Section 4: Variables

Introduction: A variable is a storage location that is referenced by name and contains a value of a particular data type. In lecture, we likened this to containers that come in different molds: the name lets you know which container to use and each container can only hold contents/values that fit in that particular mold.

Variables are most often used for (1) storing and manipulating a value that will change over the course of your program and/or (2) giving a meaningful name to a particular value used in your program.

Data Types: The following is a list of the most commonly used data types in CSE 120.

`int` – Stores an integer: a positive or negative number with no fractional part (e.g. -7, 0, 120).
`float` – Stores a positive or negative number that can have a fractional part (e.g. 3.14, -1.0).
`color` – Stores a color triplet, which can be defined using the command `color(red,green,blue)`.
`char` – Stores a character: a single symbol that you can find in text files (e.g. `'a'`, `'C'`, `'5'`, `'$'`, `'_'`).
`boolean` – Stores a logical value, which can only be `true` or `false`.

Variable Declaration: Before you can use a variable, you must declare it (i.e. ask Processing to create the container). Because programs are executed in sequence, the variable declaration must occur *before* you try to use the variable. A variable declaration is of the form: `var_type var_name;`

- `var_type` is one of the data types listed above and indicates the mold of your container (and determines what values it can store).
- `var_name` is the name associated with this variable. You should try to use intuitive names that indicate what the variable is used for. Variable names can include letters, numbers, and non-arithmetic symbols, but *cannot start* with a number.

When Processing creates a new variable/container, it starts "empty." Processing will complain if you try to use a variable when it is empty. To fill a variable while you are declaring it, you may initialize it like so:
`var_type var_name = initial_value;`

Examples:
```
int len = 50;
color purple = color(128, 0, 128);
```

Using Variables: You can freely use variable names in your program and they will automatically be substituted with the *current* values stored in those variables!

Examples:
```
ellipse(40, len+20, len, len);  // circle of diameter 50 at (40,70)
background(purple);             // set background to (128, 0, 128)
```

To *change* the value stored in a variable, we use an assignment statement: `var_name = new_value;`
Although we use an equals sign, you should think of the effect/behavior as `var_name` ← `new_value`.

Examples:
```
len = len + 10;                 // new value of len is 50+10 = 60
purple = color(75, 42, 133);    // darker shade of purple
```

Exercises:

1) For the following values, what data type does a variable need to be in order to store them?

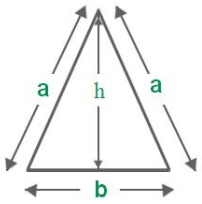
`true`	_____	`color(0,0,255)`	_____
42	_____	2.71	_____

2) Write out a Processing statement below to declare and initialize a variable that holds our course number (120). Make sure that you give it an *intuitive* and *legal* variable name.

3) Describe what will be drawn when the following Processing program is run:

```
int i = 10;
int j = i * 3/2;
color c = color(255,0,0);
i = i + j;
fill(c);
noStroke();
rect(i, j, 30, 30);
```

4) Complete the program below that draws an isosceles triangle with the top point at (**topX**, **topY**) no matter what we initialize the following variables to (*note that declaring multiple variables on one line, as shown here, does work as long as they are the same data type*):




```
int h = 60, b = 60;
int topX = 100, topY = 100;

triangle(_____,_____,_____,_____,_____,_____);
```

5) Complete the commands described below for `float x` and `float y` using the `min()` and `max()` functions.

```
// increases x in increments of 2, but stops at 120

x = _____________________________;

// decreases y in increments of 3.5, but stops at -1

y = _____________________________;
```

6) Go to the course website and get started on the homework titled “Lego Family.” [*individual*]