Boston Dynamics’ sprinting robot, Cheetah, has now broken the land-speed record for humans, clocking speeds of 29.3 miles per hour, meaning not even the 27.79mph Usain Bolt can escape. Fortunately for us, it's got a fatal flaw; a balance problem that means it can only remain upright with a boom keeping it steady. Unfortunately, that's not going to be a problem for long, since field-testing on an independently upright version begins early next year.

• Lab: don’t worry precision of $-3.3333333333$
  – We’ll come back to this

• Difference between interpreter and IDLE editor?
• Lab 1 graded
  – Hand in .py files

• Lab 2 continuing today
• Other questions about lab?
Practice: Distance Averager

• Problem:
  – Our poor European study-abroad student needs to know average distance between three cities, in miles
  – 1 Kilometer = 0.62 Mile

Analyze the Problem
Determine Specification
Create a Design
Implement Design
Test/Debug the Program
Maintain Program
Numeric Data Types (Chapter 3)

• Info stored & manipulated by programs is *data*

• Two kinds of numbers:
  – Whole numbers (5, 4, 3, 6) don’t have fractional part
  – Decimal numbers (3.25, 2.10, 1.05, 0.01) do
Numeric Data Types

• decimal numbers and whole numbers are two different *data types*

• The *data type* of an object determines what values it can have and what operations can be performed on it.
Numeric Data Types

- Whole numbers are represented using the `integer (int)` data type
- Values can be positive or negative
Numeric Data Types

• Numbers that *can* have fractional parts are represented as *floating point* (or *float*) values

• Which is which?
  – Numeric literal without a decimal point: an int
  – Literal with decimal point is: float
    • (even if the fractional part is 0)

Which witch is which?
type(val)

• Python has a special function to tell us the data type of any value.

```python
>>> type(3)
<class 'int'>
>>> type(3.1)
<class 'float'>
>>> type(3.0)
<class 'float'>
>>> myInt = 32
>>> type(myInt)
<class 'int'>
```
Numeric Data Types

• Why do we need two number types?
  – Values that represent counts can’t be fractional
    • you can’t have 3 ½ quarters
    • can you ever half ½ a piece of chalk?
  – Most mathematical algorithms very efficient with integers
  – The float type stores only approximation to the actual number being represented!
    • 1/3
    • 3.0 + 1/3
  – Float: Slow & inexact, but sometimes necessary
Numeric Data Types

• Operations on ints produce ints, operations on floats produce floats (except for /).

```python
>>> 3.0+4.0
7.0
>>> 3+4
7
>>> 3.0*4.0
12.0
>>> 3*4
12
>>> 10.0/3.0
3.3333333333333335
>>> 10/3
3.3333333333333335
>>> 10 // 3
3
>>> 10.0 // 3.0
3.0
```
Numeric Data Types

• Integer division produces a whole number: \(10 \div 3 = 3\)

• Mod produces the remainder (of ints): \(10 \% 3 = 1\)
a = (a//b)(b) + (a\%b)
Cash Register

• Input: Number of cents, from 0 to 99
• Output: Number of pennies and quarters

Create a Design
Implement Design
Test the Program
Using the Math Library

• Besides (+, -, *, /, //, **, %, abs), we have lots of math functions available in the **math library**

• **library**: a module with definitions/functions
Using the Math Library

• Program to compute roots of a quadratic equation?

\[ x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \]

• Don’t know how to find a square root… but it’s in the math library
Using the Math Library

• To use a library, add this line:

```python
import math
```

• Importing a library makes whatever functions are defined within it available to the program.

• Key point: want to use others’ code when possible.
Using the Math Library

• To access the \texttt{sqrt} library routine:
  \begin{itemize}
  \item \texttt{math.sqrt(x)}
  \end{itemize}

• Using \texttt{dot notation} tells Python to use \texttt{sqrt} function found in math library module
Using `math.sqrt()`

\[ x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \]

\( x_1 \) = ?

\( x_2 \) = ?
Help!

• One nice feature of python is the help mechanism
• As we start doing more complex things it will be useful
>>> import math
>>> help(math)
Help on built-in module math:

NAME
   math

FILE
   (built-in)

DESCRIPTION
   This module is always available. It provides access to the
   mathematical functions defined by the C standard.

FUNCTIONS
   acos(...)
       acos(x)
• (10 % 4) + (6 / 2)
• (10 % (4+6)) / 2
• ((3 ** 3) // 3) % 3

• Find the sum of the first n integers, where n is specified by the user
• Sum a series of numbers entered by the user. First prompt for how many numbers there will be then let them input the numbers. Finally, print the sum.