Functions and Modules
Functions in Python

- A function in math, often denoted $f : X \to Y$, associates with $x$ in $X$ a unique value $f(x)$ in $Y$.

- **Examples:** (a) $f(x) = x^2$. Here $x$ can be any real number and $f(x)$ is a non-negative real number.
  
  $f(3) = 9, \ f(-1.1) = 1.21, \ f(15) = 225$, etc.

(b) $f(x) = \sqrt{x}$. Here $x$ can be any *positive* real number and $f(x)$ is a positive real number.

  $f(25) = 5, \ f(100) = 10, \ f(20) = 4.47213$, etc.

- $x$ is called the *argument* to the function $f$.
- We are also taught to sometimes view $f : X \to Y$ as a “black box” to which you provide an $x$ as input and out comes $f(x)$.
Most programming languages provide ways of defining the *computational* equivalent of this.

For example, the *math* module contains the definition of a function called *sqrt*.

This is a piece of Python code that, when given the value of an *argument*, computes and returns the square root of that argument.

This allows us to write code such as:

```python
factorBound = math.sqrt(n)
```
One way to categorize functions in Python is:

1. **Built-in functions**: these functions are pre-defined and are always available.

2. **Functions defined in modules**: these functions are pre-defined in particular modules and can only be used when the corresponding module is imported.

3. **User defined functions**: these are defined by the programmer.
Python documentation lists 80 built-in functions at: http://docs.python.org/library/functions.html

- Math functions: abs(x), round(x, n)
- Type conversion functions: bool(x), float(x), int(x), long(x), str(x)
- Input functions: raw_input(x), input(x)
- Miscellaneous: len(x), id(x)
The function `input(prompt)` treats what the user types as input as a Python expression and returns the evaluated value.

I prefer `raw_input(prompt)` to `input(prompt)` in general because it gives the programmer more control on how to interpret the input.

`input(prompt)` is okay when all you are expecting is simple numeric input.

In Python version 3, `raw_input(prompt)` has been renamed as `input(prompt)`. 
Functions in modules

- The modules we have used so far are: `sys`, `math`, `time`

- There are 100s of “standard” modules in Python:
  - Generation of random numbers and probability distributions
  - Accessing files and directories
  - Web development
  - Network programming
  - Graphics, etc.

- A module is simply a file (just like the files that you have been creating your programs in) that contains related Python statements and function definitions.

- Programmers can define their own modules. There are 1000s of third-party modules available for Python.
Importing from modules

- We have used statements of the form
  ```python
  import math
  ```
  to import from modules.
- When we import a module $X$ in this manner, we need to use $X.name$ to refer to an item called $name$ that is defined in the module $X$.
- **Examples:** `math.sqrt(25)` or `math.pi`
- There are some other ways of importing from modules as well.
Another way of importing from modules

- You can also use
  ```python
  from X import *
  ```
- In this case, you can directly refer to items in the module `X`, without using the “`X.`” prefix.
- Try
  ```python
  from math import *
  ```
  You can use `sqrt(35)` or `pi` or `e` without the “`math.`” prefix.
- Beware of new items (variables, functions, etc.) that you don’t know about coming into existence.
The random module

- Programs for games and simulation use *randomization* extensively.

- In games, you want to add an element of randomness to the obstacles or adversaries.

- In simulations (e.g., traffic simulation) you want to introduce actors into your simulation according to certain probability distribution.
Some functions in the `random` module

- `random.randint(a, b)`: return a random integer $N$ such that $a \leq N \leq b$.

- `random.random()`: Return the next random floating point number in the range $[0.0, 1.0)$.

- `random.uniform(a, b)`: Return a random floating point number $N$ such that $a \leq N \leq b$ for $a \leq b$ and $b \leq N \leq a$ for $b < a$. 

Is Python’s coin (die) unbiased?

- **Problem**: Write a program that takes as input a positive integer $n$ and reports the number of heads that appear when a coin is tossed $n$ times.

- **Problem**: Roll a 6-sided die $n$ times, where $n$ is a positive integer provided as input, and report the number of times each outcome appears.
If we take a random walk, will we go places?

**Problem:** Simulate a *random walk* in which a person starts at point 0 and at each step randomly picks a direction (left or right) and moves 1 step in that direction.

- Take a positive integer $n$ and terminate the simulation when the walk reaches $n$ or $-n$.
- Report the average number of steps it took for the walk to terminate.
- Do this for various $n$ and plot the results to get a sense of how rapidly the walk terminates, as a function of $n$. 