What Constitutes a Good Report?

(Adapted from James Michaud)

In general, here's a list of what we look out for in the reports.

- The report should be written in clear and concise English, free of superfluous content.
  - Superfluous content meaning those filler sentences we all used in papers in middle school. There's no need for that here, and there isn't a length requirement to justify it. Be concise.
  - The report is what shows us that your solution to the project is your own work. A student that put in 30 hours on the project should absolutely be able to explain what they worked on without being vague or resorting to filler sentences.
- The **Completion Time** section must include the number of hours you spent on the project.
- The **Difficulty Level** section must include your rating of the difficulty level (1: very easy; 5: very difficult) of the project.
- The **Help Received** section may save you if your solution looks a bit too similar to that of another student. If we see that you acknowledged help from that student, it could mean the difference between getting called for an explanation or simply receiving your grade with no fuss.
- The **Approach** section is often tricky. We don't need you to re-state the directions from the project writeup, and we don't need a recap of each step you took in solving the project.
  - Reading the writeup or textbook isn't really an approach (even though most students put that in this section). Reading these things is part of taking the class, and any student **not** doing these things is not being a good student.
  - If you're stumped on what to put here, keep it short and focus on the Issues and Resolution section instead. Shorter is infinitely better than filling it with things that all students should be doing.
  - That said, the Approach section is a great place to show us how well you understood the problem and the directions from the writeup. If you didn't use the directions, tell us about the steps you followed to solve the problem.
- The **Issues and Resolution** section is by far the most important part of the report.
  - Having difficulties completing the project shows us that you put in the work. Having difficulties completing the project also gives you something to talk about in the report. However, you must also demonstrate what you learned from those difficulties. It's also best to be able to specify what was going wrong with your code before the difficulty was resolved.
  - Be concise here as well, but more than that be precise! That means giving specific details of what you had difficulty with. A good format to follow is "I had difficulty with __. It was causing __. I was doing __, but found out I should have done __. After fixing it, __."
  - Saying something to the effect of "I didn't understand XYZ, but then looked it up, and that fixed my problem" is no good. That's not an issue or a resolution. That's doing your due diligence in getting a project started.
- The **Comments** section is completely up to you. If you want to mention anything about the exercises, or if you have some comments regarding the problems that doesn't fit into "approach" or "issues", then put it here. A report with very brief approach and issues sections may be redeemed in the comments, depending on what's written there.
  - There's not much guidance we can give in this section, as it's a bit more open-ended. However, if you find that your report is very short and that you didn't struggle at all with the project, try to embellish a bit here. As was mentioned previously, the report should show us that you understand the project and the code that solves it.

In addition, your report **must** use the given template, **must not** contain lines that exceed 80 characters, and **must not** contain spelling mistakes.

Sample Project

Problem 1. (*Day of the Week*) Write a program called `day_of_week.py` that accepts m (int), d (int), and y (int) as command-line arguments, representing a date, and writes the day of the week (0 for Sunday, 1 for Monday, and so on) *dow* to standard output, computed as

$$\begin{aligned} y_0 &= y - (14 - m)/12, \\ x_0 &= y_0 + y_0/4 - y_0/100 + y_0/400, \\ m_0 &= m + 12 \times ((14 - m)/12) - 2, \\ dow &= (d + x_0 + 31 \times m_0/12) \bmod 7. \end{aligned}$$

```
>_ ~/workspace/project1
$ python3 day_of_week.py 3 14 1879
5
$ python3 day_of_week.py 2 12 1809
0
```

Directions:

- Accept three integers m, d, and y as command-line arguments.
- Compute and write the value of day of the week *dow*.
- Use `//` (floored division) for `/` and `%` for mod.

Sample Report

```
1. Enter the number of hours it took you to complete the project between
   the <<< and >>> signs below (eg, <<<10>>>).

   <<<0.5>>>

2. Enter the difficulty level (1: very easy; 5: very difficult) of the project
   between the <<< and >>> signs below (eg, <<<3>>>).

   <<<2>>>

3. Did you receive help from anyone? List their names, status (classmate,
   CS110 grad, TA, other), and the nature of help received.

   Name               Status       Help Received
   ----               ------       -------------

   Jane Doe           Classmate    Suggested a fix for a TypeError.

4. Provide a short description of how you approached each problem, issues you
   encountered, and how you resolved those issues.

   Problem 1 (Day of the Week)
      Approach: Read in three command-line arguments m, d, and y. Used the
      given formulas to compute the corresponding day of the week,
      dow. Used the function stdio.writeln() to write the value of dow
      to standard output.
```

Issues and resolution: When I first ran my program, I received a TypeError at the following line:

```
y0 = y - (14 - m) / 12
```

I couldn't figure out what was wrong with my code until a classmate pointed out that the command-line arguments m, d, and y must be converted to ints before they are used in the formulas. I looked through the lecture slides on Basic Data Types and found out how that's done using the built-in int() function. Once I made the relevant changes, my program would run but wouldn't produce the correct result. I read through the directions for the problem carefully and realized that I was using floating-point division ('/') operator instead of the floored division ('//') operator. I fixed the issue and my program ran like a charm.

5. List any other comments here. Feel free to provide any feedback on how much you learned from doing the assignment, and whether you enjoyed doing it.

   The project was fairly straightforward, but I learned a lot by doing it, especially from the mistakes I made along the way. It was quite satisfying when I eventually got my code to work and produce the expected results. I imagine the future projects to be much more difficult, but I am totally up for the challenge.