Python...

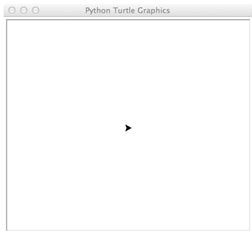
Python is an Object Oriented programming language that works within an interpreter... An example:

```
$ python
>>> 1000 / 10.2
98.03921568627452
>>>
```

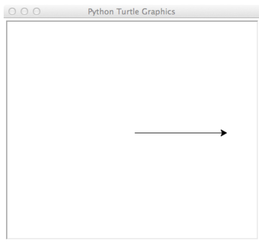
Another aspect is the use of graphics; a very simple drawing package is the turtle module, which provides a very simple set of graphics primitives. The following program draws a simple line:

```
$ python
Python 2.7.1 (r271:86832, Jul 31 2011, 19:30:53)
[GCC 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2335.15.00)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import turtle
>>> mTurtle = turtle.Turtle()
```

This generates:



mTurtle.forward(length)

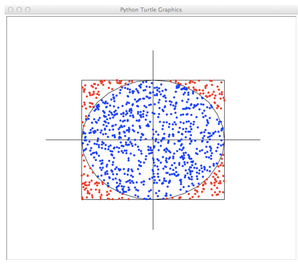


Below is an example of the dart program in Python that we previously done in Scratch and Java that generated an approximated the value of Pi.

Please listen to the lecture in class carefully as there is a lot going on here, which is fairly simple once the concepts are understood.

```
import random
import math
import turtle

def     darts(numberOfDarts):
        wn = turtle.Screen()
        t = turtle.Turtle()

        wn.setworldcoordinates(-200, -200, 200, 200)

        t.up()
        t.goto(-150, 0)
        t.down()
        t.goto(150, 0)

        t.up()
        t.goto(0,150)
        t.down()
        t.goto(0,-150)

        t.up()
        t.goto(0,-100)  # from drawing the circle

        t.down()
        t.circle(100)

        t.up()                  # draw a border around the circle
        t.goto(-100, 100)
        t.down()
        t.goto(100, 100)
        t.goto(100, -100)
        t.goto(-100, -100)
        t.goto(-100, 100)

        hits = 0

        t.up()

        for i in range(numberOfDarts):
                x = random.random() * 200 - 100  # want -100 to 100
                y = random.random() * 200 - 100

                distance = math.sqrt(x**2 + y**2)

                t.goto(x,y)

                if distance <= 100:
                        hits = hits + 1
                        t.color("blue")
                else:
                        t.color("red")

                t.dot()

        print "hits: ", hits
        print "number of darts: " , numberOfDarts

        pi = (hits*1.0)/numberOfDarts * 4

        print pi

        wn.exitonclick()

        return pi
```

Sample run:



One downside of the above program is that it takes a long time to run -- about 4 minutes on an Intel Core 2 Duo

One aspect of computer science is creating efficient, scalable systems, which can take a fair amount of time and effort -- often there are parts of a system that take most of the time, i.e. make it overly slow and unusable. One way to determine where bottlenecks exist is to time various aspects of the system. Below is the timing of throwing 1,000,000 darts, without a GUI...

Does the following indicate anything signifiant?

```
>>> import time
>>> time.asctime(time.gmtime()); darts_nogui(10000000); time.asctime(time.gmtime())
'Sun Apr 27 21:22:28 2014'
hits:  7853581
number of darts:  10000000
3.1414324
3.1414324
'Sun Apr 27 21:22:45 2014'
>>>
```

- Without the Graphical User Interface (GUI), the program only took 17 seconds to simulate throwing 1,000,000 darts!
- Python docs: http://docs.python.org