Good Programming Practices

Coding

Consistently applying a well-defined coding standard and proper coding technique will create high quality code that is more easily maintains especially when working on a software project with a team of programmers.

a) **Variables and Methods**

- Variables and methods should be used for one and only one purpose. Do not create multi-purpose methods that perform a variety of unrelated functions.
- Keep the number of lines of code in a method to a manageable size.
- Parameters for a method should be limited to the data that must be passed from the calling routine to the method in order for the method to perform its task.
- Define variables locally where they are used.
- All instance variables of a class should be private.

b) Format

- Establish a standard size indent, e.g. three spaces, and use it consistently.
- Align the opening and closing braces if the opening brace is on a separate line. Align the closing brace with the first character at the beginning of the block of code.
- Align *else* with the corresponding *if*.
- Use comments to explain more complex loops or conditional statements.
- Avoid comments that exceed the line size and wrap around on the next line.
- Use named constants instead of specific values whenever possible, e.g. DAYS_IN_WEEK instead of 7.

c) Logic

- An *if* statement does not necessarily require an *else* statement.
- The same statement(s) should not appear in both the *if* and *else* portion of a conditional statement.
- Never execute a condition unnecessarily.
- Keep the logic as straight forward as possible avoiding unnecessary conditions or statements.
- Do not exit from a loop through a *break* or *return* statement. Set up the condition properly.
- When calling a method, store the result returned by that method if the value will be needed again.
- Specifically initialize all variable that must have a given starting value even if the programming language used automatically initializes variables according to their type.
- Fix any logic error in your code by taking the time to understand where and how the problem occurred and devising a solution based on it.