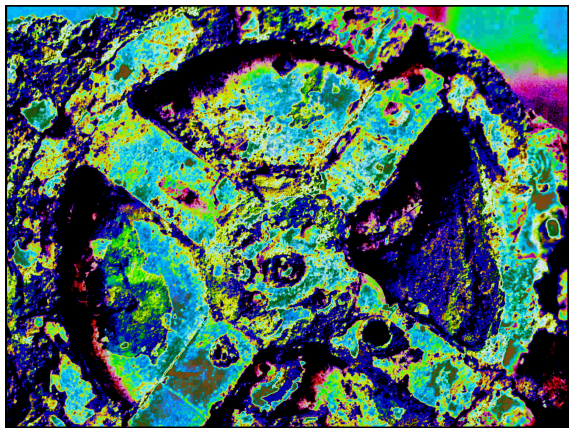


CS101 Computer Science I

Fall 2013

Robert Muller
Boston College

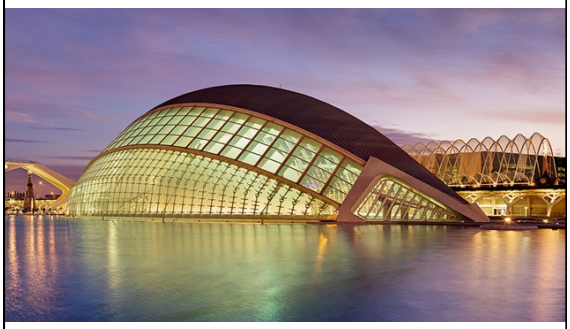
Today

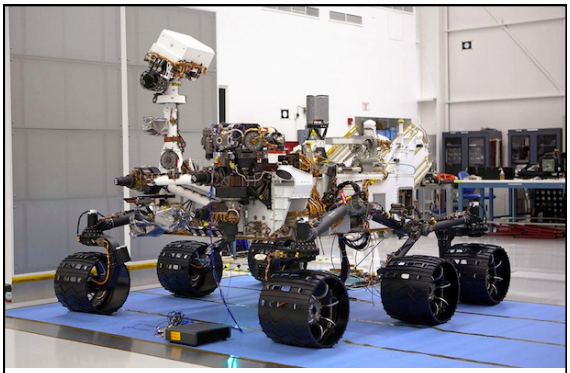
- What this course is about
- Logistics
- Course administration

Super TA Staff

- Head TA: Meg Bednarcik
- Jon Hegarty
- Danny Schlitt
- Milo Watanabe

What CS101 is About





350 Million Mile Journey, Lands 1.5M from Target

```
# This python 2.7 program simulates Archimedes method for approximating pi.
#
import math
import Tkinter
from stddraw import *

DEBUG = True

# N is the number of sides of a regular polygon.
#
def area(angle, side, N):
    width = side * math.cos(angle)
    height = side * math.sin(angle)
    return N * abs(width * height)

def background(picture, x0, y0, side):
    picture.clear()
    picture.circle(x0, y0, side)
    picture.line(.5, 0, .5, 1.0, color='red')
    picture.line(0, .5, 1.0, .5, color='red')
```

```
def points(x, y, N, inSide, angle):
    def point(side, i):
        totalAngle = angle * (2 * i - 1)
        return (x + side * math.cos(totalAngle), y + side *
)

    outSide = inSide / math.cos(angle)
    inner = [ point(inSide,  i) for i in range(1, N + 1)]
    outer = [ point(outSide, i) for i in range(1, N + 1)]

    return (inner, outer)

def render(picture, x0, y0, N, side, under, over):
    def renderSegments(inner, outer, color):
        if len(inner) >= 2:
            p1 = outer[0]
            p2 = outer[1]
```


What this course is about

- An introduction to Computer Science
- First step is learning how to “code”
  - Drafting, Model Building
  - A way to express your ideas
- Have an idea? You can make it happen!

What this course is about

- We will use ~~Java~~ Python as our programming language
- We will have 10 programming projects
- By the end of the semester
  - You’ll be a competent python programmer
  - You’ll have a better understanding of computer science

What this course is about

- Central idea: learning to develop and express algorithms
- How to build stuff!
- CS M/O:
  - **Learn by doing!**
  - The more you do the better you get at it

Required Background

- High School Algebra
- Familiarity with Trigonometry and Geometry also helpful.
- No programming experience required.
- A taste for building things also helpful.

Our Launch Point: High School Algebra

We saw a lot of quadratic expressions:

$$ax^2 + bx + c$$

Where a, b and c are **constants** and x is a **variable**. We learned to solve for roots, how to factor them, we learned the properties of their curves, etc.

Algebraic Expressions

Our quadratic expressions:

$$ax^2 + bx + c$$

were sometimes written as **functions**:

$$f(x) = ax^2 + bx + c$$

Algebraic Expressions and Functions

The latter:

$$f(x) = ax^2 + bx + c$$

is a major upgrade with two important payoffs:

1. Makes variable "x" explicit,
2. Associates a name (i.e., "f") with the function.

Functions and Code

- Roughly speaking, a piece of computer software is a collection of functions.
- In HS algebra our functions usually worked with real numbers.
- In programming, there are lots and lots of interesting **types** of inputs for our functions.

CS101 and CS102

- A principal theme of CS101 is mastering the art of expressing algorithms as functions, **procedural abstraction.**
- A principal theme of CS102 is mastering the art of writing new types, (values and functions), **data abstraction**.

Function Definitions and Uses

- A **definition** of f:

$$f(x) = ax^2 + bx + c$$

- Two **uses**, **calls** or **applications** of function f:

f(5) f(2 + 2)

Function Definitions and Uses

- A **definition** of f:

x is a **variable** because it stands for something that will **vary** from one use of f to the next.

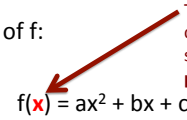
$$f(x) = ax^2 + bx + c$$

- Two **uses**, **calls** or **applications** of function f:

f(5) f(2 + 2)

Function Definitions and Uses

- A **definition** of f:



This occurrence of x is called a **parameter** or sometimes **formal parameter** of f.

$f(x) = ax^2 + bx + c$

- Two **uses**, **calls** or **applications** of function f:

$f(5)$ $f(2 + 2)$

5 and **2 + 2** are **arguments** provided as inputs to function f.

Computing the Value of a Call

- Lets say constants a = 1, b = 2 and c = 3:

$f(x) = x^2 + 2x + 3$

- Then to compute the **value** of a call f(5), we plug the argument 5 in for parameter x and **reduce**:

$f(5) = 5^2 + 2*5 + 3$
$= 25 + 10 + 3$
$= 35 + 3$
$= 38$

Computing the Value of a Call

- Lets say constants a = 1, b = 2 and c = 3:

$f(x) = x^2 + 2x + 3$

- Then to compute the **value** of a call f(5), we plug the argument 5 in for parameter x and **reduce**:

$f(5) = 5^2 + 2*5 + 3$
$= 25 + 10 + 3$
$= 35 + 3$
$= 38$

In computer science we are very much interested in the number of steps required to get our answer!

Computer Science and Math

- Like mathematics, computer science emphasizes problem solving
- In many ways programming is like applied or "active" algebra:

$$f(x) = ax^2 + bx + c$$

- In Python:

```
def f(x): return a * x ** 2 + b * x + c
```


Euclid's GCD Algorithm, 300BCE

$$\gcd(m, n) = \begin{cases} m & \text{if n is 0,} \\ \gcd(n, m \,\%\, n) & \text{otherwise} \end{cases}$$

Euclid's GCD Algorithm, 300BCE

$$\gcd(m, n) = \begin{cases} m & \text{if n is 0,} \\ \gcd(n, m \,\%\, n) & \text{otherwise} \end{cases}$$

```
gcd(25, 10) = gcd(10, 25 % 10)
            = gcd(10, 5)
            = gcd(5, 10 % 5)
            = gcd(5, 0)
            = 5
```

Euclid's GCD Algorithm, 300BCE

$$\text{gcd}(m, n) = \begin{cases} m & \text{if n is 0,} \\ \text{gcd}(n, m \% n) & \text{otherwise} \end{cases}$$

```
def gcd(m, n):
    if n == 0:
        return m
    else:
        return gcd(n, m % n)
```


IDLE Demo

How Programming Works

- Using an *editor* program, a programmer develops the TEXT of a program in some language, e.g., Python
- They then use another program, a *compiler*, to *translate* the text into the binary language of the machine.

Programming (Basic Model)

Binary Program is in the native language of the computer so the binary program can be executed.

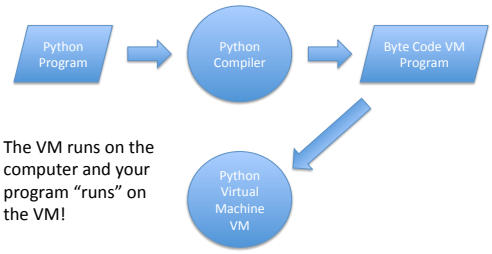
Programming (Basic Model)

Since each computer has it's own native language, a compiler that can produce binaries for one computer won't necessarily be able to produce binaries that will run on a different computer.

Programming (VM Model)

The Byte Code Program is in the native language of a "virtual" computer. The virtual machine (VM) is just a program that can be implemented on any computer, no matter it's binary language.

Programming (VM Model)



The VM runs on the computer and your program "runs" on the VM!

Course Admin

Course Admin

- Two 75-minute lectures each week;

  No laptops/screens in lecture.

- One one-hour lab;

  Laptops required in lab.

NB: LABS MEET TODAY AND TOMORROW.

Rough Outline - September

1. Basics, Expressions, Functions and Libraries
2. Booleans, Branching, Repetition, Tessellation
3. Repetition
4. Structured Data, Tuples and Lists, Function Values

Rough Outline - October

5. Comparing and Searching
6. Sorting
7. Storage, Storage Diagrams
8. Machines
9. Strings

Rough Outline - November

10. Digital Audio
11. Digital Images
12. Mutation
13. Mutable Dictionaries
14. Review and Preview

Tour of course website

Workload

- Most of our material is covered in lecture, background reading in HM CS5 Book CS for All, Guttag or Downey.
- 10 Programming projects
- Two midterm exams and a final (roughly every 5 weeks)

Grading

- 45% for 10 problem sets, plenty of opportunity for extra credit
- 40% for 3 exams
- 15% for consistent course participation
  - Lab, lecture, Piazza forum

How to Succeed in CS 101

- Start problem sets *right away*!
- Pay careful attention to detail.
- Seek help when you need it.
- Show up consistently, participate in class, ask questions.

Rules of the Road

- Late homework penalty 25% each day, penalty excused for documented medical problems or family emergencies only;
- Honor code strictly enforced.

Where did computing come from?

Blombos Stone, South Africa 77,000 Years Ago

From Counting to Arithmetic

Sumeria

5,000 Years Ago

MS 2462
Lexical list of birds, animals and objects, preceded by numbers, "The Tribute".
Sumer, ca. 2500 BC

YBC 7289 – 1800BCE

From Arithmetic to Algebra

Brahmagupta (640AD)

Foundational writings on algebra and astronomy. His text on astronomy, *The Brahmasphutasiddhanta* is the earliest known text to treat zero as a numeral in its own right. A copy of this text was taken to Baghdad ~800AD. Its translation transformed the Arab world.

Hindu Numeral System

- Developed in fits and starts over 5,000 years
- Key Innovations:
  - Numerals as strings of decimal digits {0, 1, …, 9}
  - Meaning (value) of an occurrence of a digit depends on its right-to-left position in the string
  - Zero (!) – first known occurrence in India ~600AD

Muhammad ibn Mūsā al-Khwārizmī

~830AD A mathematician and scholar in the House of Wisdom in Baghdad. After the Islamic conquest of Persia, Baghdad became the center of scientific studies and trade.

- *The Compendious Book on Calculation by Completion and Balancing* (al jabr)
- *On the Calculation with Hindu Numerals.*

Fibonacci (1170 – 1250)

Book of Calculation

Introduction in Europe revolutionized western culture.

Leibniz (1646-1714)

- Infinitesimal Calculus
- Binary Numeral System

  101_{10}

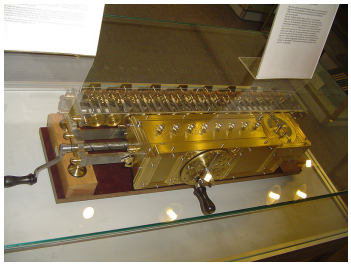
  101_{2}
- Function

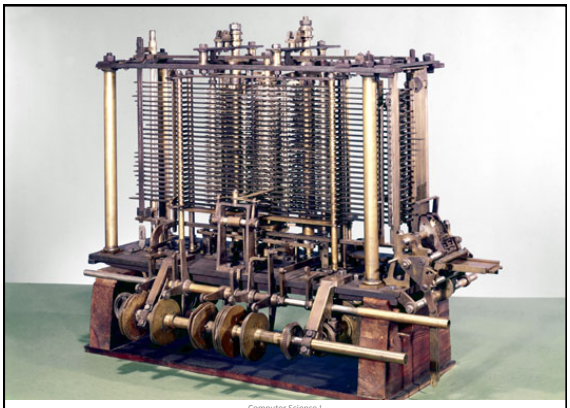
... it is beneath the dignity of excellent men to waste their time in calculation when any peasant could do the work just as accurately with the aid of a machine.

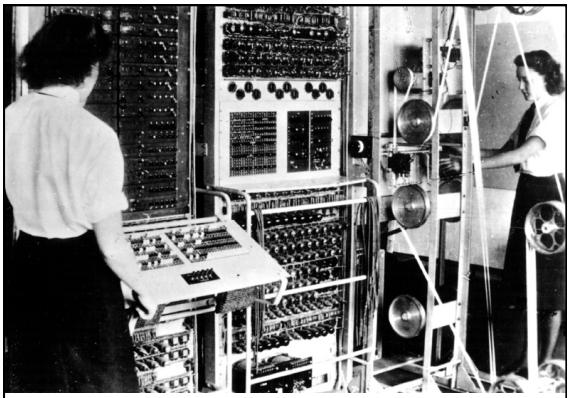
—-- Gottfried Leibniz

Leibniz' *Stepped Reckoner* (1672)

Addition, Subtraction, Multiplication, Division








April 1, 1976