Review

- Commenting your code
- Random numbers and printing messages
- mouseX, mouseY
- void setup() & void draw()
- frameRate(), loop(), noLoop()
- Arcs, curves, bezier curves, beginShape/endShape
- Example Sketches
- Dropbox
- Assignment #1

void mousePressed() {
    // Called when the mouse is pressed
}

void mouseReleased() {
    // Called when the mouse is released
}

void mouseClicked() {
    // Called when the mouse is pressed and released
    // at the same mouse position
}

void mouseMoved() {
    // Called while the mouse is being moved
    // with the mouse button released
}

void mouseDragged() {
    // Called while the mouse is being moved
    // with the mouse button pressed
}

void keyPressed() {
    // Called each time a key is pressed
}

void keyReleased() {
    // Called each time a key is released
}

void keyTyped() {
    // Called when an alpha-numeric key is pressed
    // Called repeatedly if the key is held down
}

keyCode vs. key

key
- A built-in variable that holds the character that was just typed at the keyboard

keyCode
- A built-in variable that hold the numeric code for the keyboard key that was touched

All built-in keyboard interaction functions ...
- set keyCode to the integer that codes for the keyboard key
- set key to the character typed
- All keyboard keys have a keyCode value
- Not all have a key value

ASCII - American Standard Code for Information Interchange
More Color

colorMode(RGB or HSB);

RGB: (red, green, blue)

HSB:

- "pure color"
- "intensity"
- "lightness"

Decimal vs. Binary vs. Hexadecimal

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Hex</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>00000000</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>00000001</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>00000010</td>
</tr>
<tr>
<td>3</td>
<td>03</td>
<td>00000011</td>
</tr>
<tr>
<td>4</td>
<td>04</td>
<td>00000100</td>
</tr>
<tr>
<td>5</td>
<td>05</td>
<td>00000101</td>
</tr>
<tr>
<td>6</td>
<td>06</td>
<td>00000110</td>
</tr>
<tr>
<td>7</td>
<td>07</td>
<td>00000111</td>
</tr>
<tr>
<td>8</td>
<td>08</td>
<td>00001000</td>
</tr>
<tr>
<td>9</td>
<td>09</td>
<td>00001001</td>
</tr>
<tr>
<td>10</td>
<td>0A</td>
<td>00001010</td>
</tr>
<tr>
<td>11</td>
<td>0B</td>
<td>00001011</td>
</tr>
<tr>
<td>12</td>
<td>0C</td>
<td>00001100</td>
</tr>
<tr>
<td>13</td>
<td>0D</td>
<td>00001101</td>
</tr>
<tr>
<td>14</td>
<td>0E</td>
<td>00001110</td>
</tr>
<tr>
<td>15</td>
<td>0F</td>
<td>00001111</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>00010000</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>00010001</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>00010010</td>
</tr>
</tbody>
</table>

Primitive Data Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>(true, false)</td>
<td>false</td>
<td>7</td>
</tr>
<tr>
<td>byte</td>
<td>[0..255]</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>int</td>
<td>[-2,147,483,648 ... 2,147,483,647 ]</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>long</td>
<td>[-9,223,372,036,854,775,808 ... 9,223,372,036,854,775,807 ]</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>float</td>
<td>[-3.40282347E+38 ... 3.40282347E+38 ]</td>
<td>0.0</td>
<td>4</td>
</tr>
<tr>
<td>double</td>
<td>much larger/smaller</td>
<td>0.0</td>
<td>8</td>
</tr>
<tr>
<td>color</td>
<td>{ #00000000 .. #FFFFFFFF }</td>
<td>black</td>
<td>4</td>
</tr>
<tr>
<td>char</td>
<td>a single character 'a', 'b',...</td>
<td>'u0000'</td>
<td>2</td>
</tr>
</tbody>
</table>

Variables

- A name to which data can be assigned
- A variable is declared as a specific data type
- A variable is assigned a value using '='
- Variable names must begin with a letter, "_" or "$"
- Variables can contain letters, digits, "_" and "$"

Syntax: type name; type name = expression;

```pde
int i;
float x;
int j = 12;
boolean bReady = true;
float fSize = 10.0;
color _red = color(255,0,0);
```

Rewrite randomEllipse using Variables

```pde
void draw() {
    fill( random(255), random(255), random(255) );
    ellipse(mousex, mousey, 30, 30);
}
```

```pde
void draw() {
    float R, G, B;
    R = random(255);
    G = random(255);
    B = random(255);
    fill( R, G, B );
    ellipse(mousex, mousey, 30, 30);
}
```

```pde
void draw() {
    float R = random(255);
    float G = random(255);
    float B = random(255);
    fill( R, G, B );
    ellipse(mousex, mousey, 30, 30);
} sim.```
### Using Variables

**Draws a line from last mouse position to current.**

Variables used to store last mouse position.

```java
// Variables that store the last mouse pressed position.
int lastX;  // Note where these are declared!
int lastY;

void setup() {
  size(500, 300);
}

void draw() {
  // Draw a line from the last mouse position
to the current position.
  line(lastX, lastY, mouseX, mouseY);
  lastX = mouseX;
  lastY = mouseY;
}
```

---

### Using Variables

**Orbit mouse with two shapes.**

Variables used for temporary calculated values.

```java
// Mouse orbiter
float angle;                   // Orbit angle state variable

void setup() {
  size(500, 300);
  background(255);
}

void draw() {
  background(255);
  fill(0, 0, 255);
  angle = angle + 0.3;         // Increment angle
  float dX = 30.0*cos(angle);  // Mouse position offset
dY = 30.0*sin(angle);         // Draw two orbiting shapes
elipse(mouseX + dX, mouseY + dY, 5, 5);
elipse(mouseX - dX, mouseY - dY, 5, 5);
}
```

---

### Data Type Conversion

- Variables of some types can be converted to other types.
- Type conversion function names are the types to which data will be converted.

```java
// binary(…), boolean(…), byte(…),
// char(…), float(…), str(…)
float f = 10.5;
int i;
//i = f; // Throws a runtime error
i = int(f);
println(char(65)); // Prints the character 'A'
```

### Other "things" ...

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>a series of chars in quotes &quot;abc&quot;</td>
<td>null</td>
<td>?</td>
</tr>
<tr>
<td>PImage</td>
<td>an image</td>
<td>null</td>
<td>?</td>
</tr>
<tr>
<td>PFont</td>
<td>a font for rendering text</td>
<td>null</td>
<td>?</td>
</tr>
</tbody>
</table>

```java
String message = "Hello World!";
```

---

### Images

<table>
<thead>
<tr>
<th>PImage img;</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Declares a variable to hold an image</td>
</tr>
</tbody>
</table>

```java
img = loadImage("filename");
- Loads an image from a file in the data folder in sketch folder.  
- Must be assigned to a variable of type PImage.
```

```java
image(img, X, Y, [X2, Y2]);
- Draws the image img on the canvas at X, Y  
- Optionally fits image into box X,Y and X2,Y2
```

```java
imageMode(CORNER);
- X2 and Y2 define width and height. 
imageMode(CORNERS);
- X2 and Y2 define opposite corner.
```

---

### Image Example

```java
PImage img;
void setup() {
  // Image Example
  img = loadImage("natura-morta.jpg");
}
```

```java
Image Example
```

---

---

---

---

---

---

---
Expressions
- Series of data values, variables and/or sub-expressions, related by operators and functions, and grouped by parentheses.
- Expressions are automatically evaluated and replaced by the final evaluated value.
- Expressions can be assigned to variables using “=”
  - Expression is always on right
  - Variable name is always on left

\[
\text{variable\_name} = \text{expression};
\]

Operators
Symbols that operate on one or two sub-expressions. Infix, prefix, or postfix

- Mathematical (\(+\), \(-\), \(*\), \(/\), \(\ldots\))
  - Perform standard mathematical operations (PEMDAS)
- Relational (\(<\), \(>\), \(<=\), \(>=\), \(!=\), \(\ldots\))
  - Test relationship between related expressions.
  - Always returns a boolean value (true or false).
- Logical (\(\&\&\), \(||\), \(!\))
  - Logical conjunction (and), disjunction (or), negation (not).
  - Always returns a boolean value (true or false).

Mathematical Operators
\(+\), \(-\), \(*\), \(/\) and …
\i\ ++\;  \; \text{equivalent to} \; i = i + 1;
\i\ += 2;  \; \text{equivalent to} \; i = i + 2;
\i\ -= 1;  \; \text{equivalent to} \; i = i - 1;
\i\ -= 3;  \; \text{equivalent to} \; i = i - 3;
\i\ *= 2;  \; \text{equivalent to} \; i = i \times 2;
i\ /= 4;  \; \text{equivalent to} \; i = i / 4;
i\ %\ 3;  \; \text{the remainder after} \; i \; \text{is divided by} \; 3 \; \text{(modulus)}

Examples:
\+ \; 2
\text{slope} = (y_2 - y_1) / (x_2 - x_1);
i++

Relational Operators
\(<\) less than
\(>\) is greater than
\(<=\) is less than or equal to
\(>=\) is greater than or equal to
\(==\) is equivalent
\(!=\) is not equivalent

Examples:
\text{true}
10 \geq 10
\'A\' != \'A\'

Logical Operators
\(\&\&\) logical conjunction (and)
  - both expressions must evaluate to ‘true’ for conjunction to evaluate to ‘true’
\(||\) logical disjunction (or)
  - either expression must evaluate to ‘true’ for disjunction to evaluate to ‘true’
\(!\) logical negation (not)
  - true \rightarrow false, false \rightarrow true

Examples:
\text{true} \&\& \text{true}
\text{true} || \text{false}
\text{!false}

Evaluating Logical Expressions

1. If I've already had two desserts, then don't serve me any more. Otherwise, I'll take another, thank you.

\(A = \text{had\_dessert\_1}, \; B = \text{had\_dessert\_2}\)

2. I'll have dessert, as long as it is not flan (A) or beef jerky (B).

\(\text{!}(A \&\& B) \; \text{or} \; \text{!A} \; || \; \text{!B}\)
**Some Built-in Mathematical Functions**

- $\sin(x)$, $\cos(x)$, $\tan(x)$, $\arcsin(x)$, …
- $\text{abs}(x)$, $\exp(x)$, $\text{pow}(x, y)$, $\log(x)$, $\sqrt{x}$, …
- $\text{max}(x_1, x_2)$, $\text{min}(x_1, x_2)$, $\text{floor}(x)$, $\text{ceil}(x)$, …
- $\text{dist}(x_1, y_1, x_2, y_2)$ → distance between two points
- $\text{norm}(\text{value}, \text{low}, \text{high})$ → normalizes a value to [0-1]

... and many more, all of which can be included in an expression.

**Evaluating Expressions**

- $1 + 2$
- $\text{pow}(\sin(x), 2) + \text{pow}(\cos(x), 2) = 1.0$
- $\text{max}(1, 2, 3) \geq 2$
- $\text{floor}(2.9) = \text{ceil}(1.8)$

```cpp
void setup()
{
  float r = 200.0;
  size( 500, 200+300 );
  background( 0.5*r, 0, 0 );
}
```