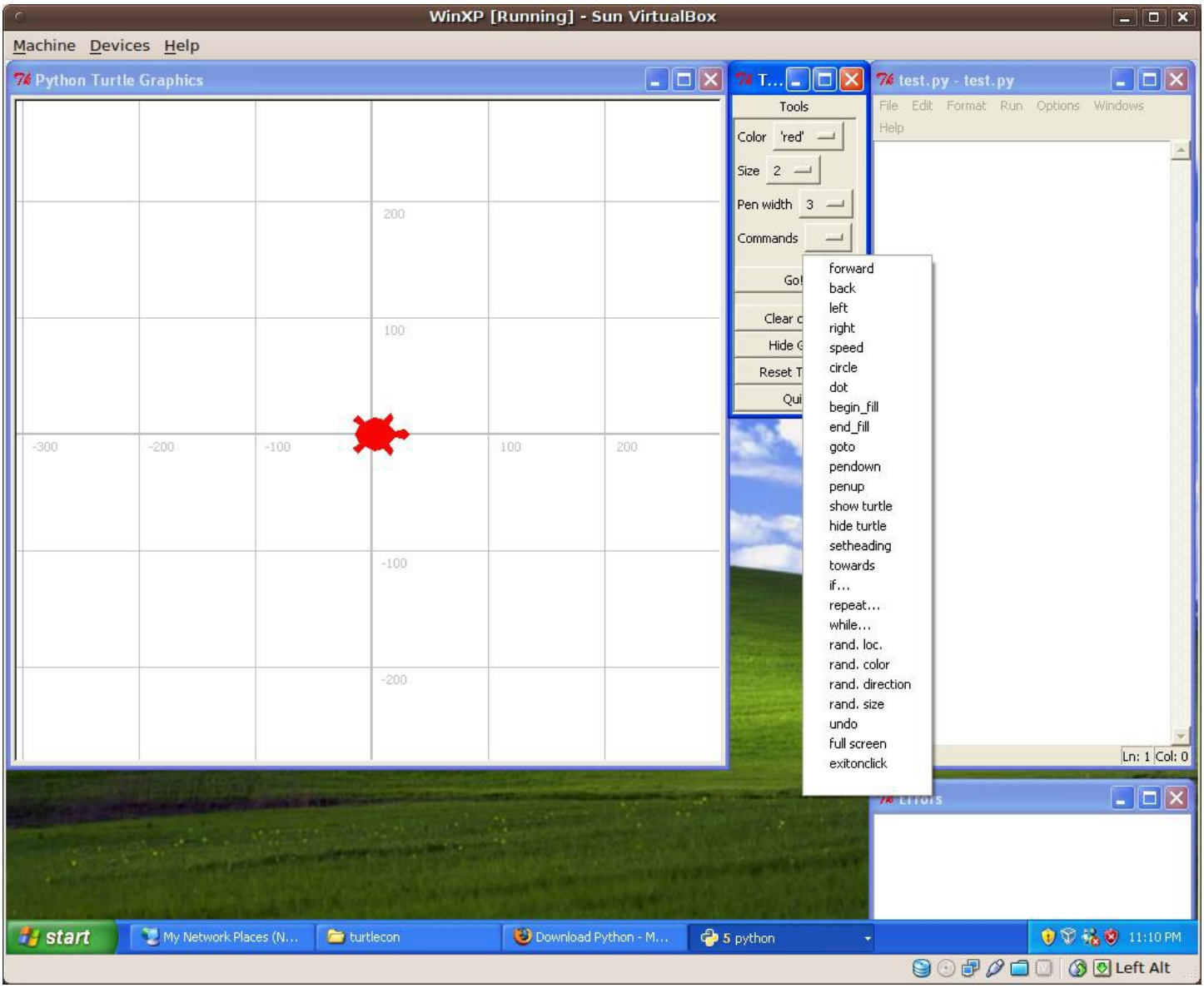
Turtle graphics – the easy way

Python comes with some support for graphics. To get started, you need to run `turtlecon.py`, which will give you a control panel of commands and a new editor window.

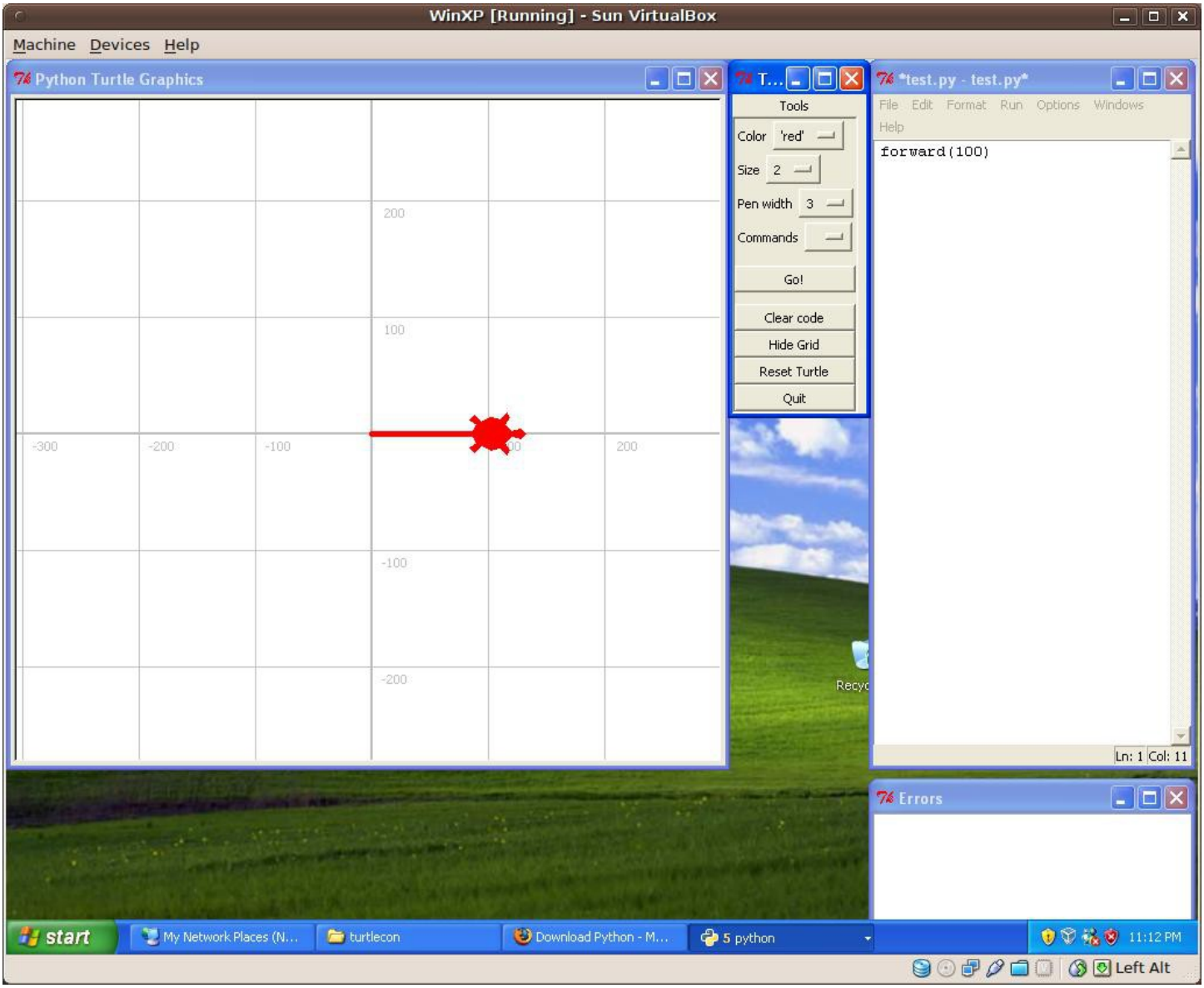


The pointer that Python uses to indicate where the pen is and which direction it's facing iscalled a “turtle”, so we'll use one shaped like a turtle. (It can also be hidden, if you want.) As the turtle draws things you can see it move.

For your first graphics program, let's draw a line. Use the control panel or type in:

```
forward(100)
```

then, run your program with by hitting the “Go!” button.



The “turtle” starts in the middle, facing right, and then follows the commands given it.

Note that you can repeat the command by hitting the Go! button again.

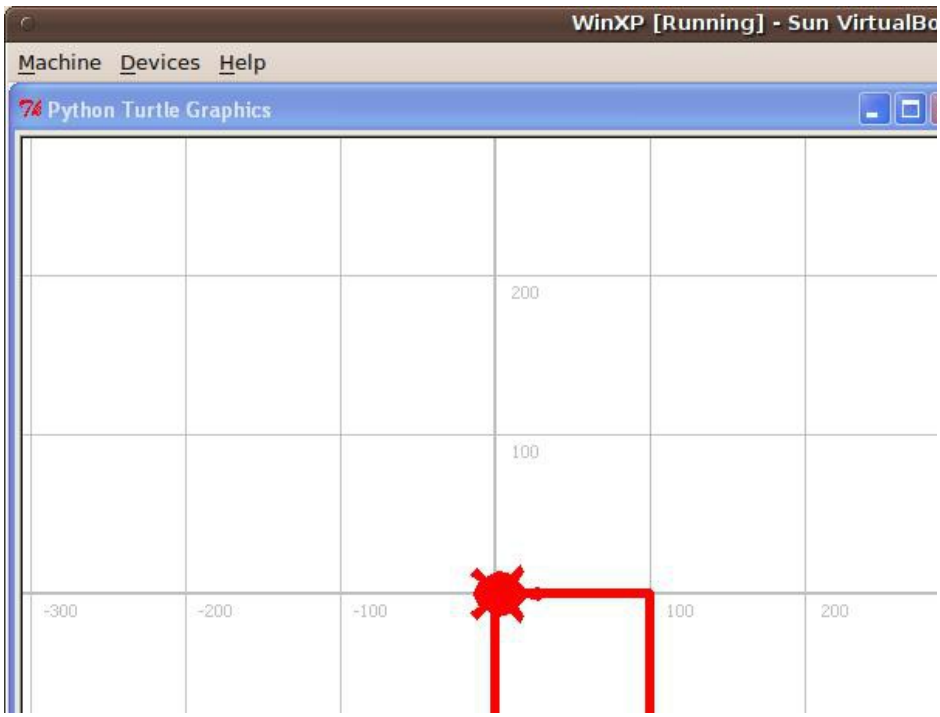
Exercises

1. Add a 90 degree right turn and hit the Go! button enough times to make a square:
2. Modify the code above to make a triangle. (Use the 'Reset turtle' button to clear the screen and reset the turtle.

The above works just fine, but it needs to be repeated 4 times. Using a for loop (the 'repeat...' button) we can make it even easier to repeat the same code 4 times:

```
for i in range(4):
    forward(100)
```

```
right(90)
```



Exercise

3. Use the code above to make a square

In fact, there is an easy formula for making shapes:

```
for i in range(number_of_sides):
    forward(length_of_side)
    right(360/number_of_sides)
```

All you have to do is replace *number_of_sides* and *length_of_side* with the values you want to use.

You can also make your shapes solid, rather than outlines by giving the **begin_fill()** command before drawing and the **end_fill()** command after drawing.

To make a solid octagon with each side 75 pixels long use:

```
begin_fill()
for i in range(8):
    forward(75)
    right(360/8)
end_fill()
```

Exercises

1. use the method above to make solid shapes with 5, 6, and 12 sides
2. use the method above to draw a shape with 360 sides and a side length of 1. You now know one way to make a circle.

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.

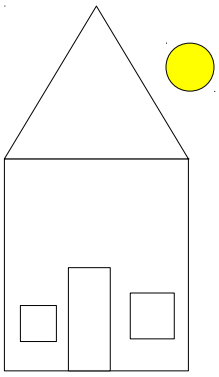
There is also an easier way to draw circles – the circle() command. You just tell the circle command the diameter you want:

```
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)

Exercise

1. Experiment with the commands below to create a simple picture of a house with a door and two windows (and whatever else you can think of). **Hint:** you will need to use the up() and down() commands to lift the "pen" off of the page and put it back down.



List of Turtle commands:

clear() - clears window

forward(distance) – moves turtle foward

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.