LAB 11

INTRODUCTION TO SERVER-SIDE DEVELOPMENT WITH PHP

What You Will Learn

- How to run PHP code to using XAMP as your local development server
- How to make use of variables and conditionals in PHP
- How to write PHP functions

Approximate Time

The exercises in this lab should take approximately 50 minutes to complete.

Fundamentals of Web Development, 2nd Ed

Randy Connolly and Ricardo Hoar

Textbook by Pearson
http://www.funwebdev.com

Date Last Revised: Feb 20, 2017

SETTING UP PHP

PREPARING DIRECTORIES

1. If you haven't done so already, create a folder in your personal drive for all the labs for this book.

   Note: To complete the labs moving forward that use PHP, you will have to have access to a web server. While the HTML files can be loaded from disk, PHP files must be interpreted by the web server which then outputs HTML.

   Your instructor may have guidance on accessing a shared host through your institution. Alternatively consider using an easy-to-set-up all-in-one tool like WAMP, EasyPHP, or MAMP if you are not comfortable installing and configuring Apache and PHP from scratch.

2. From the main `labs` folder (either downloaded from the textbook's web site using the code provided with the textbook or in a common location provided by your instructor), copy the folder titled `Lab11` to your course folder created in step one.

This lab gets you started with setting up your environment and helps you write your first PHP scripts. In order to keep the examples small, this lab omits much of the CSS and HTML from previous chapters to focus on basic program syntax and simple examples. Subsequent chapters reintroduce more of those elements.

SYSTEM CONFIGURATION

EXERCISE 11.1 — INSTALL LAMP

1. If you already have access to a shared host, then you can skip this step for now. Eventually you will at some point need to install a development environment locally.

2. For production servers LAMP is normally installed and configured one service at a time. You will need to install the latest Linux OS, then ensure httpd and mysqld daemons are installed and running (see chapter 22). In addition you will have to configure apache to serve files from a particular directory (often /var/www/html).

   For local Windows development machines, you might consider using XAMPP (or WAMP, or MAMP). When you start these systems, the http and the mysql daemons are started for you, and the configuration is made easier.

   Since these tools are always changing and updating, you may have to follow updated directions from the XAMPP/WAMP/MAMP website for your particular OS.

   The remainder of this exercise walks you through setting up XAMPP.

3. To set up XAMPP, download the XAMPP package for your operating system from their website. Run the installation package, accepting the installation location, and options

provided, and then start the XAMPP server.

4 Start the control panel (as shown in Figure 8.1) and click Start next to Apache. A green box will show it is running. By clicking the Explorer button you a file manager window will open to the root folder for xampp (in my case c:/xampp/). The subfolder htdocs, is where you will put the files you want to interpret as PHP. That root folder is associated with the URL http://localhost/.

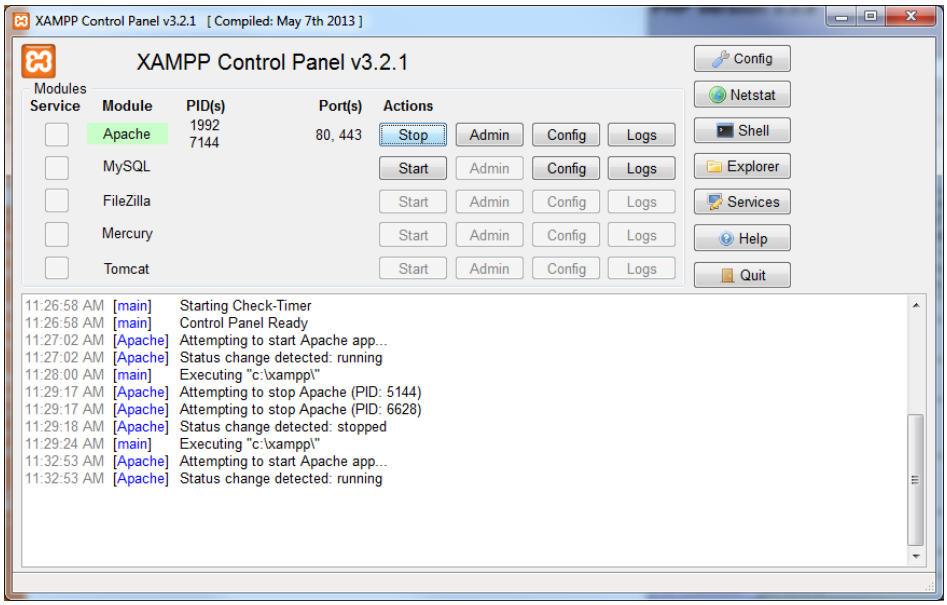


Figure 8.1 – Screenshot of the XAMPP Control Panel

The administration window will appear within a browser window. It allows you to make configuration changes, test code, and setup the local location for your PHP test files.

Your configuration for XAMPP may be different than that shown here.

Figure 8.2 – Screenshot of the XAMPP Administration window

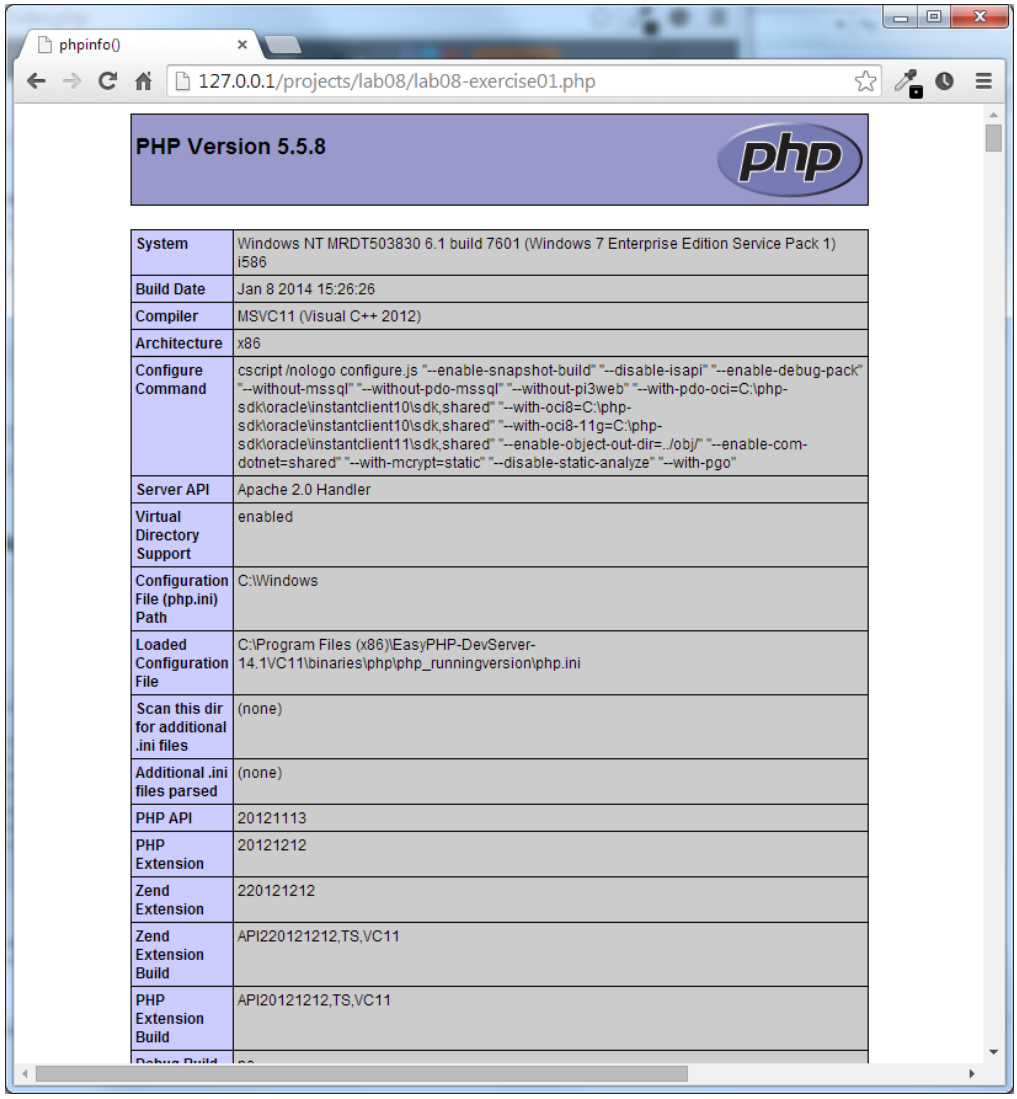
5 You have been provided with a file called Lab11-exercise01.php. It simply contains one line of code: `phpinfo()`.Place that file into the directory served by XAMPP (htdocs), optionally in a subfolder (say Lab11).

If your server is installed correctly you will be able to see out similar to that in Figure 8.3 by surfing to a location in the browser associated with your file, in our case:

localhost/Lab11/Lab11-exercise01.php

Note, rather than typing in this URL, you can access it by clicking the Local Web link via the EasyPHP administration window or by choosing the Local Web option accessible by right-clicking on the EasyPHP icon in the status bar.

6 Copy the other starting files for this lab into your Lab11 subfolder.



System	Windows NT MRDT503830 6.1 build 7601 (Windows 7 Enterprise Edition Service Pack 1) i586
Build Date	Jan 8 2014 15:26:26
Compiler	MSVC11 (Visual C++ 2012)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--disable-static-analyze" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	C:\Program Files (x86)\EasyPHP-DevServer-14.1VC11\binaries\php\php_runningversion\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20121113
PHP Extension	20121212
Zend Extension	220121212
Zend Extension Build	API220121212,TS,VC11
PHP Extension Build	API20121212,TS,VC11

Figure 8.3 – Completed Exercise 11.1– phpinfo() is interpreted, rather than output as text.

EXERCISE 11.2 — YOUR FIRST PHP SCRIPT

1 Examine `Lab11-exercise02.php` in a text editor and then load it in a browser (that is, using the same technique as in the previous exercise). You should see that the PHP `echo` statement prints out a string and that an `echo` statement outside of the PHP tags does not get executed.

2 Add an `echo` statement inside of the PHP tags that outputs the data and time.

```
echo "This page was generated: " . date("M dS, Y");
```

The dot operator in PHP is used to concatenate values. Your page will now also output a

String like: Jun 19th, 2015

3 Examine the markup received within the browser (e.g., View Source).

PHP is processed by the server and is not sent to the browser.

4 Modify the code you just entered as follows and then test in browser.

```
echo "This page was generated: " . date("M dS, Y") . "<hr/>";
```

Notice that we can programmatically output HTML via the `echo` command.

5 Modify the previous line by removing one of the dot operators and test.

This will generate an error. As long as your error reporting settings are at the default setting, you should see a syntax error message in the browser along with an indication of the line number of the error.

6 Fix the syntax error and retest.

7 Modify the code as follows and then test in browser.

```
$d = date("M dS, Y");
echo "This page was generated: " . $d . "<hr/>";
```

You should notice no change in the browser. This new version differs in using a variable (`$d`). Variables can be named anything but must begin with the $ symbol.

8 Modify the code as follows and then test in browser.

```
$date = date("M dS, Y");
echo "This page was generated: " . $date . "<hr/>";
```

You should again notice no change in the browser. The new variable name (`$date`) is to remind you that variables begin with the $ symbol, while functions have brackets after their name. So, in this example, the variable `$date` is assigned the value returned by the function `date()`.

9 Experiment with the string passed into the `date()` function. See if you can make the following formatted string (note the day will be today)

```
Wednesday, February 26th , 2014 15:43:22
```

You will need to make use of documentation from:
http://ca1.php.net/manual/en/function.date.php

10 To calculate the number of days remaining in the year, consider the format string "z" passed to the `date()` function. For instance, `date("z")` returns the numbers of days elapsed this year, so we can calculate how many days are remaining by subtracting it from 365.

```
$remaining = 365 - date("z");
echo "There are ". $remaining . " days left in the year";
```

11 The above calculation does not work if the current year is a leap year. Let us pretend that the current year is a leap year by correcting the above calculation to account for leap years.

Hint: leap years have an extra day in the year. Simply add one to the calculation!

Quick Tour of PHP

Exercise 11.3 — PHP Variables

1 Open and examine the file Lab11-exercise03.php in an editor of your choice.

Notice that this file already has defined within it several PHP variables already. As we progress through the book, such variables will later be populated from arrays, files, and then databases.

2 Use the PHP `echo` statement to output the relevant PHP variables so that your page looks similar to that shown in Figure 8.4. Note: the CSS styling has already been provided.

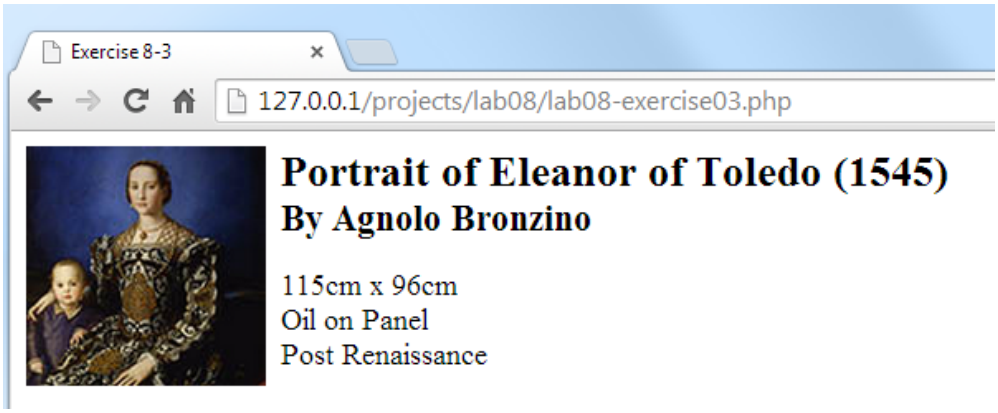


Figure 8.4 - Preliminary appearance for Exercise 11.3.

3 Add the following new variable to your page (that is, after the other variables):

```
$medium = "Oil on Panel";
$era = "Post Renaissance";
$dimensions = $width . "cm x " . $height . "cm";
```

4 Modify your dimension output markup to use this new variable. Test in browser.

Exercise 11.4 — PHP Operators

1 Open the file Lab11-exercise04.php in an editor of your choice.

2 Inside you will find some initial HTML to output a message regarding a person's age.

Create a new set of `<?php ?>` tags after the `<h1>` element. Define two variables inside of

the PHP tags as follows.

```
<?php
$birthday = mktime(0,0,0,1,15,2004); //Jan 15, 2014 00:00:00
$today = time(); // current time in seconds since 1970.
?>
```

You can enter any date that interests you, for instance, your birthday. The parameters to `mktime()` are: Hours, Minutes, Seconds, Month, Days, Year

3 You can calculate the time elapsed from the first date (the birthday) to the present day in seconds by subtracting them and storing the result in an new variable:

```
$secondsOld = $today - $birthday;
```

4 Echo the starting date before the `<ul>` in its own `<p>`:

```
echo "<p>Time elapsed since " . date("M d, Y",$birthday) . ":</p>";
```

5 Now calculate and display how many days, months and years those seconds represent.

To calculate the age in days you would output:

```
echo $secondsOld/(60*60*24);
```

6 Calculate the number of elapsed months and years so you get output similar to Figure 8.4. For simplicity sake, assume 30.4 days per month and 365.242375 days per year.

Your numbers will be different than that shown in Figure 8.4. Why?

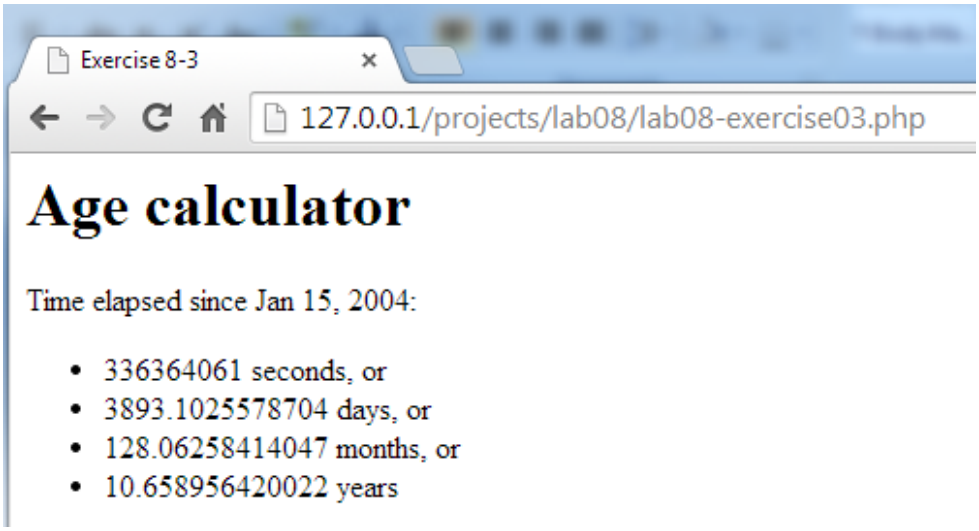


Figure 8.4 - Completed Exercise 11.4 - The output of the age in years, months and seconds

EXERCISE 11.5 — PHP OUTPUT

1 The last exercise demonstrated some simple calculations, but the output of the numbers suffered, since many decimal places were showing. Continue working on the file from Exercise 11.4.

2 Replace the number of days, months, and years using the `number_format()` function. Consult http://php.net/number_format for more information.

Number of second and days should have no decimal places; months should have one and years two, as shown in Figure 8.4.

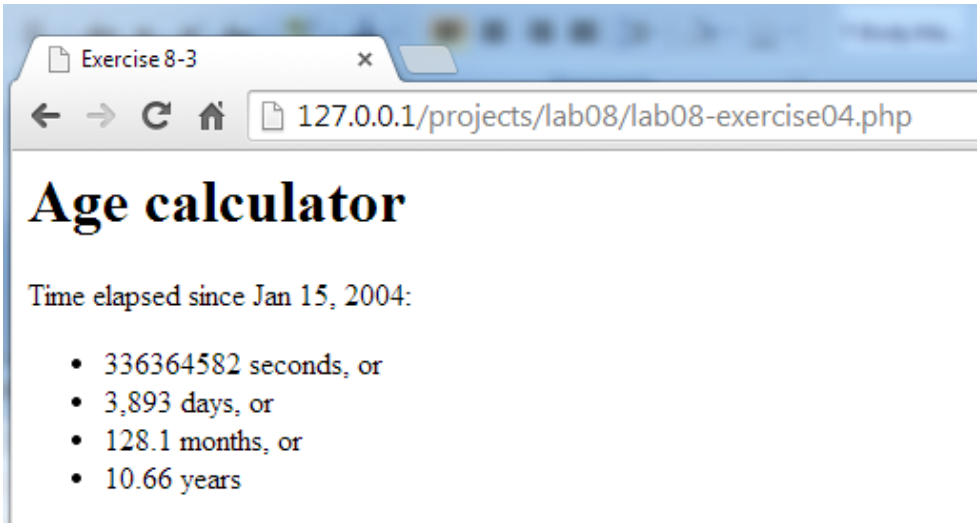


Figure 8.4 – Exercise 11.4 complete.

PROGRAM CONTROL

EXERCISE 11.6 — PHP CONDITIONALS

1 This exercise will use conditionals to format a greeting differently depending on how long until the hard-coded birthday. This exercise builds on your completed Exercise 11.3.

2 Modify the variable block in your Exercise 11.3, to add the following conditional statement.

```
$era = "Post Renaissance";
if ($year < 1530)
   $era = "Renaissance";
```

3 Test in browser. Why didn't the output change?

It didn't change because the hard-coded year is not less than 1530.

4 Change the value of the `$year` variable to 1520 and test. The era should now display as Renaissance.

5 Change the value of the `$year` variable to 1530 and test. Why did the era switch back to Post Renaissance?

6 Add the following, change the `$year` to 1630, and test.

```
$era = "Post Renaissance";
if ($year < 1530)
   $era = "Renaissance";
else
   $era = "Baroque";
```

Your page should now display Baroque as the era.

7 Make the following changes and test.

```
$era = "Post Renaissance";
if ($year < 1530) {
   $era = "Renaissance";
} else if ($year > 1600) {
    $era = "Baroque";
}
```

While this doesn't change the output, the addition of { } braces is necessary if our conditional block needs to have more than a single line of code. As well, notice that we can chain together a series of conditional tests.

8 Add the following, change the `$year` to 1200, and test. Why doesn't the era display change?

```
$era = "Post Renaissance";
if ($year < 1530) {
   $era = "Renaissance";
} else if ($year > 1600) {
    $era = "Baroque";
} else if ($year < 1400) {
    $era = "International Gothic";
}
```

The reason is due to how program flow works within a series of conditionals. The number 1200 is indeed less than the number 1530, so the condition `$year < 1530` evaluates to true, and thus the era is set to Renaissance.

9 Fix the conditional so that it works properly for years less than 1400. Test it with the years 1200, 1450, 1550, and 1650.

EXERCISE 11.7 — PHP LOOPS

1 Loops in PHP lend themselves nicely to building lists and tables. Open and examine Lab11-exercise07.php, and see that it currently outputs a simple table with the days of the week for headers.

2 Using the `date()` function, determine the current month as a word (e.g., February). Weave that variable into the HTML above the table (maybe with an extra `colspan` row)

3 Your first attempt at building a calendar will be by looping through all 31 days and outputting each in a `<td>` cell. Every 7 days, we will add a new row to the table. Notice the use of the mod or % operator.

```
while ($day<=31) {
   //when we need a new row go ahead.
   if ($day%7==0) {
      echo "</tr><tr>";
   }
   echo "<td>".($day+1)."</td>";
   $day++;
}
```

You should see output similar to

February						
Sun	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

4 Since most months don't start on Sunday, and they don't all have 31 days we will now improve our loop to account for those things.

Firstly, determine how many days are in the month, and change our loop to loop only that many times. Hint: Change the constant 31, to a value determined by the date function.

5 Determine which day of the week the 1st is as follows:

```
$dayOne = date("w",mktime(0,0,0,date("n"),1, date("Y")));
```

The value, 0-6 tells us if the first was Sunday (0) through Saturday (6).

Add a flag to our earlier loop so that the first time through we print empty cells (`<td></td>`) until we reach the first day of the month. Your output table should look like that in Figure 8.5

Note: You may need to change some logic in your loop

February						
Sun	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	

Figure 8.5 – Completed Exercise 11.7

EXERCISE 11.8 — WRITING FUNCTIONS

1 Functions allow us to group code into modules that can be reused.

In this case we will write a function that converts between various kitchen units (cups, teaspoons, etc).

Begin by opening Lab11-exercise08.php

2 Inside the empty `<?php ?>` tags define a function named `convertUnits()` that takes 3 parameters. To begin make the function look like:

```
function convertUnits($startVal, $startUnits, $endUnits) {
   return "???";
}
```

3 Now write code to handle converting values from millilitres to cups and ounces.

Add the following constants to your new function.

```
$mlToOz = 0.033814;
$mlToCup = 0.00422675;
```

Also add conditional logic to ensure that the starting units are "ml", and the end values are either "cups" or "oz".

4 Integrate the function so that it gets called each time though the loop with the correct parameters. Since `$i` represents millilitres, you should be calling the function as follows:

```
for($i=50;$i<=1000;$i+=50){
   echo "<tr>";
   echo "<td>$i</td>";
   // replace the ??? with the calls to convertUnits function
   echo "<td>" . convertUnits($i,"ml","cups") . "</td>";
   echo "<td>" . convertUnits($i,"ml","oz") . "</td>";
   echo "</tr>";
}
```

5 Now run your program and your table of output should resemble that in Figure 8.6. Notice that the cups and ounces values have only two decimal values of precision.

Making and using functions

milliliters	Cups	Ounces
50	0.21	1.69
100	0.42	3.38
150	0.63	5.07
200	0.85	6.76
250	1.06	8.45
300	1.27	10.14
350	1.48	11.83
400	1.69	13.53
450	1.90	15.22
500	2.11	16.91
550	2.32	18.60
600	2.54	20.29
650	2.75	21.98
700	2.96	23.67
750	3.17	25.36
800	3.38	27.05
850	3.59	28.74
900	3.80	30.43
950	4.02	32.12
1000	4.23	33.81

Figure 8.6 – Completed Exercise 11.8