Programming

ME 2984

“ If debugging is the process of removing software bugs, then programming must be the process of putting them in. ” -Edsger Dijkstra

GOALS FOR TODAY

- Introduce some basic concepts of programming
- Introduce Python
    - Experiment in the interpreter
- Fast track to hack
- Ask questions

Virginia Tech
Invent the Future

PROGRAMMING IN A MINUTE

- Writing a sequence of imperative statements to accomplish some desired task
  - Commanding a very studious idiot
- Constructing things which represent higher level concepts
- Languages have common types of statements
  - Turing Complete - all languages can do the same thing
- Computer reads file(s), executing statements in order

SIMPLE STATEMENTS

- Commanding computer to do specific things:
  - print “Hello, class!”
  - 2 + 2
  - 4 / 2
  - 2 ** 8
- Store result of commands in variables:
  - statement = “Hello, class!”
  - four = 2 + 2

SIMPLE THINGS, AREN’T

- Python uses a concept of types to reason about things
    - Similar to types of numbers in math (e.g. integers, reals, complex numbers, etc.)
    - Types help dictate how commands should work
        - 1 + 1 - both integers, result must be integer
        - 1 + 1.5 - integer and real, result must be real
        - “Hello” + “ class” - ???
- Python is “duck typed”

NUMBERS

- Integers are restricted to the integers
  - Precise, fast, and limited
- Floats approximate the reals
  - Memory is finite, the reals aren’t
  - What does this mean?
- Python tries to change type to best represent your output

STRINGS

- Strings are sequences of characters which often represent text
- Some arithmetic operations make sense (or can)
  - “Hello” + “, class!”
  - “Hi” * 3
- Others don’t
  - “Cat” / “Apple” - ???
- Operations which aren’t arithmetic, but make sense
  - len(“Four”)

FUNCTIONS

- Blocks of commands to execute
  - Similar to math functions (f(x, y, …) => a)
- More than just data
  - Functions have signatures
  - Object can have side effects
  - Can return some value from the computation

BOOLEANS

- Values which can take only two states - true or false
  - Core of having programs make decisions
  - Computers are binary
- Boolean Algebra - logic as math
  - True is 1, False is 0
  - Logical operands as arithmetic ones

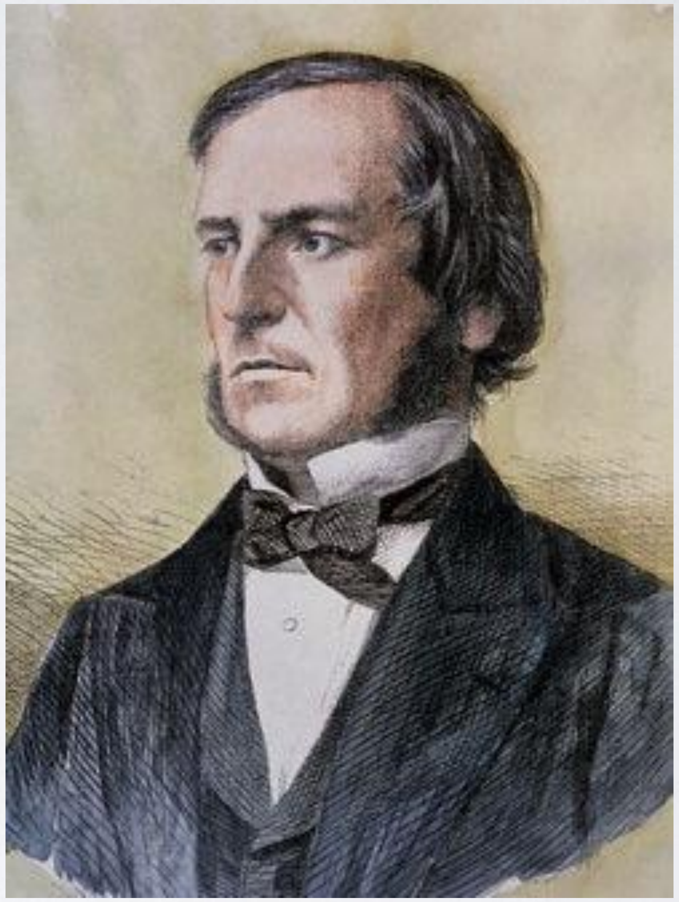

Image Credit: Wikipedia

NOT THAT HARD

- NOT (¬ in math, ! in Python) produces the opposite state
  - !True?
  - !False?
  - !!True?
- NOT is the same as f(x) = 1 - x

AND ALMOST EASY

- AND ($\wedge$, &) is $f(x, y) = x * y$
  - True & True?
  - True & False?
  - False & False
  - True & True & False?

OR JUST BE EASYGOING

- OR (v, |) is more complicated - f(x, y) = x + y - (x * y)
  - True | True?
  - True | False?
  - False | False?

EXCLUSIVELY WEIRD

- XOR (⊕, ^ in Python) is trickier
  - X ^ Y = (X | Y) & !(X & Y)
- Intuitively, operation is “one or the other, and not both”
- True ^ False?
- False ^ False?
- True ^ True?

LOGICAL IDENTITIES

- Associative properties
  - X | (Y | Z) = (X | Y) | Z
  - X & (Y & Z) = (X & Y) & Z
- Commutative properties
  - X | Y = Y | X
  - X & Y = Y & X
- Distributive
  - X & (Y | Z) = (X & Y) | (X & Z)
  - X | (Y & Z) = (X | Y) & (X | Z)

NEW LOGICAL PROPERTIES

- Idempotent - having the same power
  - X | X?
  - X & X?
- Simplification/Short circuiting
  - True & X?
  - False & X?
  - True | X?
  - False | X?

CREATING BOOLEAN VALUES

- Many operations can take arguments and produce a boolean value
- Comparison operators
  - Greater than (>), less than (<), Greater-than-or-equal (>=), less-than-or-equal (<=)
  - Equal to (==), Not equal to (~=)
- Functions can return boolean values

CONTROL FLOW

- Boolean logic is the basis for controlling program execution
  - Choosing what to do, not just doing it
- Three basic control types
  - If/Else
  - For
  - While

FORK IN THE ROAD

- If/Else tells the program how to choose between different sequences
    - “If A, do this. Otherwise, do that.”
- Allows selective operation
- Often provides an “else-if” command to provide more flexibility.

CRANKING A WIDGET

- For loops provide a way to execute a repetitive set of commands
- “For every slide in this lecture, take notes”
  - Can also work on some set number of cycles if you mangle the English
  - “For every number from 1 to 2, read this”
- Break out of a loop using “break”
- Skip one iteration of the loop using “continue”

WHILING AWAY TIME

- Repeat some actions until told to stop
- Similar to for loops, but more open ended
- Don't become the Sorcerer's Apprentice

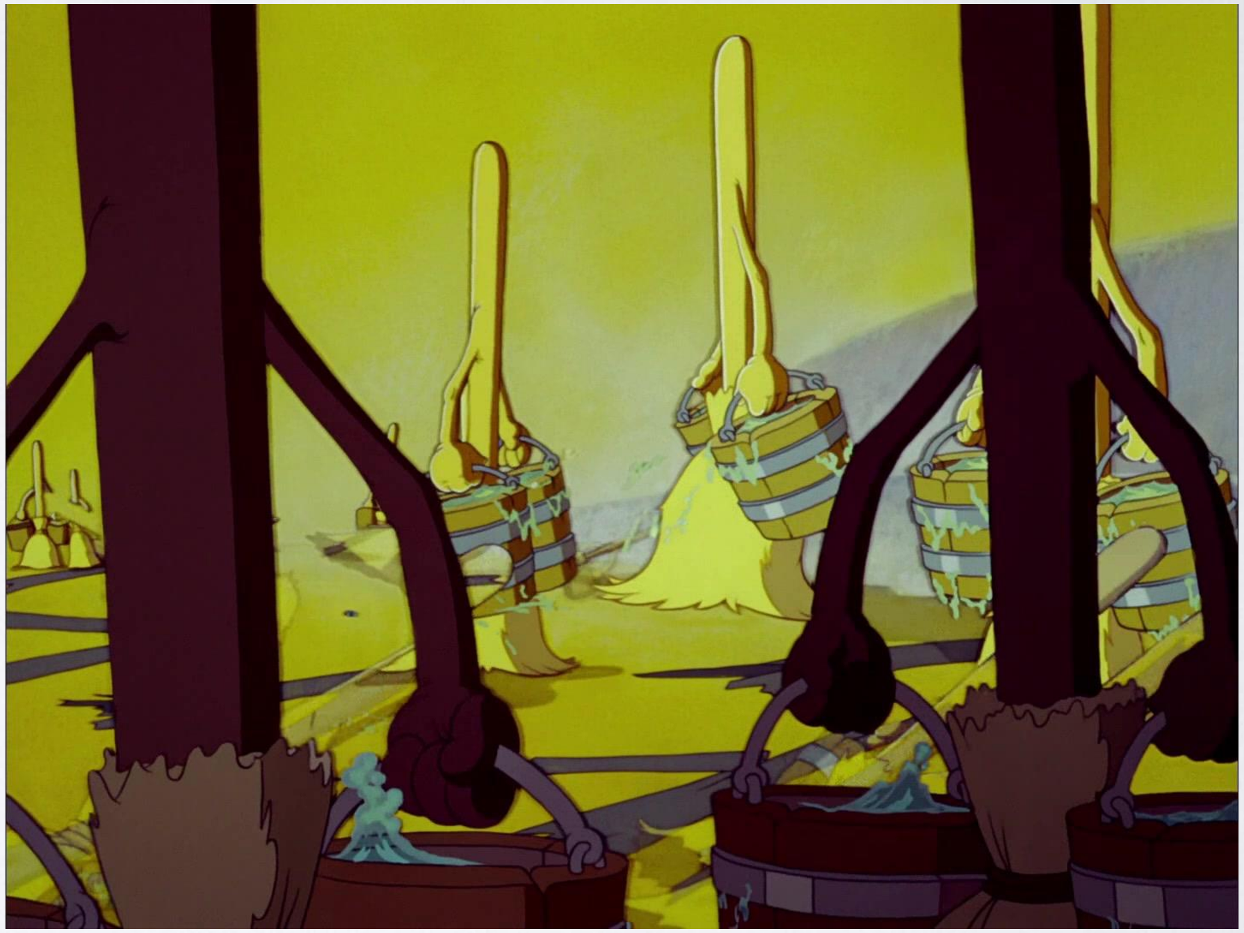

Image Credit: Disney

LET’S PUT IT ALL TOGETHER

- Demo of many of these ideas being used
- Chance to play around and understand code
- A quick word about whitespace

EXPLORING PYTHON

- Programming is a big place
- Python is easy on the scale, but still big
- Luckily, Python is interpreted
  - Explore by using the interpreter!
  - Python provides a tutorial (http://docs.python.org/2.7/tutorial/)

ASSIGNMENT 2 IS OUT

- Written and coding section
- Due 1 week from today
- Coding shouldn’t be too hard, but isn’t trivial
    - Start early
    - Share ideas, not code