Review

• Commenting your code
• Random numbers and printing messages
• mouseX, mouseY
• void setup() & void draw()
• frameRate(), loop(), noLoop()
• Arcs, curves, bézier curves, beginShape/endShape
• Example Sketches
• Dropbox
• Assignment #1
vevents.pde
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 holds 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

<table>
<thead>
<tr>
<th>Char</th>
<th>Dec</th>
<th>Char</th>
<th>Dec</th>
<th>Char</th>
<th>Dec</th>
<th>Char</th>
<th>Dec</th>
<th>Char</th>
<th>Dec</th>
<th>Char</th>
<th>Dec</th>
<th>Char</th>
<th>Dec</th>
<th>Char</th>
<th>Dec</th>
<th>Char</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>(null)</td>
<td>0</td>
<td>(dc4)</td>
<td>20</td>
<td>( )</td>
<td>40</td>
<td>&lt;</td>
<td>60</td>
<td>P</td>
<td>80</td>
<td>d</td>
<td>100</td>
<td>x</td>
<td>120</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(soh)</td>
<td>1</td>
<td>(nak)</td>
<td>21</td>
<td>( )</td>
<td>41</td>
<td>=</td>
<td>61</td>
<td>Q</td>
<td>81</td>
<td>e</td>
<td>101</td>
<td>y</td>
<td>121</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(stx)</td>
<td>2</td>
<td>(syn)</td>
<td>22</td>
<td>( * )</td>
<td>42</td>
<td>&gt;</td>
<td>62</td>
<td>R</td>
<td>82</td>
<td>f</td>
<td>102</td>
<td>z</td>
<td>122</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(etx)</td>
<td>3</td>
<td>(etb)</td>
<td>23</td>
<td>( + )</td>
<td>43</td>
<td>?</td>
<td>63</td>
<td>S</td>
<td>83</td>
<td>g</td>
<td>103</td>
<td>{</td>
<td>123</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(eot)</td>
<td>4</td>
<td>(can)</td>
<td>24</td>
<td>( , )</td>
<td>44</td>
<td>@</td>
<td>64</td>
<td>T</td>
<td>84</td>
<td>h</td>
<td>104</td>
<td></td>
<td></td>
<td>124</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(enq)</td>
<td>5</td>
<td>(em)</td>
<td>25</td>
<td>( - )</td>
<td>45</td>
<td>A</td>
<td>65</td>
<td>U</td>
<td>85</td>
<td>i</td>
<td>105</td>
<td>}</td>
<td>125</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ack)</td>
<td>6</td>
<td>(sub)</td>
<td>26</td>
<td>( . )</td>
<td>46</td>
<td>B</td>
<td>66</td>
<td>V</td>
<td>86</td>
<td>j</td>
<td>106</td>
<td>~</td>
<td>126</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(bel)</td>
<td>7</td>
<td>(esc)</td>
<td>27</td>
<td>( / )</td>
<td>47</td>
<td>C</td>
<td>67</td>
<td>W</td>
<td>87</td>
<td>k</td>
<td>107</td>
<td>(del)</td>
<td>127</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(bs)</td>
<td>8</td>
<td>(fs)</td>
<td>28</td>
<td>( 0 )</td>
<td>48</td>
<td>D</td>
<td>68</td>
<td>X</td>
<td>88</td>
<td>l</td>
<td>108</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ht)</td>
<td>9</td>
<td>(gs)</td>
<td>29</td>
<td>( 1 )</td>
<td>49</td>
<td>E</td>
<td>69</td>
<td>Y</td>
<td>89</td>
<td>m</td>
<td>109</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(nl)</td>
<td>10</td>
<td>(rs)</td>
<td>30</td>
<td>( 2 )</td>
<td>50</td>
<td>F</td>
<td>70</td>
<td>Z</td>
<td>90</td>
<td>n</td>
<td>110</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(vt)</td>
<td>11</td>
<td>(us)</td>
<td>31</td>
<td>( 3 )</td>
<td>51</td>
<td>G</td>
<td>71</td>
<td>[</td>
<td>91</td>
<td>o</td>
<td>111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(np)</td>
<td>12</td>
<td>(sp)</td>
<td>32</td>
<td>( 4 )</td>
<td>52</td>
<td>H</td>
<td>72</td>
<td>\</td>
<td>92</td>
<td>p</td>
<td>112</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(cr)</td>
<td>13</td>
<td>( ! )</td>
<td>33</td>
<td>( 5 )</td>
<td>53</td>
<td>I</td>
<td>73</td>
<td>]</td>
<td>93</td>
<td>q</td>
<td>113</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(so)</td>
<td>14</td>
<td>( &quot; )</td>
<td>34</td>
<td>( 6 )</td>
<td>54</td>
<td>J</td>
<td>74</td>
<td>^</td>
<td>94</td>
<td>r</td>
<td>114</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(si)</td>
<td>15</td>
<td>( # )</td>
<td>35</td>
<td>( 7 )</td>
<td>55</td>
<td>K</td>
<td>75</td>
<td>_</td>
<td>95</td>
<td>s</td>
<td>115</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(dle)</td>
<td>16</td>
<td>( $ )</td>
<td>36</td>
<td>( 8 )</td>
<td>56</td>
<td>L</td>
<td>76</td>
<td>`</td>
<td>96</td>
<td>t</td>
<td>116</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(dc1)</td>
<td>17</td>
<td>( % )</td>
<td>37</td>
<td>( 9 )</td>
<td>57</td>
<td>M</td>
<td>77</td>
<td>a</td>
<td>97</td>
<td>u</td>
<td>117</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(dc2)</td>
<td>18</td>
<td>( &amp; )</td>
<td>38</td>
<td>( : )</td>
<td>58</td>
<td>N</td>
<td>78</td>
<td>b</td>
<td>98</td>
<td>v</td>
<td>118</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(dc3)</td>
<td>19</td>
<td>( ' )</td>
<td>39</td>
<td>( ; )</td>
<td>59</td>
<td>O</td>
<td>79</td>
<td>c</td>
<td>99</td>
<td>w</td>
<td>119</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
More Color

colorMode\(\text{RGB or HSB}\); 

RGB: (red, green, blue)

HSB: 
  
  hue 
  \begin{itemize} 
    \item “pure color” 
  \end{itemize} 

  saturation 
  \begin{itemize} 
    \item “intensity” 
  \end{itemize} 

  brightness 
  \begin{itemize} 
    \item “lightness” 
  \end{itemize}
<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>?</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;`

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

*Variables are both **declared** and **assigned** a value*
Rewrite randomEllipse using Variables

```java
void draw() {
    fill( random(255), random(255), random(255) );
    ellipse(mouseX, mouseY, 30, 30);
}
```

```java
void draw() {
    float R, G, B;
    R = random(255);
    G = random(255);
    B = random(255);
    fill( R, G, B );
    ellipse(mouseX, mouseY, 30, 30);
}
```

```java
void draw() {
    float R = random(255);
    float G = random(255);
    float B = random(255);
    fill( R, G, B );
    ellipse(mouseX, mouseY, 30, 30);
}
```
Using Variables

Draws a line from last mouse position to current. Variables used to store last mouse position.

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

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

void draw() { /* must exist */ }

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

Orbit mouse with two shapes.
Variables used for temporary calculated values.

// 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
  float dY = 30.0*sin(angle); // Draw two orbiting shapes
  ellipse(mouseX + dX, mouseY + dY, 5, 5);
  ellipse(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.

// 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>

... 

String message = "Hello World!";
Images

PImage img;

– Declares a variable to hold an image

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.

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

imageMode(CORNER);

– X2 and Y2 define width and height.

imageMode(CORNERS);

– X2 and Y2 define opposite corner.
Image Example

```java
PImage img;

void setup()
{
    size(500, 400);
    img = loadImage("natura-morta.jpg");
    image(img, 50, 40);
}
```
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 ( +, −, *, /, … )
  – Perform standard mathematical operations (PEMDAS)

• Relational ( <, >, ==, !=, … )
  – 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 ++; equivalent to i = i + 1;
i += 2; equivalent to i = i + 2;
i --; equivalent to i = i - 1;
i -= 3; equivalent to i = i - 3;
i *= 2; equivalent to i = i * 2;
i /= 4; equivalent to i = i / 4;
i % 3; the remainder after i is divided by 3 (modulo)

Examples:

1 + 2
slope = (y2 - y1) / (x2 - x1);
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:
true
10 >= 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 → false,  !false → true

Examples:
true && true
true || false
!false
### Evaluating Logical Expressions

| Negation ( !A ) | Conjunction "AND" (A && B) | Disjunction "OR" (A || B) |
|----------------|---------------------------|--------------------------|
| A              | !A                        | A                        |
| false          | true                      | true                     |
| true           | false                     | false                    |

#### Conjunction "AND" (A && B)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A &amp;&amp; B</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

#### Disjunction "OR" (A || B)

| A | B | A || B |
|---|---|--------|
| true | true | true |
| true | false | true |
| false | true | true |
| false | false | false |

#### Derive new tables by combining operators…

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

   A = had_dessert_1,  B = had_dessert_2

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

   !(A && B)  or  !A || !B

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>!(A &amp;&amp; B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>
Some Built-in Mathematical Functions

\[ \sin(x), \cos(x), \tan(x), \arcsin(x), \ldots \]
\[ \text{abs}(x), \exp(x), \text{pow}(x, y), \log(x), \sqrt{x}, \ldots \]
\[ \text{max}(x_1, x_2), \text{min}(x_1, x_2), \text{floor}(x), \text{ceil}(x), \ldots \]

\[ \text{dist}(x_1, y_1, x_2, y_2) \rightarrow \text{distance between two points} \]
\[ \text{norm}(\text{value}, \text{low}, \text{high}) \rightarrow \text{normalizes a value to [0-1]} \]

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

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

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