Section 14: Files

Introduction: Files are useful for reading and writing data because they exist *outside* of your programs. This allows for lots of different possibilities: (1) you can store the result of your program execution somewhere more permanent, (2) you can edit the data values between program executions, (3) you can pass data in files between *different* programs, or (4) you can change the *amount* of data your program reads by modifying the file contents and length.

Importing a File: There are more general ways to import files but in this course, we will be using data files in comma-separated values (CSV) format. The simplest way to import these kinds of files is to call the special function `loadStrings(String filename)` and store its return value into a `String` array. Each *line* of the file will be stored in a different index of the array (*i.e.* the 1st line will be in index 0, the 2nd line in index 1, etc.) as a `String`.

- It is easiest if you put your CSV file into your Processing project folder and then you can just use the filename as the argument.
- Like images, files should be imported *once* at the beginning of your program (*i.e.* inside **`setup`**`()` or at the beginning of a *static* program).

As the name implies, each row/line of a CSV file contains values with columns separated by commas. So we will want to *split* each row into its values using the function `split(String s, char delim)`. This function breaks `s` into pieces (returns a `String[]`) using `delim` as the delimiter, a boundary marker between values.

- Note that `split()` takes a `String`, not a `String[]`, so it should be used on a *row* of imported data, not the whole imported file.

Example:

```
String[] importedData, header;
void setup() {
   importedData = loadStrings("data.csv");
   header = split(importedData[0], ",");     // split header/1st row
}
```

Converting Data: `loadStrings()` imports your CSV file as a `String` array and `split()` returns the values in a row in a `String[]` as well. However, if the file was not intended to be text, you will need to first convert the data before you use it. Luckily, Processing has a handy set of *conversion* functions that will do this for you! These conversion functions are intuitively named: `char()`, `float()`, `int()`, and `str()`.

Example:

```
String row = "120,3.14,hi";
String[] vals = split(row,",");  // split into array of Strings
int i = int(vals[0]);            // stores 120
float f = float(vals[1]);        // stores 3.14
String s = vals[2];              // stores "hi" - no conversion needed
```

Exporting to a File: To save data to a file, we can use `saveStrings(String filename, String[] data)`. If there is already an existing file at the path `filename`, this will *overwrite* that file so be careful! For CSV files, `filename` should end in `".csv"` and `data` should be an array of `String`s, each using commas as delimiters. Each index of `data` is written into the file as a separate line/row.

Example:

```
int[] row1 = {1, 20, 120, -5};
float[] row2 = {0.33, 1.41, 1.62, 2.71, 3.14};
String[] data = {str(row1[0]), str(row2[0])};   // include 1st columns
int i = 1;
while (i < row1.length) {    // skip 1st entry
   data[0] = data[0] + ", " + str(row1[i]);     // add commas and cols
   i = i+1;
}
int i = 1;
while (i < row2.length) {    // skip 1st entry
   data[1] = data[1] + ", " + str(row2[i]);     // add commas and cols
   i = i+1;
}
saveStrings("myData.csv", data);
```

Exercises:

1) Go to the course website and find this section on the Course Schedule. Download "`file_ex.pde`" and "`animals.csv`" to your computer and put both in a folder called `file_ex`.
   a) Open `animals.csv` in a *text editor* (*e.g.* VS Code) to see what a CSV file looks like to Processing.
   b) Open `file_ex.pde` in Processing and run it. It should print the word "**`film`**" to the console.
   c) Read through the code and its comments and try to figure out what Line 23 (the `print()` call) is doing.
   d) Once you think you know how it works, go to `animals.csv` and modify *only one entry* so that running `file_ex.pde` will now output "**`file`**" to the console instead. Ideally, you would use an actual animal name! (https://en.wikipedia.org/wiki/List_of_animals_by_common_name)
   e) Below, write your changed entry: `old_animal → new_animal`

2) [optional - tricky!] Now modify only Line 23 (the `print()` call) of `file_ex.pde` in order to get the program to print the word "**`best`**" to the console. Only two small changes are needed, but you'll want to stare at `animals.csv` a while (without changing it!) to identify the pattern that gets you "**`best`**". Write your new Line 23 below:

3) Go to the course website and continue working on the lab titled "Word Guessing." [*partners*]