CS 100: Gates and Drawing and Turtles
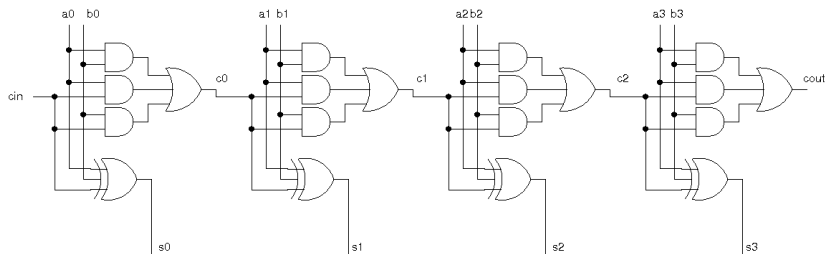
Chris Kauffman

Week 3-1

Logistics

Homeworks

- HW 2 due Friday at 11:59pm
  - Code.org plus a few additional exercises
- HW 3 Python programming
  - Make sure you have access to a computer
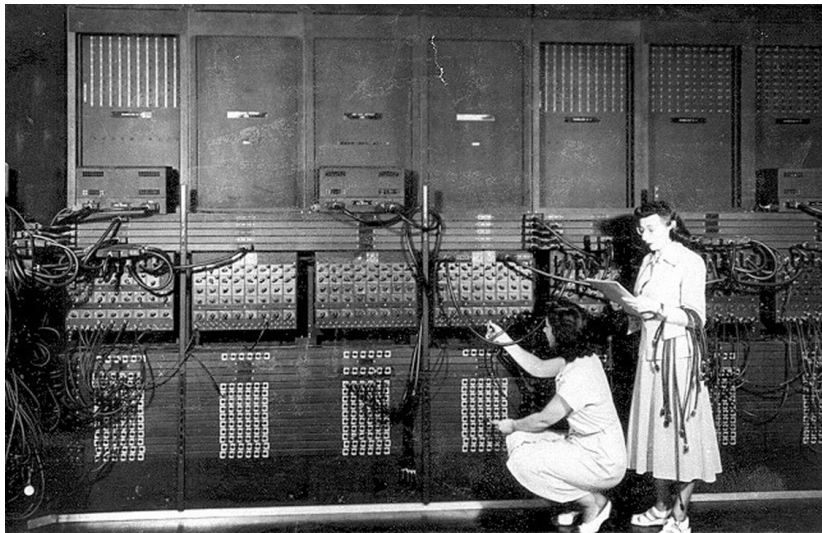 - Install Python 3
  - Will be posted over the weekend

Mini-Exam

- Thursday
- Last 30 minutes of class
- 1 page, front and back
- Open notes, book, slides
- Stuff like HW 1 and code.org exercises

Goals Today

- Gates and Programming
- Drawing with python
- Basic programming elements in python

Gates That "Do" Stuff

4-Bit Adder

- ▶ Two input number a, b (top)
- ▶ Each input has 4 bits
- ▶ Output s which is the sum of them (bottom)
- ▶ Also a carry bit *cout* (right)

On a Breadboard

Source

Programming the ENIAC = Rewiring

Source: Huffington Post

The Machine Today

Repeat

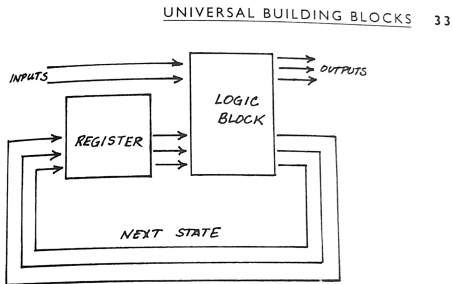- Input goes in
- Logic changes registers
- Output goes out

FIGURE 14

Finite-state machine, with logic block feeding register

Codes for Instructions and Data

Primitive Operations

Sequence of bits for each thing device can do

ADD 0001

SUBTRACT 0010

MULTIPLY 0011

Registers

Each place bits can be STORED is assigned a sequence of bits

REGISTER 0 0000

REGISTER 1 0001

REGISTER 2 0010

Machine Instructions

Usually DOING something involves an action and a few locations

```
ADD REGISTER 0 to REGISTER 1, put answer in REGISTER 2

Bits: 0001 0000 0001 0010
      ADD  REG0 REG1 REG2

MULTIPLY REG2 by REG4, put answer in REG3
Bits: 0011 0010 0100 0011
      MULT REG2 REG4 REG3
```
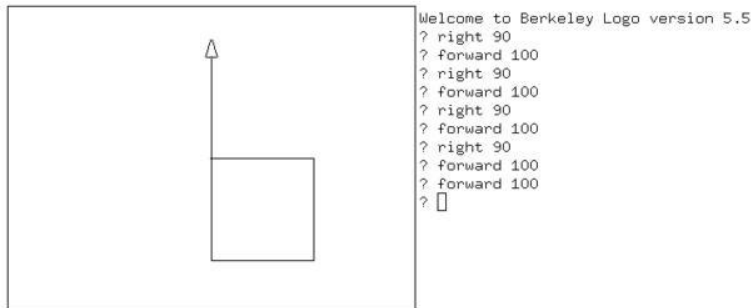
Layers on Layers

Anyone want to program in binary? Me neither luckily. . .
This was an early problem that got solved

Turtles Then

My first experience programming: drawing with a "turtle"

- ▶ Didn't tell me it was programming
- ▶ Language called *Logo*, still used in some settings today
- ▶ Similar to the "Artist" exercises on code.org


```
Welcome to Berkeley Logo version 5.5
? right 90
? forward 100
? right 90
? forward 100
? right 90
? forward 100
? right 90
? forward 100
? forward 100
? []
```

Source

Turtles Today

- We will do some programming in Python in this class
- Python comes with the turtle built in
- Used extensively to demonstrate in How to think like a Computer Scientist starting in Chapter 3
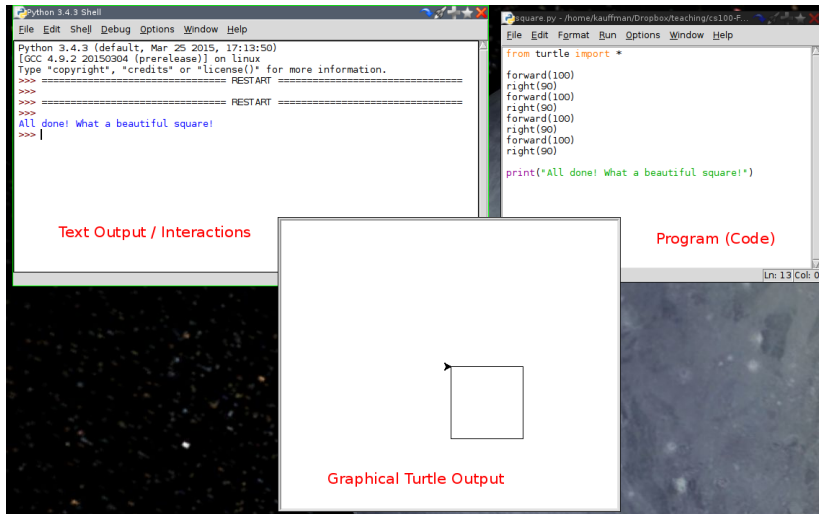- Demonstrations on screen are in order

Items TODO

Install Python on your personal system

1. Go to https://www.python.org/
2. Click "Downloads"
3. Click "Download Python 3.5.0"
   - Your Platform (Windows/Mac/Unix) should show up
4. Save the file in a sensible spot (Downloads folder)
5. Install
   - Windows: double click and run installer
   - Mac(?): double click to mount disk image and open, then double click "Python.mkpg" to run installer
6. Look for "IDLE (Python GUI)" program
7. Run it to start a python loop

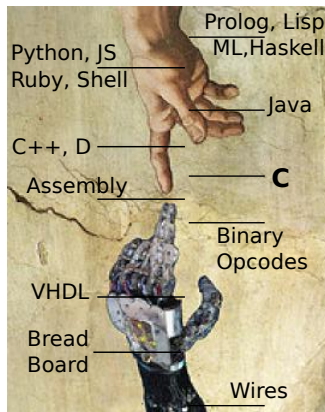If you get stuck with install, see me or a TA in office hours

IDLE - A program to write Python Programs

Text Output / Interactions

Program (Code)

Graphical Turtle Output

Overview of Python

Pure Abstraction

- A programming language and environment
- A **higher-level** way of interacting with a computer
- Used by lots of companies to real work (e.g. Google)

Bare Metal

Programming

1. Write down some instructions
2. Ask the computer to execute those instructions
3. Look at the result
4. If happy with result, declare victory
5. Otherwise, change the instructions and go to 2

Tiny Little Turtle Commands

```python
from turtle import *          # Use the turtle module

forward(50)                   # move forward
backward(100)                 # move backward
right(90)                     # turn left
left(120)                     # turn right

circle(30)                    # draw a circle
stamp()                       # stamp the turtle symbol
shape("turtle")               # Change shape of pen
                              # to a turtle
# arrow, turtle, circle, square, triangle, classic

square(100)                   # ERROR: need to teach this
```

Differences from "Think"

```python
import turtle              # Allows us to use turtles
wn = turtle.Screen()       # Creates a playground for turtles
alex = turtle.Turtle()     # Create a turtle, assign to alex

alex.forward(50)           # Tell alex to move forward by 50 units
alex.left(90)              # Tell alex to turn by 90 degrees
alex.forward(30)           # Complete the second side of a rectangle

wn.mainloop()              # Wait for user to close window
```

- Can have many turtles around: alex, beth, clarence, debbie
- Can tell them each to do things individually with alex.forward(100) and debbie.right(30)
- We will mostly just tell the global turtle do stuff with forward(100)

Python: Repetition and Conditions

```python
print("Hello")                      # Print hello

for i in range(4):                  # Repeat, i=0,1,2,3
    print("Number: "+str(i))        # Print the i vals

myVar = 7                           # Assign a variable
if(myVar > 5):                      # Check something
    print("Bigger than five");
else:
    print("Smaller than five");

if(myVar % 2 == 0):                 # % is remainder op
    print("Even")
else:
    print("Odd")
```

Python: Functions

```python
def reportOddness(x):          # Define a function
    if(x % 2 == 0):
        print("Even")
    else:
        print("Odd")

reportOddness(2)               # Use the function
reportOddness(9)

def repeatedGreeting(n):       # Repeats in functions
    print("Saying hello "+str(n)+" times")
    for i in range(n):
        print("Hello!")

repeatedGreeting(3)
someValue = 5
repeatedGreeting(someValue)
```

Next Time

- Finish Code.org exercises and HW 2
- Install Python
- Finish "Pattern" Ch 3
- Read "Think"