Section 7: Input and Output

Introduction: Here we will cover some basic ways to allow a user to interact with your program!

Output: Ways to display something for the user to see.

1) Drawing: You can draw shapes on the canvas and even change what you draw based on conditionals.
2) Printing: Using the commands `print()` and `println()` allows us to print text to the Console, which is the black window at the bottom of the Processing IDE. `println()` adds a new line.
3) Text: The command `text("your text",x,y);` will draw "your text" onto the drawing canvas with its *lower-left* corner at position `(x,y)`. The color of the text is set by `fill()`. The size of the text can be adjusted using `textSize()`.

Keyboard Input: Processing always detects when a key on your keyboard is pressed, as long as the drawing canvas window is active. You can use these keys in your program with the following special system variables and functions:

`key` – System variable of type `char` that contains the most recently used key (ASCII). Contains the value `CODED` if a non-ASCII key is pressed.

`keyCode` – System variable that is like `key`, but for special (non-ASCII) keys such as `UP`, `DOWN`, `LEFT`, `RIGHT`, `ALT`, `CONTROL`, and `SHIFT`.

`keyPressed()` – Function (return type `void`) that *you define*. Runs once every time a key is **pressed**.

`keyTyped()` – Function (return type `void`) that *you define*. Runs once every time a key is **pressed**, but ignores special (non-ASCII) keys.

Example:

```
void keyPressed() {
   if ( (key == CODED) && (keyCode == UP) ) {
      println("UP pressed!");
   } else if (key == 'J') {
      println("J pressed.");
   }
}
```

Mouse Input: Processing also detects mouse clicks, as long as the click occurred within the bounds of the drawing canvas. You can use these clicks in your program with the following special system variables and functions:

`mouseX` – System variable of type `int` that contains the horizontal coordinate of the mouse in the *current frame*. There is a similar variable called `mouseY`.

`pmouseX` – System variable of type `int` that contains the horizontal coordinate of the mouse in the *previous frame*. There is a similar variable called `pmouseY`.

`mouseButton` – System variable that contains the most recently clicked mouse button (either `LEFT`, `RIGHT`, or `CENTER`).

`mousePressed()` – Function (return type `void`) that *you define*. Runs once every time a mouse button is **pressed**.

`mouseReleased()` – Function (return type `void`) that *you define*. Runs once every time a mouse button is **released**.

Example:

```
void mousePressed() {
   if ( mouseButton == LEFT ) {
      print("mouse's horizontal position = ");
      println(mouseX);
   } else if ( mouseButton == RIGHT ) {
      print("mouse's vertical position = ");
      println(mouseY);
   }
}
```

Exercises:

1) Complete the following code to print a red "`Hello, world!`" onto a white drawing canvas. Feel free to play around with the numbers in Processing, but the text should take up almost all of the canvas.

```
size(500,100);
background(_____________);
fill(_____________);
textSize(_____);
text(_____________________, ____, ____);
```

2) Write Processing code below that draws the last-typed key onto the middle of the drawing canvas. Make sure that the previous text is erased once a new key is pressed.

```
void draw() {}  // empty draw() needs to be part of program
```

4) Write a `mousePressed()` function in Processing that prints the difference in positions *between mouse clicks*. For example, if I clicked on (0,0) and then (30,50), it should print something along the lines of "`difference in X = 30`" and "`difference in Y = 50`" to the Console.

5) Continue working on the homework titled "Jumping Monster." [*individual*]