Day 3

Input s
Agenda

1. Recap: Conditionals
2. Debugging
3. Input: Keyboard
4. Input: Mouse
Recap: Conditionals

If I am hungry, then I will eat food. Otherwise I will not eat.

// code
Recap: Conditionals

If I am hungry, then I will eat food. Otherwise I will not eat.

```c
if (hungry) {
    EAT FOOD;
}
else {
    DO NOT EAT;
}
```
Recap: Conditionals

If I am hungry and thirsty, then I will eat food. Otherwise I will not eat.

```plaintext
if (hungry) {
    EAT FOOD;
}
else {
    DO NOT EAT;
}
```
Recap: Conditionals

If I am hungry and thirsty, then I will eat food. Otherwise I will not eat.

```c
if (hungry && thirsty) {
    EAT FOOD;
}
else {
    DO NOT EAT;
}
```
Recap: Conditionals

If I am hungry or thirsty, then I will eat food. Otherwise I will not eat.

```java
if (hungry && thirsty) {
    EAT FOOD;
}
else {
    DO NOT EAT;
}
```
Recap: Conditionals

If I am hungry or thirsty, then I will eat food. Otherwise I will not eat.

```c
if (hungry || thirsty) {
  EAT FOOD;
}
else {
  DO NOT EAT;
}
```
Recap: Conditionals

If I am hungry and the temperature is 70° F, then I will eat food. Otherwise I will not eat.

```c
if (hungry || thirsty) {
    EAT FOOD;
}
else {
    DO NOT EAT;
}
```
Recap: Conditionals

If I am hungry and the temperature is 70° F, then I will eat food. Otherwise I will not eat.

```c
int temp = 71;
if (hungry && temp==70) {
    EAT FOOD;
}
else {
    DO NOT EAT;
}
```
Recap: Conditionals

If I am hungry and the temperature is 70° F or below, then I will eat food. Otherwise I will not eat.
Recap: Conditionals

If I am hungry and the temperature is 70° F or below, then I will eat food. Otherwise I will not eat.
when you go (boot) camping, be careful of bugs.
0800  Started
1000  Stopped
13:00 (032) MP - MC
(033) PRO2 2.1306766/5
Contd
Relays 6-2 in 033 failed speed speed test
in relay
Relays changed
1100  Started cosine tape (sine check)
1525  Started multi adder test
1545  Relay #70 panel F
moth in relay
First actual case of bug being found.
1630  Antenna started.
1700  Closed down.
Debugging

1. Bugs are errors in computer programs

2. Common problems: forgot a semicolon; or bracket } ); speling errors, ...

3. Bugs happen all the time and it’s important to get good at finding and fixing them. Otherwise coding can become frustrating quickly.
Debugging

println("Hello there!");

// OR

int a = 4;
println(a);
Exercise: Using println();

Create a simple sketch and see what messages you get when you add errors, (like adding typos or leaving out important characters)

Declare a variable that changes while the program is running and use `println()` to print its values to the console. Or use `println()` with a condition to test when it is true or false.
Debugging

When your code doesn’t work, don’t randomly guess!

- Read the error message in the console! It’s trying to help you.
  - Check the line number if there is one.
  - Check the message: it tells you what’s wrong
  - Can’t understand it? Google It! StackOverflow usually has the right answer

- Debug your program
  - Don’t try to fix everything at once. Separate the code into pieces (Line by line if necessary) and figure out where it breaks.
  - Use /* comments */ to de-activate code blocks
  - Use the console to check if variables contain the right content; or if conditions are executed at the right time.
Inputs: Keyboard

situation-dependent speech, prepared speech.
keyPressed

boolean: true or false

Use in an if statement in the draw()
keyPressed + key

key is the currently-pressed key

```cpp
if (keyPressed == true) {
    if (key == 's' || key == 'S') {
        // doSomething
    }
    else {
        // doNothing
    }
}
```
keyPressed()

runs when keyPressed == true

int value = 0;

void draw()
{
    fill(value);
    ellipse(25, 25, 50, 50);
}

void keyPressed()
{
    if (value == 0)
    {
        value = 255;
    }
    else
    {
        value = 0;
    }
}
keyPressed() + keyCode

keyCode is a special ID number for every (or most) key.

Checking if key == coded makes sure that there is actually a keyCode for the one that’s currently pressed.

color value = color(125);

void keyPressed() {
    if (key == CODED) {
        if (keyCode == UP) {
            value += 10;
        } else if (keyCode == DOWN) {
            value -= 10;
        }
    } else {
        value = 125;
    }
}
keyReleased()

runs when keyPressed switches from true to false
String break = "10 Minutes";
Inputs: Mouse
If `mouseButton == LEFT` ... // or CENTER or RIGHT

If `mousePressed == TRUE` ... // or CENTER or RIGHT
void draw() {
    if (mousePressed && (mouseButton == LEFT)) {
        fill(0);
    } else if (mousePressed && (mouseButton == RIGHT)) {
        fill(255);
    } else {
        fill(125);
    }
    ellipse(width/2, height/2, 25, 25);
}
mousePressed()

runs once when the mouse is pressed

```java
void draw()
{
    ellipse(width/2, height/2, 25, 25);
    fill(125);
}

void mousePressed()
{
    if (mouseButton == LEFT) {
        fill(0);
    }
    else if (mouseButton == RIGHT) {
        fill(255);
    }
}
```
Integer variables that contain the X and Y position of the mouse cursor

```plaintext
void draw() {
    background(255);
    fill(0);
    ellipse(mouseX, mouseY, 10, 10);
}
```
Homework (or start now)

Create an interactive sketch using inputs like `mousePressed`, `keyPressed`, `keyTyped`, ...

Use conditionals to add even more control. For example, combine `mousePressed` and `mouseX, mouseY` to make a drawing app