Day 1

Code Section
Day 1 Agenda

Introductions
- Instructor background
- Student introductions

Creative Code
- What is code?
- Problem decomposition
- Learning languages

BREAK

Processing
- Anatomy of a sketch
- Drawing with Processing
- Live code exercises
Why do we code?
como estas
你好
como allez vous
こんにちは
how are you
Code
What is code?
Translating Instructions For a Computer
What computers do:
Store information as 1s and 0s and perform math and logic operations on it
Intermediate Languages and Libraries
Expand the Computer’s Vocabulary
Processing can be used in physical computing, interactive media, and image generation.

More Processing projects
Variables
Types of Variables

In groups of two or three, define one of these. Write your definitions on the board.
Types of Variables

These are some, but not all, primitive types.

- **int** stores an integer (e.g., 1)
- **float** stores a number with a decimal point (e.g., 9.31)
- **String** stores text (“Bootcamp 2016”)
- **boolean** true/false
Using variables

Use println() to receive values back in the **console**.

```java
int myNumber;
myNumber = 10;
println(myNumber);

> 10

myNumber = myNumber + 1;
println(myNumber);

> 11

String thisSchool = "Parsons";
println(thisSchool);

> Parsons
```
Functions
Function example

This example describes a series of materials and actions to accomplish an overall goal: putting on a shoe.

You’ll need some specific materials: a shoe, and a foot.

```java
void putShoeOnFoot(Shoe myShoe, Foot myFoot) {
    pickUp(myShoe);
    liftFoot(myFoot);
    lowerFootIntoShoe(myFoot, myShoe);
    tieShoe(myShoe);
    releaseShoe(myShoe);
}
```
Function example

This example describes a series of inputs and actions to accomplish an overall goal: putting on a shoe.

You’ll need to execute some specific actions: picking up your foot, etc.
Problem Decomposition
Peanut Butter & Jelly Sandwich

2 slices of bread
Peanut Butter
Jelly

1) Spread peanut butter on one slice of bread
2) Spread jelly on the other slice of bread
3) Put the pieces of bread together
Recipe Breakdown

**Variables:**
- breadSlice1
- breadSlice2
- peanutButter
- jelly

**Functions:**
- spreadOnBread()
- putBreadTogether()
Pseudocode 1

Pseudocode is useful for breaking down and understanding a complex task: translate code to "language"

// spread peanut Butter on breadSlice1

// spread jelly on breadSlice2

// put bread slices together
Pseudocode 2

Pseudocode is useful for breaking down and understanding a complex task.
Pseudocode: Your Turn!

1. With a partner, write out the instructions for making a sandwich or another food.

1. Swap instructions with another team.

1. Think through the instructions literally and specifically: What happens?
Processing
Processing

Processing consists of a programming language built on Java and an IDE (Integrated Development Environment).

It is specialized for visual design, drawing, and arts applications. It is also designed for teaching.

Processing has lots of built-in functions and variables for drawing on a canvas.
Program Execution

In Processing, `setup()` runs **one time**.

After that, `draw()` **repeats endlessly**, until you stop the program.

```plaintext
def setup()
    # initial setup code here


def draw()
    # drawing code here
```
Program Anatomy

In `setup()`, you’ll put things like `canvas size` and `background color`.

In `draw()`, you’ll put actions that you want to happen repeatedly. We’ll add these in a minute.

```plaintext
setup()
    // canvas size
    size(800, 800);
    // background color
    background(0);
}

draw()
```
Program Anatomy

Add **comments** by putting two slashes at the beginning of a line.

The program doesn’t run these: they are for humans to read and understand.

Use comments often!

```cpp
void setup() {
    // canvas size
    size(800, 800);
    // background color
    background(0);
}

void draw() {
}
```
Drawing functions

Processing has built-in functions for drawing, graphics, and creative coding.

You can find information about what Processing can do and how to use functions in the documentation.

```
ellipse(x, y, width, height);
rect(x, y, width, height);
line(x1, y1, x2, y2);
```
Anatomy of functions

Just as in the previous examples, the external part of these functions describes what they do.

Semicolons indicate the end of the line. Don’t forget them!
Anatomy of functions

Inside the parenthesis, you specify parameters for the shapes you’re drawing.

Look in the documentation to learn about the parameters for a specific function.

```javascript
ellipse(x, y, width, height);
rect(x, y, width, height);
line(x1, y1, x2, y2);
```
The Processing Canvas

What parameters should you use?

It helps to know something about the drawing space in Processing.

Processing’s canvas uses $x, y$ coordinates starting from the top left corner.
The Processing Canvas

Processing represents images in **pixels**.

Each image in Processing is a grid, with numbers representing the color value at each coordinate.

When you specify measurements, like square width, you’re referencing pixels.
The Processing Canvas

Processing can represent color in a few different ways.

**RGB:** Represented with three values, 0-255, and a fourth for transparency.

- \((r: 255, g: 0, b: 0)\)
- \((r: 0, g: 0, b: 255)\)
- \((r: 0, g: 255, b: 0)\)
- \((r: 0, g: 255, b: 255)\)
- \((r: 255, g: 0, b: 255)\)
- \((r: 255, g: 255, b: 255)\)

**Grayscale:** One value, 0-255.
Bringing it Together

Let’s do some live coding! You can follow along by downloading and opening Day01_exercise01.pde from Drive.

We’ll add parameters to our shape functions to create some images.
Live Code

Next, we'll create named variables to store our parameter information.

```cpp
int x = 100;
int y = 100;

int circWidth = 70;
int circHeight = 70;

do something()
{
    size(800, 800);
    background(0);
}

do draw()
{
    ellipse(x, y, circWidth,
            circHeight);
}
```
Live Code

We’ll add some color to the circle by creating a color variable, and setting the color.

In Processing, you can set a **fill** color and an **stroke** (outline) color.

There is a built in **color function** that accepts RGB or other color information.
Exercise

Let’s get off screen!

Take a sheet of paper and draw out the code step by step.

The canvas size is already set for you.

Loop through `draw()` five times.

```cpp
int x = 5;
int y = 5;

int rectWidth = 2;
int rectHeight = 2;

color rectColor = color(0);

void setup(){
  size(20, 20);
  background(255);
}

void draw(){
  fill(rectColor);
  rect(x, y, rectWidth, rectHeight);
  y = y+1
}
Exercise

Let’s get off screen!

What does your sketch look like?

```cpp
int x = 5;
int y = 5;

int rectWidth = 2;
int rectHeight = 2;

color rectColor = color(0);

void setup(){
    size(20, 20);
    background(255);
}

void draw(){
    fill(rectColor);
    rect(x, y, rectWidth, rectHeight);
    y = y+1
}
```
Live Code

Because `draw()` repeats constantly, we can create change and movement using math.

When we add 1 to `y` with each loop, the dot’s `y` position slowly increases, so the dot slides down the screen.
Live Code

Because `draw()` repeats constantly, we can create change and movement using math.

When we add `y = y+1`, the dot begins to slide down the canvas.

**Why does it leave a trail?**

Why might you want to put `y = y+1` at the **bottom of the function**?
Live Code

The dot leaves a trail because we’ve only drawn the background once, during setup.

Let’s re-draw the background at the beginning of each draw loop.

What would happen if we put `background(0)` at the end of the loop?
Wrapping up

Take a minute to explore Processing.

Draw a rectangle.

Then draw a line.

Make your rectangle change color as the sketch runs.

Make one point of your line move as the sketch runs.
Homework

- Research some artists or designers working with code. Bring in an example of work you’re interested in!
- Pseudocode: think of something simple you would like to draw in Processing. Decompose it and write the pseudocode for it.
- Bonus: actually program it (or try)
  - Look at the docs: https://processing.org/reference/
  - Good places to start: background(), fill(), rect(), triangle(), ellipse(), line()
Resources

https://processing.org/tutorials/

Dan Schiffman: The Coding Train

https://www.amazon.com/Learning-Processing-Second-Programming-Interaction