Review

- Expressions and operators
- Iteration
  - while-loop
  - for-loop

Coding styles

- Headers
- Comments

- Indentation
- Parentheses
- Spacing

text()

- Strings can be drawn on a sketch using the text() function.
- Can set text position, font, size, alignment, …
- Font files are loaded from the data folder.

```
// Set attributes
textSize( sizeInPixels );
textAlign( {LEFT | CENTER | RIGHT}
    [,{TOP, BOTTOM, CENTER, BASELINE}] );
fill( color );

// Render text
text( string, X, Y );
text( string, X, Y, width, height );
```

```
// text
void setup() {
  size(500, 500);
  noLoop();
}

void draw() {
  // bounding box
  stroke(0);
  fill(255);
  rect(50, 50, 400, 400);

  // text options
  fill(0);    // black text
  text("Default", 50, 50, 400, 400);
  textAlign(CENTER);
  text("CENTER", 50, 50, 400, 400);
  textAlign(RIGHT);
  text("RIGHT", 50, 50, 400, 400);
  textAlign(CENTER, CENTER);
  text("CENTER-CENTER", 50, 50, 400, 400);
  textAlign(RIGHT, BOTTOM);
  text("RIGHT-BOTTOM", 50, 50, 400, 400);
  textAlign(LEFT, BOTTOM);
  text("LEFT-BOTTOM", 50, 50, 400, 400);
}
```

Iteration

Repetition of a program block

- Iterate when a block of code is to repeated multiple times.

Options

- while-loop
- for-loop

Iteration: while-loop

```
while ( boolean_expression ) {
  statements;
  // continue;
  // break;
}
```

- Statements are repeatedly executed while the boolean expression remains true.
- To break out of a while loop, call **break;**
- To continue with next iteration, call **continue;**
- All iterations can be written as while-loops.

Iteration: for-loop

```
for ( initialization; continuation_test; update)
{
   statements;
   // continue;      // Continues with next iteration
   // break;         // Breaks out of loop
}
```

- A kind of iteration construct
- initialization, continuation test and increment commands are part of statement
- To break out of a loop, call **break;**
- To continue with next iteration step, call **continue;**
- All for loops can be translated to equivalent while loops

```
void mousePressed() {

  for (int i = 0; i < 10; i++ ){
    print( i );
  }
  println();

}

void draw() { }
```

```
void mousePressed() {
  for (int i = 0; i < 10; i++ ) {
    if ( i % 2 == 1 ) continue;
    print( i );
  }
  println();
}

void draw() { }
```

Functions Informally

- A function is like a subprogram, a small program inside of a program.
- The basic idea – we write a sequence of statements and then give that sequence a name. We can then execute this sequence at any time by referring to the name.
- Function definition: this is where you create a function and define exactly what it does
- Function call: when a function is used in a program, we say the function is *called*.
- A function can only be defined once, but can be called many times.

Function Examples

```
void setup() { … }
void draw() { … }


void line( float x1, float y1, float x2, float y2) { … }
… and other graphic functions


float float( … )
… and other type-conversion functions

… etc.
```

Functions

Modularity

- Functions allow the programmer to break down larger programs into smaller parts.
- Promotes organization and manageability.

Reuse

- Enables the reuse of code blocks from arbitrary locations in a program.

Function Parameters

- Parameters (arguments) can be "passed in" to function and used in body.
- Parameters are a comma-delimited set of variable declarations.
- Parameters act as input to a function.
- Passing parameters provides a mechanism to execute a function with many different sets of input
- We can call a function many times and get different results by changing its parameters.

What happens when we call a function?

- Execution of the main (calling) program is suspended.
- The argument expressions are evaluated.
- The resulting values are copied into the corresponding parameters.
- The statements in the function's body are executed in order.
- Execution of the main program is resumed when a function exits (finishes).

More Examples

```
/*****************************************
** This function squares a number
** Inputs: a value to be squared
** Outputs: returns the square of the number
**              provided
*****************************************/
double square (double n) {
 return n*n;
}
```

```
/*****************************************8
** Function: FindMinimum ()
** Finds the minimum of two integers
** Inputs: integers n1 and n2 to be compared
** Outputs: returns the smaller of n1 and n2.
*****************************************/
int findMinimum (int n1, int n2) {
 int min;
 if(n1<n2) {
  min = n1;
 }
 else {
  min = n2;
 }
 return min;
}
```

Functions that return values

- The return value of a function is the output of a function.
- A function evaluates to its return value.
- Function must return a value whose type matches the function declaration.

```
return_type function_name( argument_decl_list ) {
    statements;
    return value;
}
```

Variable Scope

The part of the program from which a variable can be accessed.

Rules:

1. Variables declared in a block are only accessible within the block.
2. Variables declared in an outer block are accessible from an inner block.
3. Variables declared outside of any function are considered global (available to all functions).

Variable Lifetime

- Variables cannot be referenced before they are declared.
- Variables can be declared in…
  - the global scope
  - the body of a function or constructor
  - the arguments of a function or constructor
  - a statement block (for, while, if, …).
- A variable is created and initialized when a program enters the block in which it is declared.
- A variable is destroyed when a program exists the block in which it was declared.

```
int v1 = 1;

void setup() {
  int v2 = 2;

  for (int v3=3; v3 <= 3; v3++) {
    int v4 = 4;
    println("-------------");
    println("v1=" + str(v1));
    println("v2=" + str(v2));
    println("v3=" + str(v3));
    println("v4=" + str(v4));
    //println("v5=" + str(v5));
  }

  int v3 = 6;
  println("v3=" + str(v3));

  aFunction(v2);
}

void aFunction(int v5) {
  println("-------------");
  println("v1=" + str(v1));
  //println("v2=" + str(v2));
  //println("v3=" + str(v3));
  //println("v4=" + str(v4));
  println("v5=" + str(v5));
}

void draw() { }
```

- What is printed?
- What happens if the second v3 declaration is removed?
- What would happen if the v5 print statement is executed?
- What would happen if commented statements in aFunction were called?