Turtle Graphics for Python

Idea

You have a small turtle that moves around the screen.

As it moves, it leaves a trail.

You can tell it to hold its pen up and then it moves wihtout leaving a trail.

Pen down lets it resume making a visible trail.

It can turn left or right. When you next tell it to move, it goes in the direction it faces.

Using Turtle Graphics in Python

Define your turtle functions before using them.

This line goes first in your program:

```
from turtle import *
```

Remember the principle

Define first, then use

You also :

Import first, then use

Move the turtle:

Move the turtle forward 100 units:

```
forward(100)
```

Move the turtle back 20 units:

```
back(20)
```

Turn the turtle:

Turn the turtle right 90 degrees:

```
right(90)
```

Degree Facts
90 degrees = a right turn, such as in a square
45 degrees = a diagonal across a square
360 degrees = go full circle to back where you started

Turn the turtle left 45 degrees:

```
left(45)
```

Pen up (no trail), Pen Down (trail)

pendown – turtle moves without a trail on screen

```
pendown()
```

penup – turtle moves showing a trail on screen

```
penup()
```

```
penup()       # leave no trail
forward(20)   # move 20 steps

pendown()     # moves will now show
forward(30)   # leave a line on screen
```

Our Python program:

IDLE File Edit Format Run Window Help

turtle00.py – /Users/kuahiwi/

```
from turtle import *

# Move turtle, then turn, move again

forward(100)
right(90)
forward(60)
```

Results on the screen:

Move Turtle without drawing a line

```
# move 40, no line:
penup()
forward(40)
pendown()
```

Define a function to move turtle without drawing a line

```
def move_turtle(distance):
    ''' Move turtle, no line drawn '''
    penup()
    forward(distance)
    pendown()
```

Use move_turtle function (defined above)

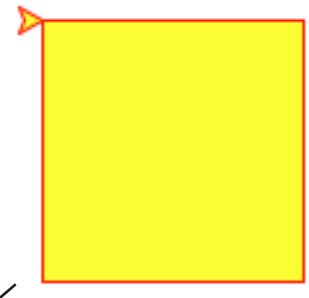
```
move_turtle(15)
```

or

```
this_far = 20
move_turtle(this_far)
```

Draw a circle, radius 75:

```
circle(75)
```



Add some color

Color the lines red, the area fill yellow

```
color("red", "yellow")
forward(100)  # line is red
```

Fill a closed area with yellow

```
color("red", "yellow")
begin_fill()
# draw square
for i in range(4):
    forward(100)
    right(90)
end_fill()   # now yellow fills square
```

```
from turtle import *

def draw_square(side):
      ''' Draw a square
      length is side units long
      '''
      # repeat 4 times
      for i in range(4):
            forward(side)
            right(90)
      return

...

draw_square(150)

...

size = 175
draw_square(size)
```

Here's one way to draw a square with Turtle Graphics.

Notice that a square has 4 sides and it turns right 90 degrees, or 1/4th of 360 degrees.

Can you create a similar function definition that will draw a polygon with n sides, each of length side?

Hint:

Give the new function a name such as draw_poly, then decide on what inputs it needs.

Figure out changes to the function.

Test.

Can you draw a polygon of 4 sides? 6 sides? 8 sides (like a stop sign)?

Control Speed of Drawing

```
speed(1)  # slowest speed


speed(10) # fast
```

You can use numbers from 1 to 10 to get the speed you want; you see the little "turtle" while it draws.

Fastest - but no turtle animation

```
speed(0)  # best for very complex
          # drawings
```

Start a fresh drawing, turtle in standard starting location and direction

```
reset()  # Start a fresh drawing
         # Erases any drawing on screen
```

It's a good idea to do a reset() before doing any drawing. Gives you a clean start.

For a lot more detail on Turtle Graphics, go to http://docs.python.org/release/3.2/library/turtle.html

Macs - set up for Turtle Graphics

We need to run

`IDLE -n`

Running IDLE -n prevents some problems with Turtle Graphics

Open Terminal, set it aside for a moment

Finder (in Dock)
Go to Applications
Go to Python 3.2 folder

Right-click (or Control-Click) on IDLE.app

Choose Show package contents

Look in MacOS folder

Drag IDLE icon to Terminal window

Add -n to the end of /IDLE

Your terminal window now says
Long-path-to-**`IDLE -n`**

You now see a
`No Subprocess`
note with the IDLE start-up screen

As you normally do, go to
IDLE's ***File*** menu,

New Window

Now you can create your graphic.py program using Turtle Graphics, and you will see it all correctly.