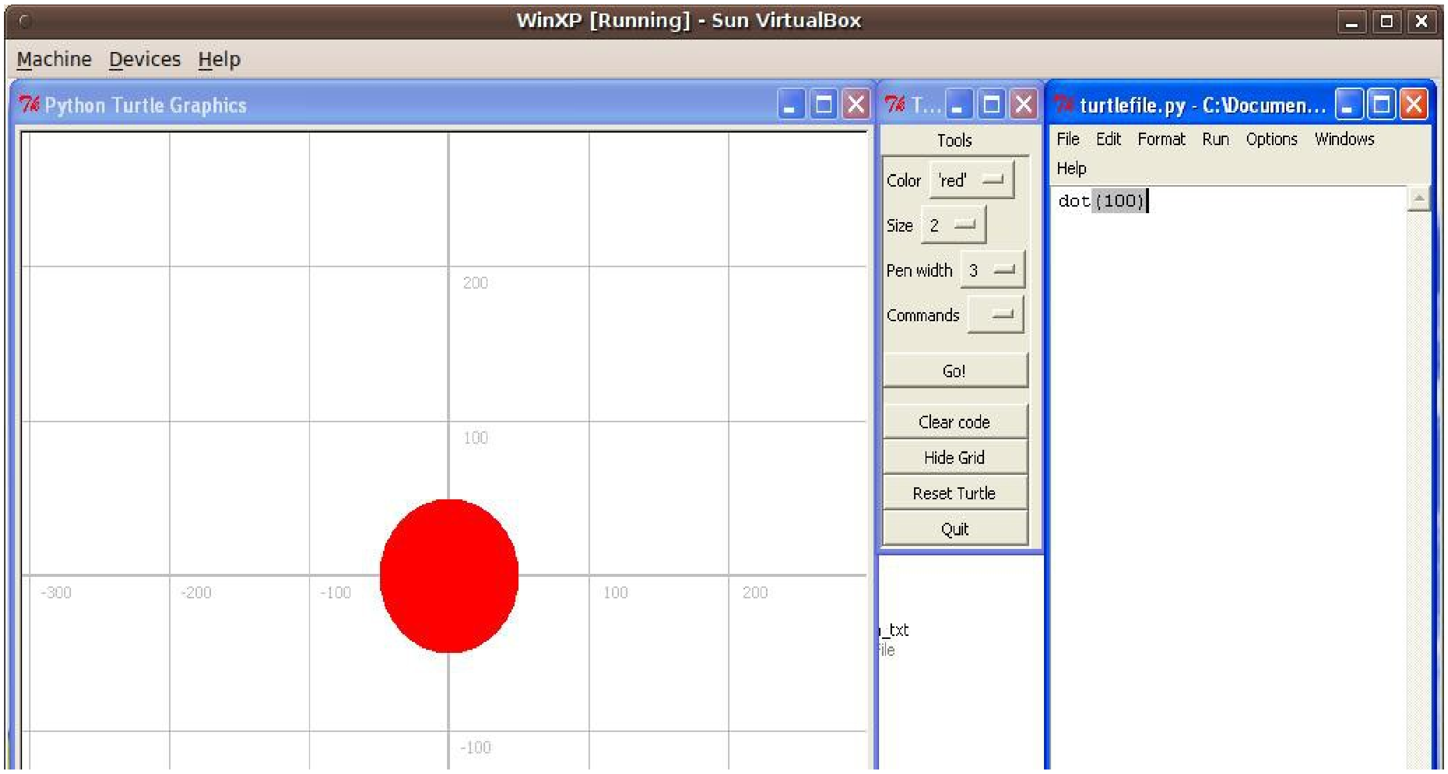
Random events

Python can generate random numbers, and in a computer program that can translate into almost anything. For example you can use the dot command to draw a big dot on the screen:

```
dot(100)
```



```
dot(100)
```

However, you can also get a random size, put it in a variable and use that:

```
dot_size = random_size(100)
dot(dot_size)
```

If you give random_size() a number, that will be the largest size it returns.

You can do the same thing with location:

```
loc = random_location()
goto(loc)
```

and color:

```
color(random_color())
dot(dot_size)
```

Exercise

1. Use random_location() and random_color() and goto() to draw a line to a random spot on the

screen. Use the Go! Button a few times to get a series of random lines.

2. Put the commands above together to put a dot of random size and color at a random place on the screen. HINT: you may want to use penup() and hideturtle() to make it look better. Use the Go! Button several times to make several different random dots.

In addition, there is a command for a random angle (random_direction()).

Repetition

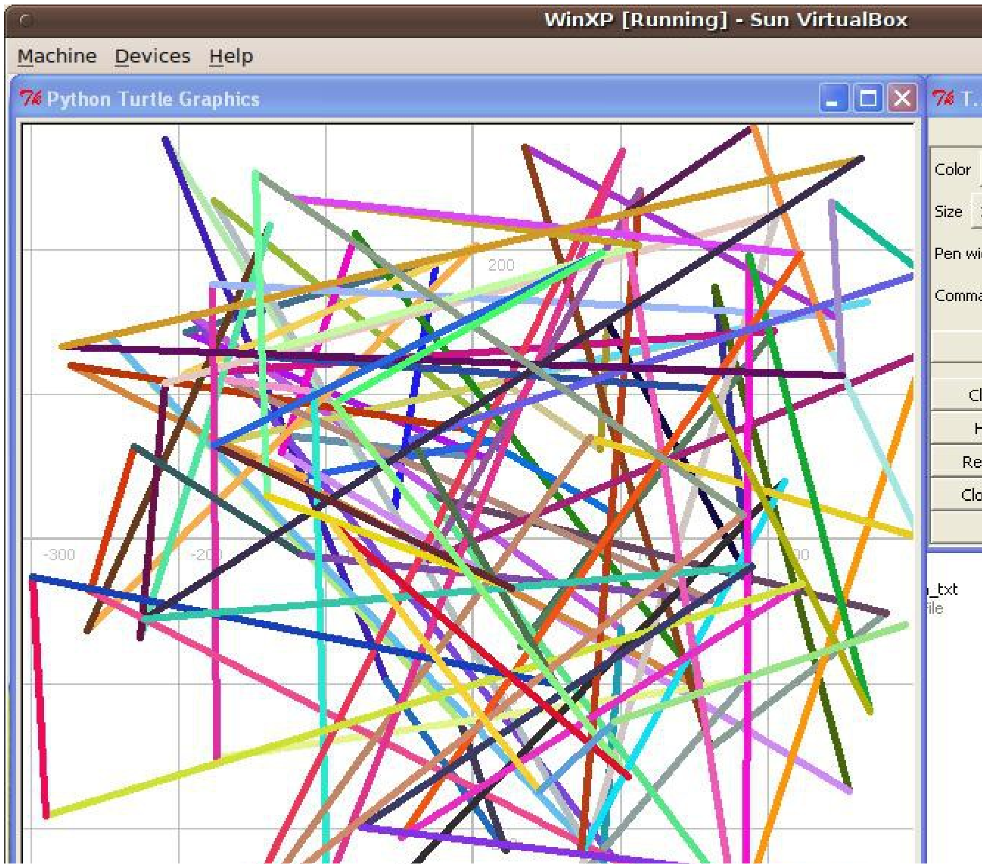
We've already used the for i in range(?): pattern to repeat some code, like drawing a side of an object. There is also a way to repeat indefinitely:

```
while True:
   # code to repeat goes here
```

This code will repeat forever, or until you close the turtle window (you sometimes will need to close the window a few times!)

Exercise

1. Put the random line code under the while True statement (be sure to indent!) and make a "screensaver" like the picture below:



2. Make the dot code into a random dot screen saver also.

To make it a more realistic screen saver, you can make the turtle window full sized and hide the lines:

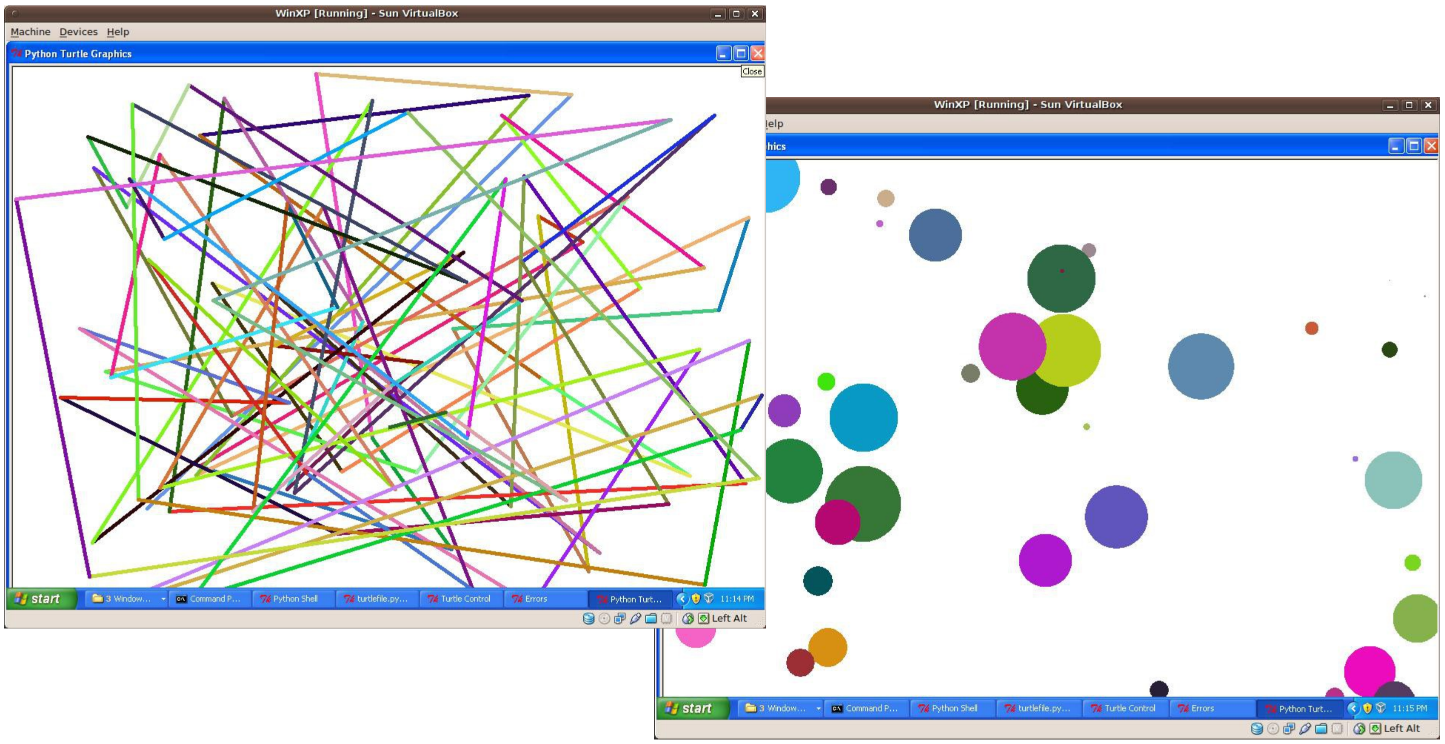
```
hide_grid()

setup(width=1.0, height=1.0)
```

You can also speed things up by using the speed() command and giving it one of the following speeds – 'fastest', 'fast', 'normal', 'slow', 'slowest'

```
speed('fastest')
```

Exercise

Make a full screen versions of your screensaver - experiment with the speed and line width, etc. as in the images below:



More Commands

If you want to draw more than one shape, you'll be annoyed to see that they are connected as you move from one space to another. The way around that is to use the penup() and pendown() commands. penup() means **STOP** drawing, as if you lifted your pen **up** off the page. pendown() puts the pen back **down** on the page so you can start drawing again.

There is also an easy way to draw circles – the circle() command. You just tell the circle command the diameter you want:

```
circle(100)
```

You can also draw just part of the circle by also telling it how much (in degrees) you want to draw. You would draw half a circle with circle(100, 180)

List of Turtle commands:

clear() - clears window

forward(distance) – moves turtle forward

backward(distance) – moves turtle backward

left(angle) – turns turtle left

right(angle) – turns turtle right

penup() - pulls pen up off page, stops drawing. It does NOT move the turtle up the page!

pendown() - puts pen down on page, starts drawing. Does not move down the page!

width(width) – sets width of line drawn

color(*args) - *args can either be a color name in quotes, like “green” or a combination of red, green and blue values between 0 and 1. For example black is either color(“black”) or color(0,0,0), white is either color(“white”) or color(1,1,1), gray would be color (“gray”) or color(.5, .5, .5), and purple might be color(.5, 0, .5)

write(text) – will write the text at the current location

begin_fill(), end_fill() – use begin_fill() before starting a shape you want filled (solid) and end_fill() at the end

circle(radius, extent) – draws a circle with the given radius. Extent is not used unless you want only a partial cricle, then extent is the number of degrees that will be drawn.

dot(radius) – draws a dot with the given radius.

goto(x,y) – will move the turtle to location x, y. The center of the screen is 0,0.

towards(x,y) – gives the angle to location x, y. The center of the screen is 0,0.

heading() - will give the direction in degrees that the turtle is facing. Right is 0, up is 90, left is 180, down is 270, etc.

setheading(angle) – will set the direction in degrees that the turtle is facing. Right is 0, up is 90, left is 180, down is 270, etc.