Day #1
Getting Comfortable with Core Concepts

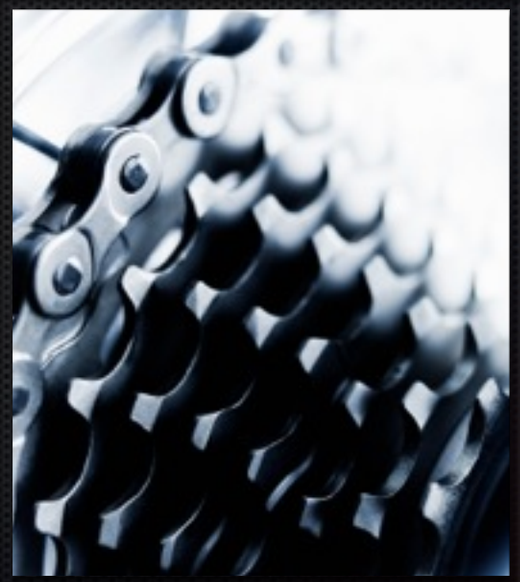
John Simpson

DHSI-2014

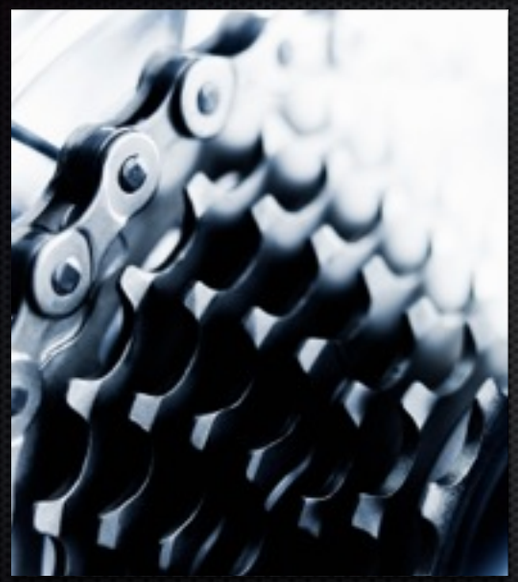
Fundamentals of Programming/Coding for Human(s|ists)

What we're doing today

1. Set down core principles of practice
2. Learn basic programming concepts
3. Practice using those concepts



What we're doing this week



Tuesday: Unix & Python

Wednesday: GitHub, RegEx, Projects Introduced & Started

Thursday: Project Work

Friday: Finalizing Projects & What Next

Core Principles of Practice

These are the core guidelines with which we will conduct the course and which you *will* use in *all* future programming activities. There are three:

1. You can do this.
2. Know computers.
3. Plan, Do, Review.

I have a copy of CL/DM as PDF (and an original)

Images From:

http://en.wikipedia.org/wiki/Ted_Nelson

http://www.electronicbookreview.com/thread/electropoetics/distributed

> “Any nitwit can understand computers, and many do. Unfortunately, due to ridiculous historical circumstances, computers have been a mystery to most of the world. And this situation does not seem to be improving…”
>
> *—Theodor H. Nelson, Computer Lib, 1974*

Two points to extract from this:

1. If nitwits can understand computers then you certainly can.
2. The fact that you do not currently understand computers *as a programmer* is not (entirely) your fault.

Computers can’t do everything but they can do many things.

Church-Turing Thesis

NP-hard vs. complete. Travelling Salesman

Things that go outside of memory limitations and start writing to disk

Plan	What will the program do? How will the program do this?
Do	How will the program do this *exactly*?
Review	What have you learned?

This “wash, rinse, repeat” model will be repeated over and over on various scales.

“Just do it” at your peril.

Form, Function, and Feedback

Plan

A program is just a set of instructions
that a computer can perform

“First, solve the problem. Then, write the code.”

–John Johnson

“I have always found that plans are useless, but planning is indispensable.”

–Dwight D. Eisenhower

Instructions Practice

Create a list of instruction for the provided scenario.
Each list must have the following features:

- Explicit detail. Be thorough and exact.
- At least one choice to be made.
- An action that is repeated until something happens

Scenarios. Bake a cake. Ride a bicycle. Change a tire. Scoop an item on eBay. Create a secure password that you can remember. Perform a literature review.

Students get split into teams and each given an different scenario. Give them 5-10 minutes to come up with a process. Share with class.

As we go through the components of a program these templates can be used as meaningful examples.

Flowchart or List? When we write programs we need to think with flowcharts but write them like they were lists.

Natural

Formal

Two language types

We live in the world of vagueness and imprecision, the computer will have little of this.

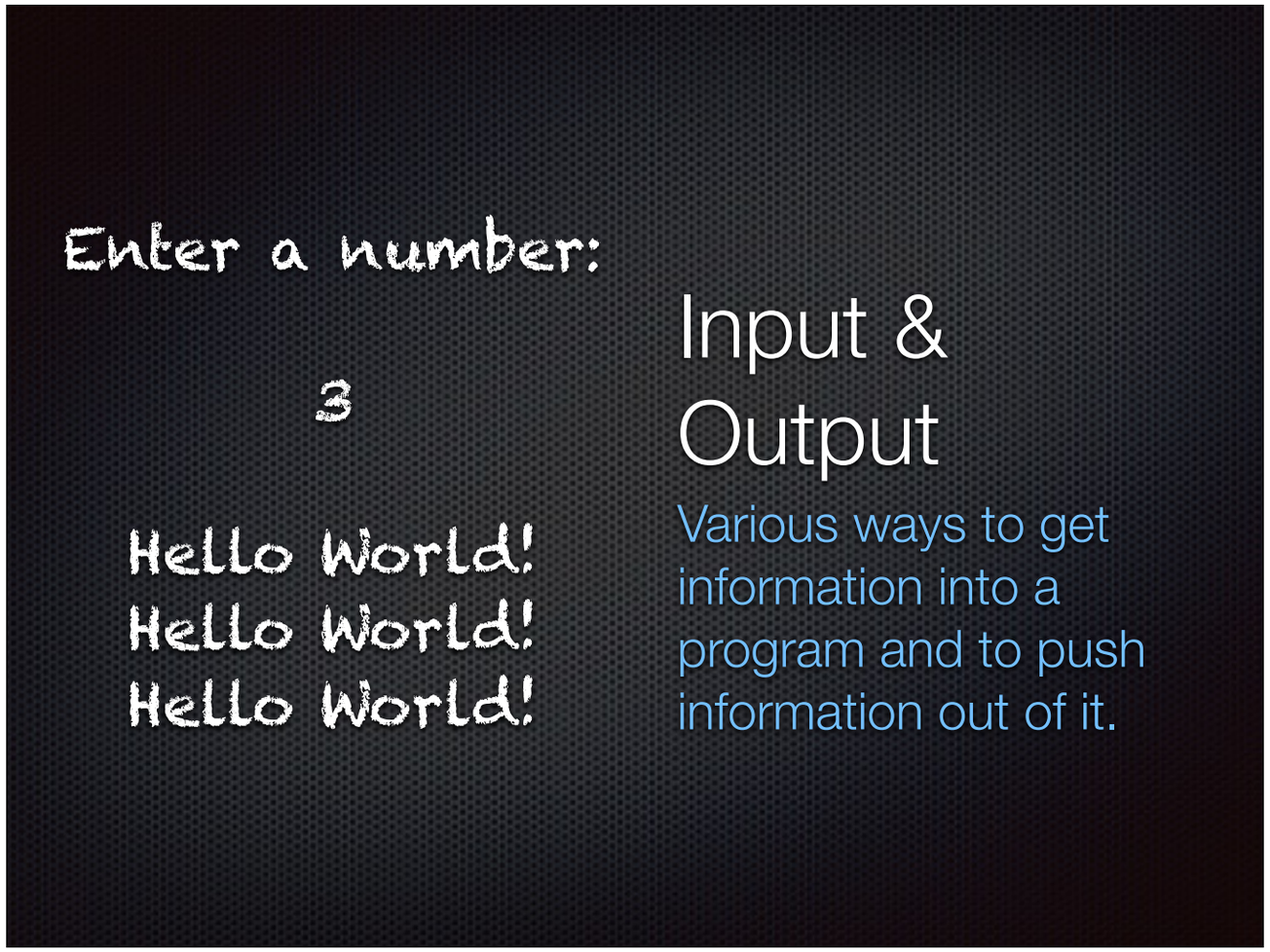
Three Error Types

If things go wrong it will be in (at least) one of these ways.

Syntactic

Runtime

Semantic



Input & Output

Various ways to get information into a program and to push information out of it.

Conditionals

Statements of the form:

if something is the case
then do something

Loops

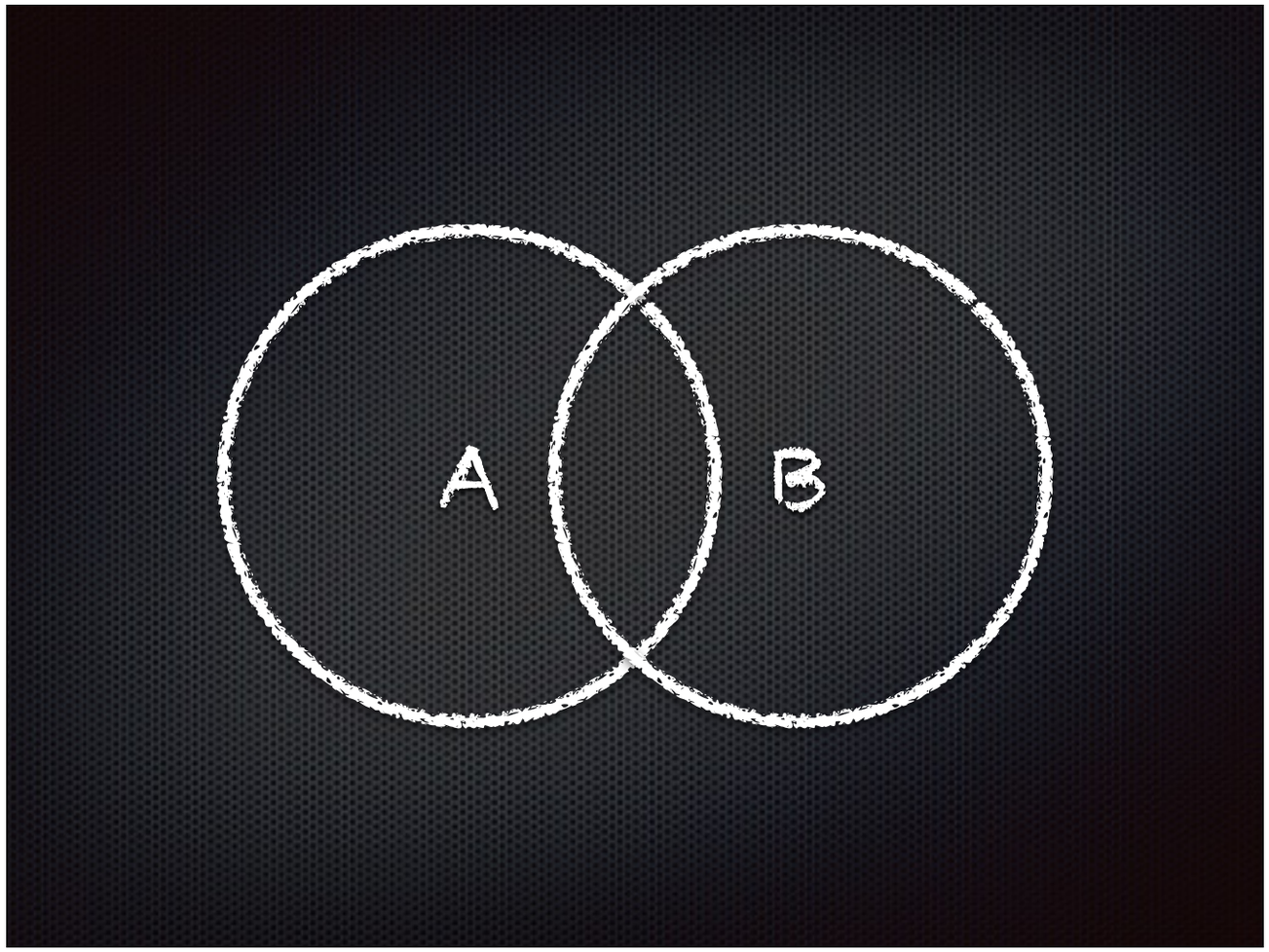
Statements of the form:

repeat doing something
until something happens

Operators

Connective components of statements:

AND, OR, NOT, EQUAL TO, GREATER THAN, Etc.



Venn Diagram. Have students draw circles and show the following statements by colouring the correct area.

A and B

A or B

not A and B

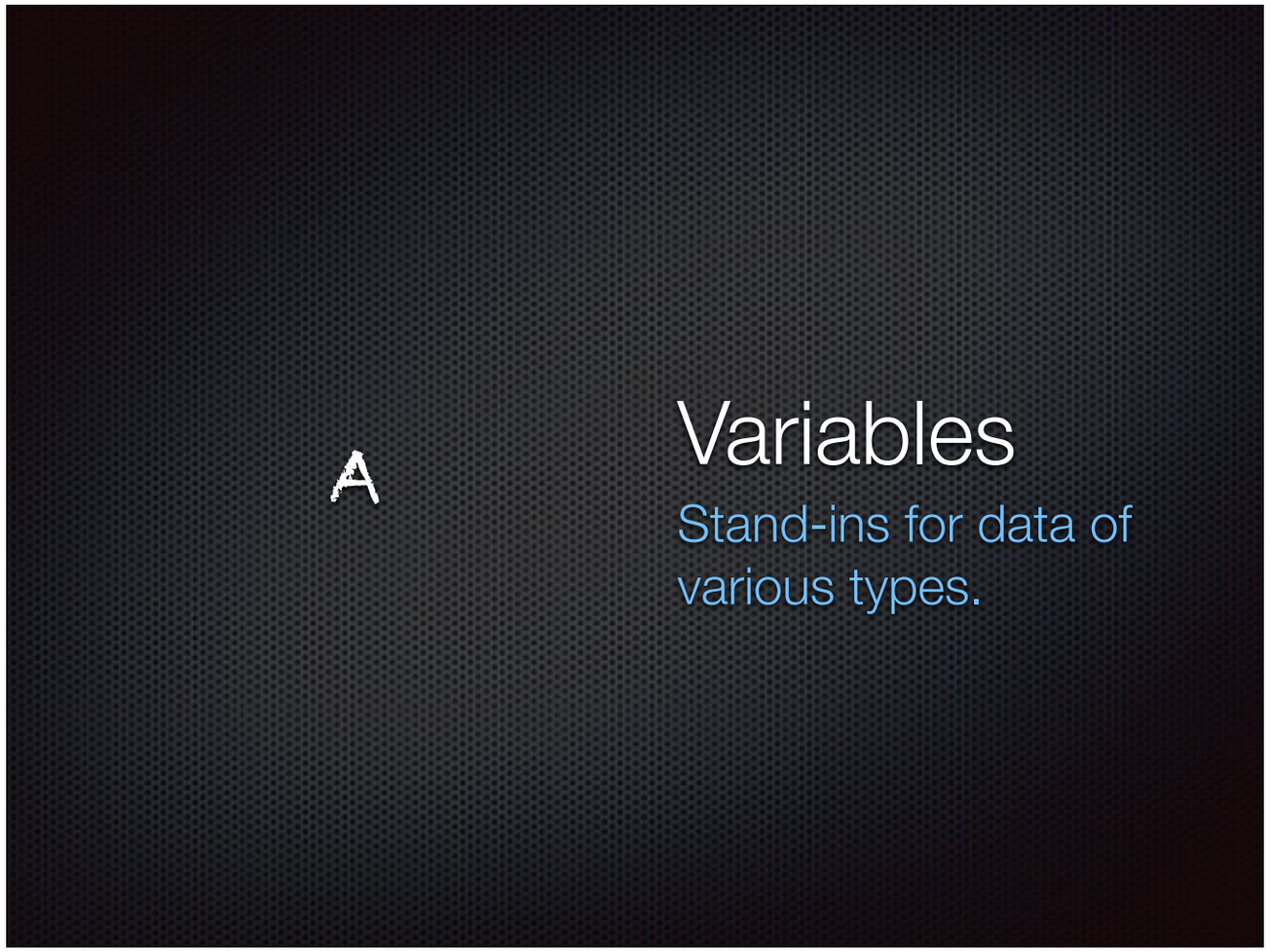
not A or B

not A and not B

not A or not B

not (A and B)

not (A or B)


A
Variables
Stand-ins for data of various types.

Collections

Various groupings of variables and objects.

A

A

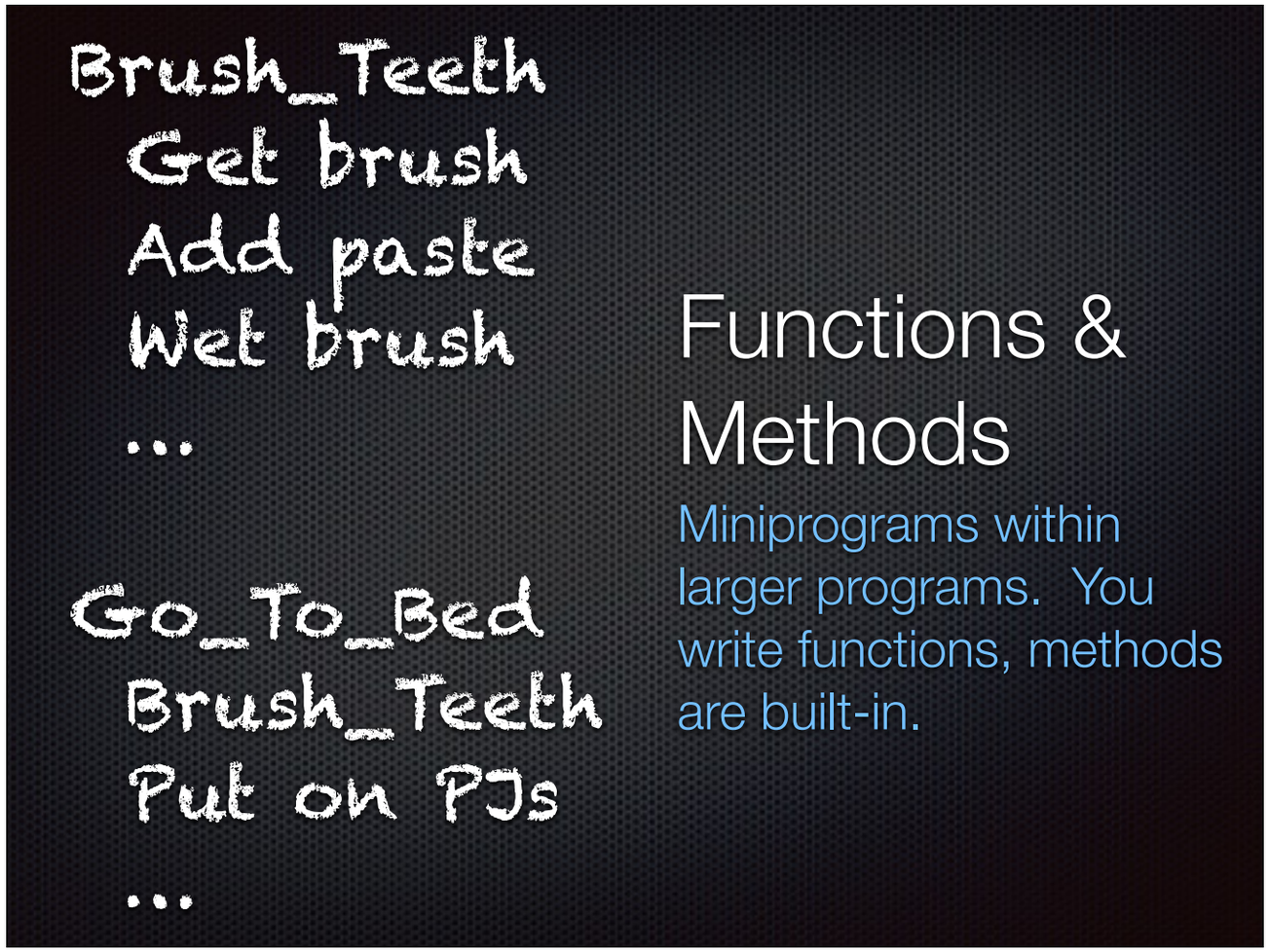
B

a

z

123

a pony



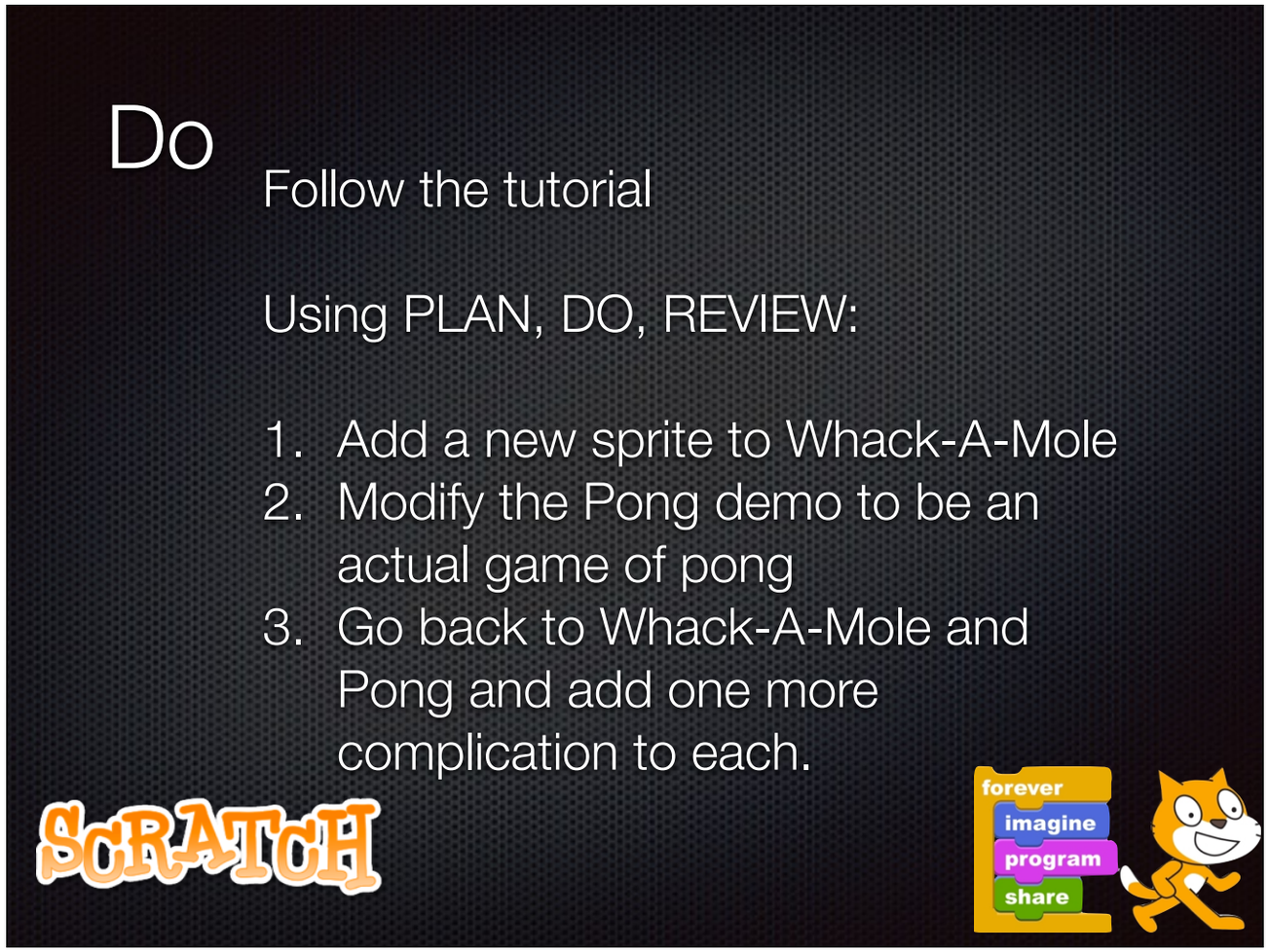
Functions & Methods

Miniprograms within larger programs. You write functions, methods are built-in.

The method function distinction is wrong (you can write methods too) but works for now.

Scratch Challenge. Follow the tutorial and then use PLAN, DO, REVIEW on the following:

1. Add another sprite to click on in the Whack-A-Mole variant
2. Modify the pong demo to actually be a game of pong, including scoring.
3. Design and build an interactive program from scratch.



Scratch Challenge. Follow the tutorial and then use PLAN, DO, REVIEW on the following:

1. Add another sprite to click on in the Whack-A-Mole variant
2. Modify the pong demo to actually be a game of pong, including scoring.
3. Design and build an interactive program from scratch.