Intro to Coding

Fall 2018 - Class 0

Doug Brantner

September 20, 2018

Contents

Contents

Ground Rules

- No such thing as a stupid question!
- Never copy/paste!
- Type every character of your program.

Useful Supplies

- Notebook
- Pen/Pencil
- USB Thumb Drive to save work

Saving Work

At the moment, we have 3 options:

- Desktop folder
- Student Shared folder
- USB Thumb Drive

You can also email your files to yourself, but this is not recommended.
Saving in at least 2 locations is recommended for a **backup**.
Always back up your work!!!

Contents

Make a New Folder

- Open the Terminal
- Follow along with these commands:

```
>> pwd
>> man pwd          # manual page, very helpful!!!
>> cd ~/Desktop
>> mkdir ____________          # new folder name
>> cd ____________             # same name
>> echo "Hello Linux" > hello.txt
>> cat hello.txt
```

- This folder is where you will store your projects

Contents

Processing

- Based on Java
- Runs on most computers (PC, Mac, Linux)
- Makes graphics, animation, games, etc. super easy
- It's free! www.processing.org
- Tons of Example programs included
- Sometimes helpful to search for "proce55ing"

Hello World!

```
void setup() {
    println("Hello World!");
}
```

- **setup** is a Function, or in Java, a **Method**
- **void** means that **setup** has no return value
- **Curly Braces** mark the beginning and end of **setup**
- **println** is a function called by **setup**
- **"Hello World!"** is a **String** argument to **println**
- A **semicolon** marks the end of each instruction

Basic Structure

```
void setup() {
    // Setup runs once at the beginning
}

void draw() {
    // Draw loops infinitely
}
```

Reserved Words

Reserved Words have special meanings in a Programming Language.

- **void** is a Java reserved word
  - So are **int**, **float**, **double**, **String**, **for**, **while**, **if**
  - So are $+, -, *, /, =$
  - Parenthesis (), Square Brackets [], and Curly Braces {} all have different special meanings.
- **setup** and **draw** are Processing reserved words
  - So are **width**, **height**, **color**, **print**
- And lots more! See the Documentation for a full list
- Capitalization matters!

Variables

- A variable is a named value. In math, $x = 10$ or $y = 98.6$
- In programming, a variable must have a datatype.

```
int x = 10;
float y = 98.6;
char c = 'A';
String s = "Hello!";
boolean b = true;
```

- Reserved words CANNOT be used as variable names.
- Variable names should be descriptive. **radius** is more useful than **r**

Datatypes

- Numbers
  - **int:** integer numbers (no decimals) $-1, 0, 1, 2, 3, \ldots$
  - **float:** floating-point (decimal) numbers $3.14, 2.72, 90.7, \ldots$
  - **double:** bigger double-precision floating-point numbers
  - Numbers can be negative or positive.
- Text
  - **char:** a single character, with single quotes
  - **String:** multiple characters, with double quotes
- Logical
  - **boolean:** a binary variable: either **true** or **false**

Comments

- Comments are notes to yourself, to explain what the code is doing

```
// This is a comment.

/*    This is a
 *    multi-
 *    line
 *    comment
 */
```

- Comments are ignored by the compiler
- Comments are also useful for temporarily disabling parts of code
  - This is useful for debugging
- Write lots of comments! You'll thank yourself later!

Printing to the Console

- The **print** and **println** methods print text to the console, at the bottom of the Processing window
  - **print** always continues on the same line
  - **println** always ends with an invisible **newline** \n character, so any following statement will start on a new line
- These are *very* useful for debugging

Functions in Math

In math, a function takes an input, does something to it, and returns a value.
First, we have to define our function:

$$f(x) = 2x$$

Then we can assign the return value to a new variable y

$$y = f(x)$$

Black Box Functions

A function can also be thought of as a **black box**. This means that we know the inputs and the outputs, but we may not know exactly what happens on the inside.
For example, a washing machine takes an input of dirty clothes, washes them, and then returns clean clothes.

$$\text{wetClothes} = \text{washer}(\text{dirtyClothes})$$

Likewise, a dryer takes in wet clothes, dries them, and returns dry clothes.

$$\text{cleanClothes} = \text{dryer}(\text{wetClothes})$$

We know how to use the machines, even if we don't know how they work on the inside.

Functions II

We can combine multiple functions in one line of code.

$$y = f(x)$$

$$z = g(y)$$

is equivalent to writing

$$z = g(f(x))$$

This allows us to eliminate an intermediate variable

$$\text{cleanClothes} = \text{dryer}(\text{washer}(\text{dirtyClothes}))$$

But at the expense of being harder to read.

Java Functions: *Methods*

Functions are the basic building blocks of code. In Java, they are called **methods**, and in other languages they are called sub-routines.

```
// this function takes no arguments
// and returns no values
void setup() {
    size(640, 320);
}

// this function takes an integer argument,
// then returns its square
int square(int x) {
    return x*x;
}
```

Arguments and Return Values

Many functions accept **input parameters** or **arguments**. These are variables that are passed to the function when it is called.

```
void setup() {  // No input parameters
    int y = square(10);      // now y == 100
}

int square(int x) {    // 1 integer input
    return x*x;        // returns an integer
}
```

Many functions also **return** values. The return datatype is specified in the function definition.

Void Methods

A method that does not return a value is called a **void** method.

```
void setup() {
    size(320, 640);
}

void draw() {
    // lots of fun stuff happens here
}
```

Void methods are just as powerful as other functions; they just don't need to return anything when they're done.

Variable Scope

- The curly braces $\{\ldots\}$ mark the beginning and end of the method, or the **scope** of the method.

```
int triple(int x) {
    int y = 3*x;
    return y;
}
```

- y is a **local** variable. It is defined within the **scope** of the method.
- Both x and y are only visible within the **scope** of the method. They are not accessible anywhere else in the program.
- When the method finishes, the value of y is returned, and y itself disappears.

Global Variables

- Global variables are defined outside of any method. Therefore, **every** method can see them.
- And every method can also **modify** them.
  - This can be very useful, but also very dangerous
  - Take care when deciding if a variable should be global or not

```
int p, q;

void setup() {
    p = 100;
    q = p;
    size(p, q);
}

void draw() {
    if (p == q) { ... do something fun ... }
}
```

For Loop

A **for** loop executes a specific number of times.

```
for (int i = 0; i < 10; i++) {
    // do something ten times...
}
```

While Loop

A **while** loop can run for an unknown number of loops.
You are responsible to **create and update** the loop variable.

```
int i = 0;
while (i < 10) {
    // do something here...
    i++;    // this is very important!!
}
```

An **infinte loop** happens when the test criteria is never true.
This is a very common bug!

Contents

Homework!

- Download Processing (www.processing.org)
  - You may need to install some Java dependencies too
  - Let me know if you need help
- Try to run some of the examples, look for interesting ones
- Look through the Tutorials & Videos
- Start thinking about project ideas