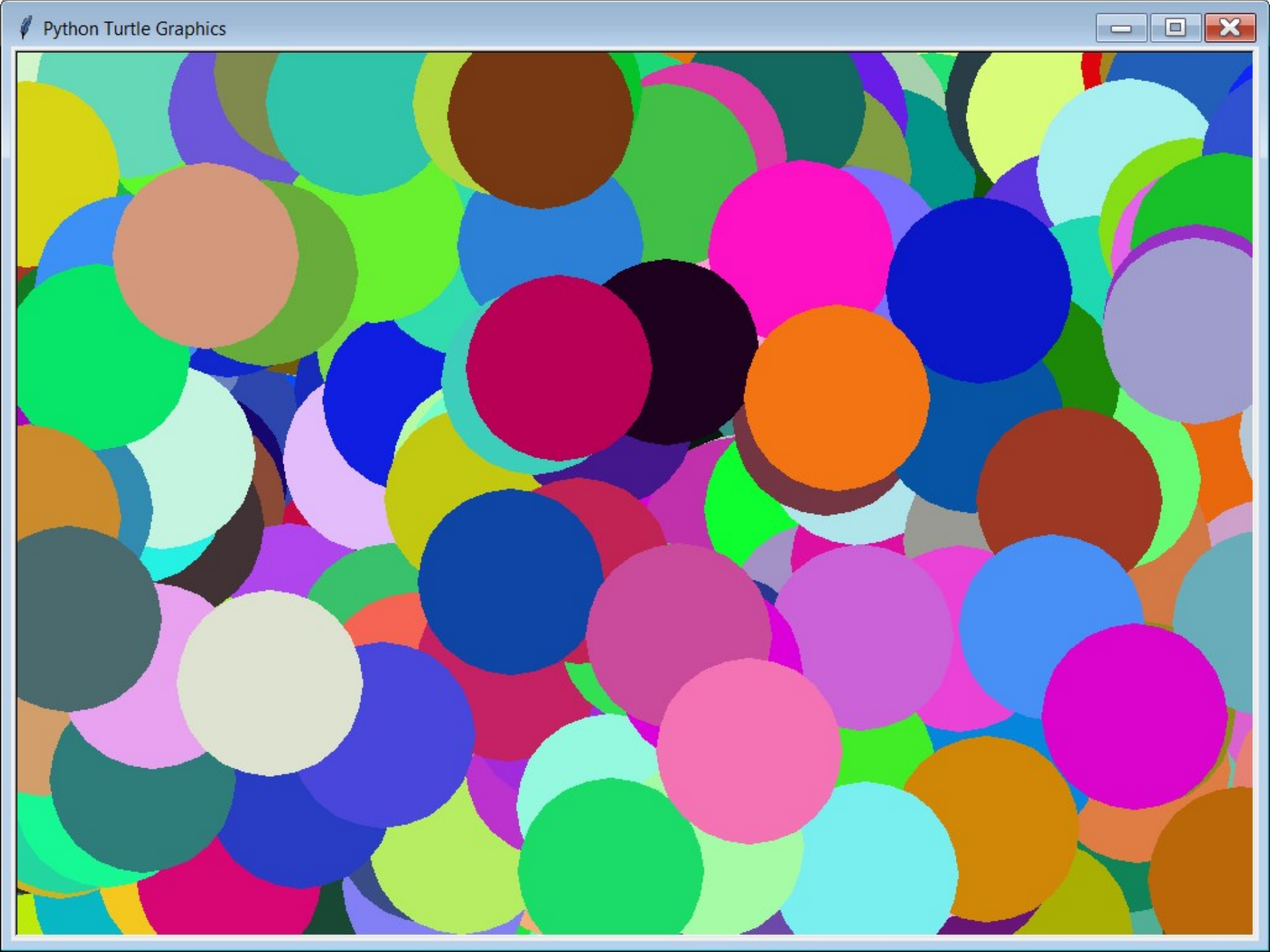
Make the random lines only show up in the bottom-right ¼ of the screen.

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

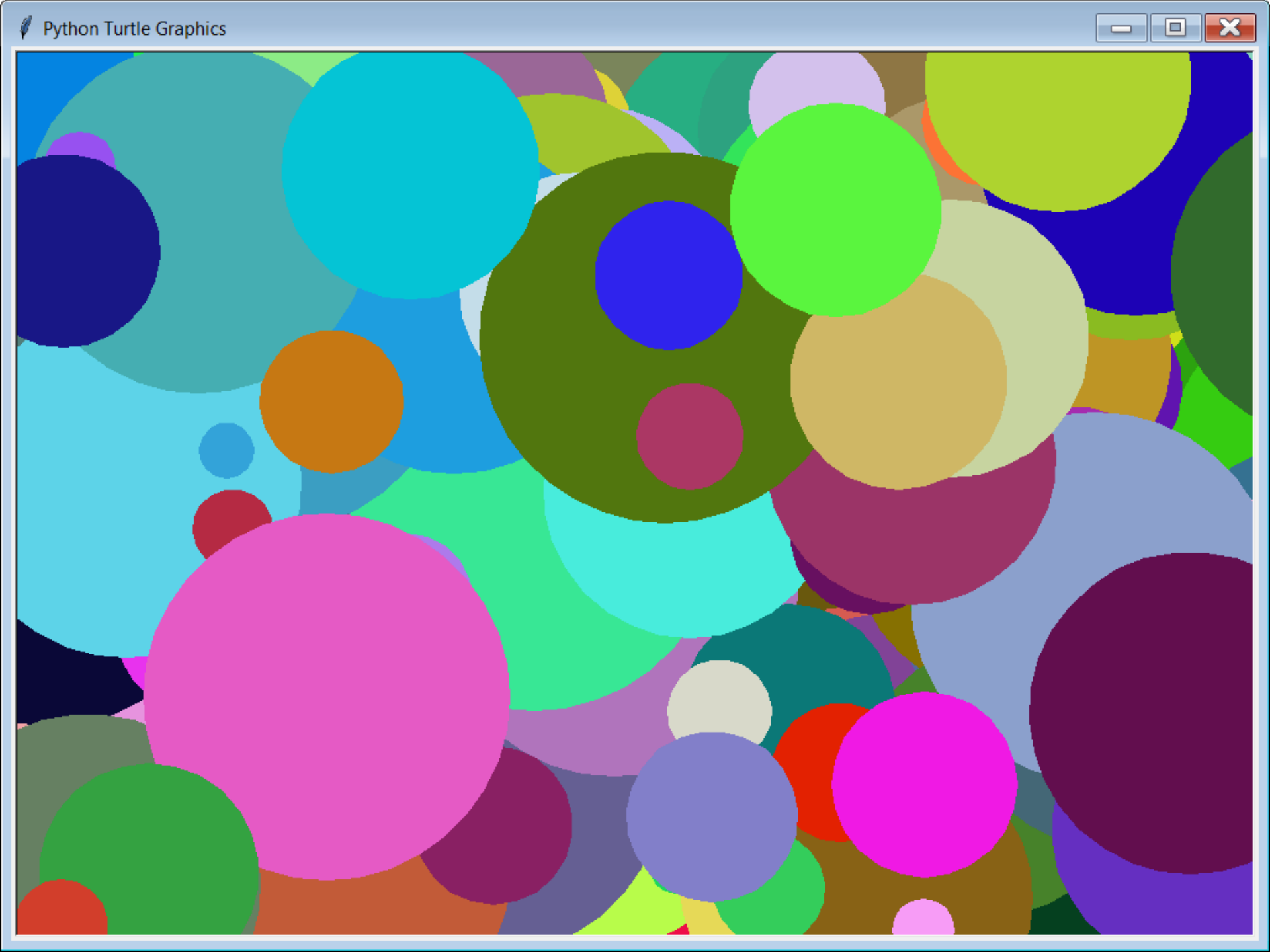


Solution

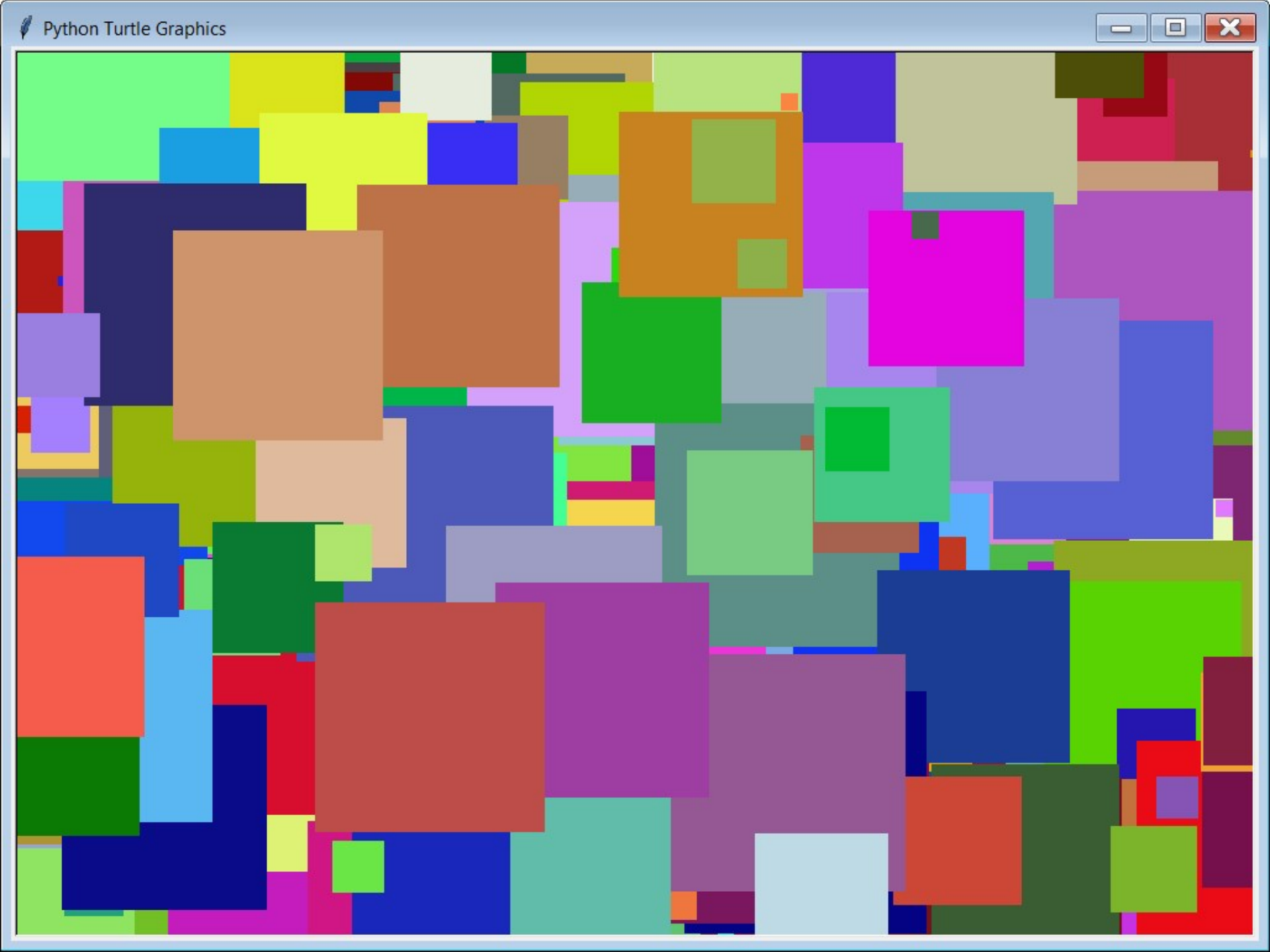
```
 1 # RandomGraphics04.py
 2 # This program displays 500 randomly colored
 3 # solid circles with a fixed radius of 75.
 4
 5
 6 from Graphics import *
 7 from random import randint
 8
 9 beginGrfx(1300,700)
10
11 for k in range(500):
12     x = randint(0,1300)
13     y = randint(0,700)
14     red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


Python Turtle Graphics

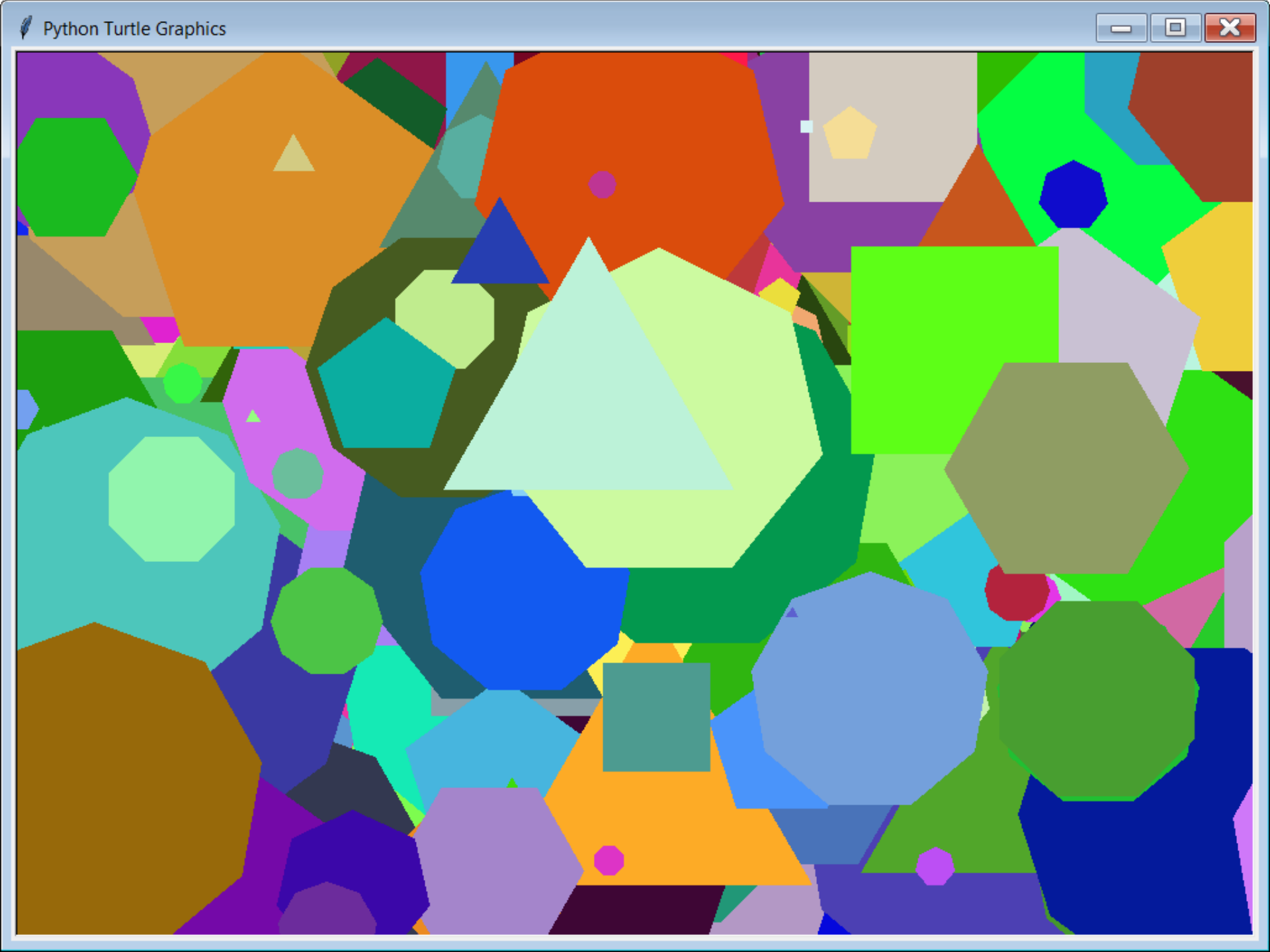
```
 1 # RandomGraphics05.py
 2 # This program displays 500 randomly colored
 3 # solid circles with random radii.
 4
 5
 6 from Graphics import *
 7 from random import randint
 8
 9 beginGrfx(1300,700)
10
11 for k in range(500):
12     x = randint(0,1300)
13     y = randint(0,700)
14     red = randint(0,255)
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()
```


Python Turtle Graphics

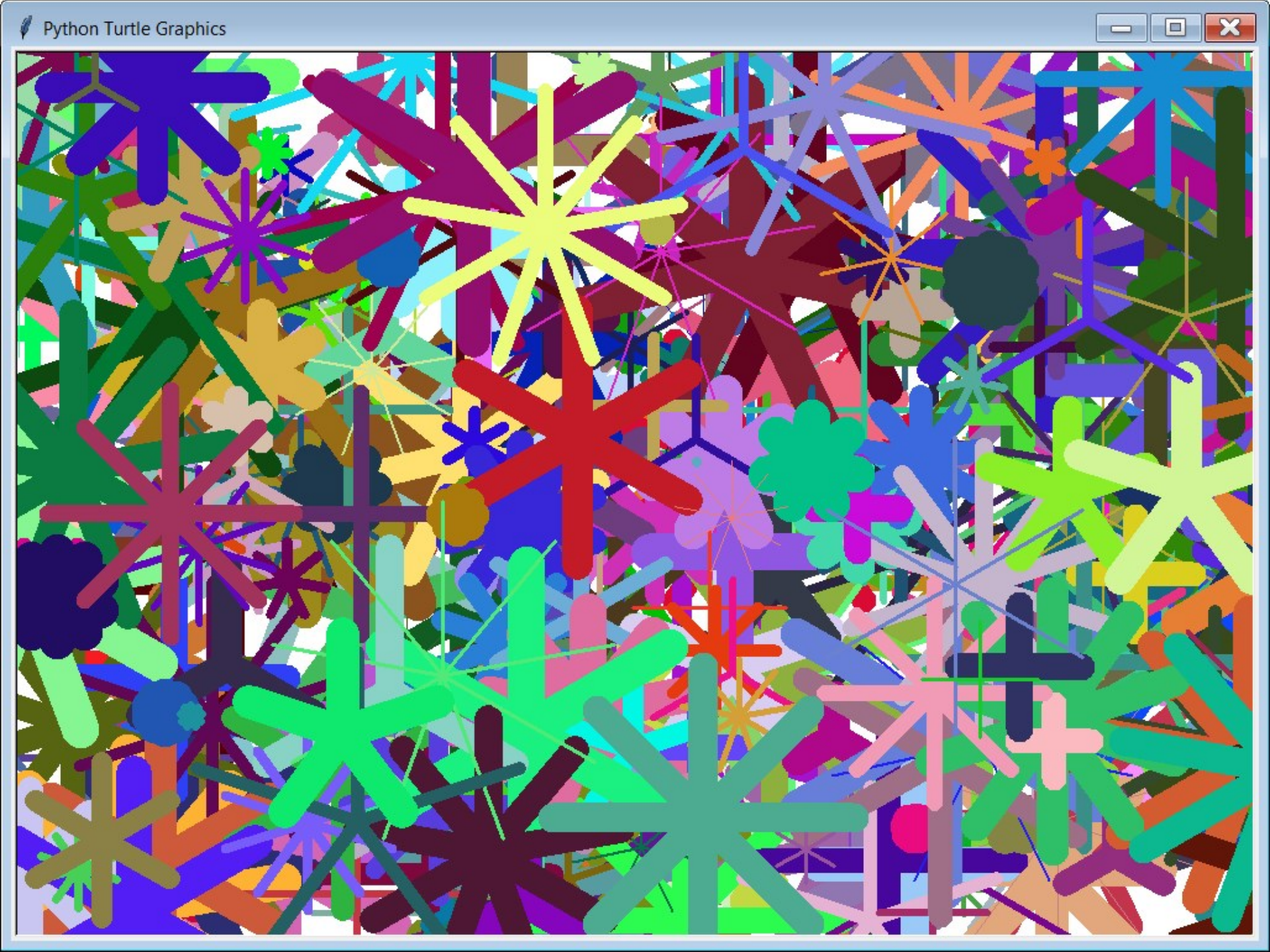
```
1  # RandomGraphics06.py
2  # This program displays 500 randomly colored squares.
3  # Creating the random colors is also simplified
4  # with the <setRandomColor> procedure.
5
6
7  from Graphics import *
8  from random import randint
9
10 beginGrfx(1300,700)
11
12 for k in range(500):
13     x = 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()
```


Python Turtle Graphics

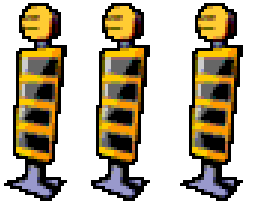
```
 1 # RandomGraphics07.py
 2 # This program displays 500 randomly colored
 3 # polygons with a random number of sides.
 4
 5
 6 from Graphics import *
 7 from random import randint
 8
 9 beginGrfx(1300,700)
10
11 for k in range(500):
12     x = randint(0,1300)
13     y = randint(0,700)
14     radius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


Python Turtle Graphics

```
 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()
```


Python Turtle Graphics

Lab 8C



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