Formatted Output

Another essential skill is being able to print clean looking output. For example, if we are printing an amount of money, this variable is probably stored as a real number (e.g. double data type in Java or float in Python). By default, when you print a real number, you get too much precision. Formatted output is a way that you can precisely control how many digits get printed and where.

Here is an example. Suppose you start with $1.12 and you are curious how much money you would have after each of the first 10 years of an investment that promises to double your money every year. (It's got to be a scam, but we'll ignore that for now.)

Here is how the code would look like in Java and Python.

```
// Java version
public class Demo
{
  public static void main(String [] args)
  {
    double amount = 1.12;

    for (int i = 1; i <= 10; ++i)
    {
      amount *= 2.0;
      System.out.printf("Year %d amount is $ %.2f\n", i, amount);
    }
  }
}
```

```
# demo.py - Python version
amount = 1.12

for i in range(1, 11):
    amount *= 2.0
    print("Year {0:d} amount is $ {1:.2f}".format(i, amount))
```

If you run the program, you will see this output.

```
Year 1 amount is $ 2.24
Year 2 amount is $ 4.48
Year 3 amount is $ 8.96
Year 4 amount is $ 17.92
Year 5 amount is $ 35.84
Year 6 amount is $ 71.68
Year 7 amount is $ 143.36
```

```
Year 8 amount is $ 286.72
Year 9 amount is $ 573.44
Year 10 amount is $ 1146.88
```

Well, it would be nice if we could print the output so that all of the figures are in neat columns. We can easily do this! We just need to specify the maximum "field width" for the numbers we wish to print. We want to tell the program that we want the year number to fit a size of up to 2 digits. And we want the monetary amount, which is a real number, to fit a field width of 7 characters, with 2 digits to the right of the decimal point.

Here is how these get specified. In Java, the print statement would change from

```
System.out.printf("Year %d amount is $ %.2f\n", i, amount);
```

To this:

```
System.out.printf("Year %2d amount is $ %7.2f\n", i, amount);
```

And in Python, the string inside the print statement would change from

```
print("Year {0:d} amount is $ {1:.2f}".format(i, amount))
```

To this:

```
print("Year {0:2d} amount is $ {1:7.2f}".format(i, amount))
```

Here is your exercise….

Write a program that will print the multiples of 50 from 50 up to 1000, inclusive. Beside each of these 20 values, print a space, and then the square root in such a way that we have room for 3 characters before the decimal point and 3 digits after. To illustrate, the first two lines of your output should look like this

```
  50   7.071
 100  10.000
```

<u>Adding File output</u>

In addition to printing to the screen, it is also essential to know how to print to a file. One reason why is that sometimes we have so much output that we cannot see it all at once. Or maybe the output of one program needs to be fed as input into another program later. Let's modify the program you just worked on to also print its output to a file. The program will create a file, write its output into that file, and then close the file. As an example, I will keep referring to the Demo program from above.

Java version: The following edits would be necessary.

1. There are 2 classes that you need to import: `java.io.PrintStream` and `java.io.FileNotFoundException`.

2. Tell Java to create an output file. For this example, I will create a file called output-java.txt, but you can call your output file anything you like, as long as it does not already exist. At the start of the main program, you should have a line of code that goes like this.

```
PrintStream out = new PrintStream("output-java.txt");
```

3. Because the file I/O process could result in some unexpected error, we need to tell Java to plan for a possible FileNotFoundException. You could create a try/catch block to handle this. However, since errors of this nature are so unusual, it is sufficient to just type `throws FileNotFoundException` at the end of the line that declares your main method.

4. Now we are ready to insert the statement(s) to do the file output. This is actually very easy. The statement to print to a file is almost exactly the same as the way we print to the screen. Instead of starting with `System.out`, you just need `out`. In other words, it's the same statement, with the "`System`." removed.

5. Remember to close the file at the end of your program. We do this by saying: `out.close();`

Make the necessary changes to your Demo and square root programs so that they also print output to a file. Be sure that each program writes to a different output file name, so that you don't erase earlier output!

Python version: Here is out to add file output to the above demo.py program.

1. Near the beginning of the program, we need to create a variable to refer to our output file.

```
outFile = open("output-python.txt", "w")
```

   Notice that the way we open a file for writing is the same as how we open a file for reading, except for the 2nd argument to open, which is "w" instead of "r". What do these letters mean? ☺

2. When you are ready to actually perform the output, the statement for doing so is almost exactly the same as for screen output. The difference is that instead of making a call to a function called print(), you will make a call to the function write(), which belongs to the file object. So, you would say: outFile.write(<the string you want to print>)

3. At the end of the program, you should close the file: `outFile.close()`

Based on the above changes to my Python program, here is what my output file looks like.

Year 1 amount is $ 2.24Year 2 amount is $ 4.48Year 3 amount is $ 8.96Year 4 amount is $ 17.92Year 5 amount is $ 35.84Year 6 amount is $ 71.68Year 7 amount is $ 143.36Year 8 amount is $ 286.72Year 9 amount is $ 573.44Year 10 amount is $ 1146.88

Evidently, in step #2 above I forgot to do something. What was it?

Fix the demo.py program so that the output file looks like the standard output on the screen. Finally, modify your square root Python program so that it also writes its output to a file.