# CODING STANDARDS

## Coding Standards :

Coding standards are a set of rules, techniques, and best practices to create cleaner, more readable, more efficient code with minimal errors.

They offer a uniform format by which software engineers can use to build sophisticated and highly functional code.

## Coding Standards & Best Practices To Follow

- Choose industry-specific coding standards

  Coding best practices and standards vary depending on the industry a specific product is being built for. The standards required for coding software for luxury automobiles will differ from those for coding software for gaming.

  For example, MISRA C and C++ were written for the automotive industry and are considered the de-facto standards for building applications that emphasize safety. Currently, they are regarded as the absolute best practices for writing code in the industry.

  Adhering to industry-specific standards makes it easier to write accurate code that matches product expectations. It becomes easier to write code that will satisfy the end-users and meet business requirements.

- Focus on code readability

Readable code is easy to follow, optimizes space and time. Below are a few ways to achieve that:

- ✓ Write as few lines as possible.
- ✓ Use appropriate naming conventions.
- ✓ Segment blocks of code in the same section into paragraphs.
- ✓ Use indentation to marks the beginning and end of control structures. Clearly specify the code between them.
  - Don’t use lengthy functions. Ideally, a single function should carry out a single task.
- ✓ Use the DRY (Don’t Repeat Yourself) principle. Automate repetitive tasks whenever necessary. The same piece of code should not be repeated in the script.
- ✓ Avoid Deep Nesting. Too many nesting levels make code harder to read and follow.

- ✓ Capitalize SQL special words and function names to distinguish them from table and column names.
- ✓ Avoid long lines. It is easier for humans to read blocks of lines that are horizontally short and vertically long.
- ✓ **Standardize headers for different modules**

It is easier to understand and maintain code when the headers of different modules align with a singular format. For example, each header should contain:

- ✓ Module Name
- ✓ Date of creation
- ✓ Name of creator of module
- ✓ History of modification
- ✓ Summary of what the module does
- ✓ Functions in that module
- ✓ Variables accessed by the module
- ✓ **Don't use a single identifier for multiple purposes**

Ascribe a name to each variable that clearly describes its purpose. A single variable can't be assigned multiple values or used for numerous functions. This would confusing while reading the code and would make future enhancements more difficult to implement. Always assign unique variable names.

- ✓ Turn daily backups into an instinct

Multiple events can trigger data loss – system crash, dead battery, software glitch, hardware damage, etc. To prevent this, save code daily, and after every modification. Back up the workflow on TFS, SVN, or any other version control mechanism.

- ✓ Leave comments and prioritize documentation

Comment the code function at various points in the script. Ensure that the comments provide the guidance.

- ✓ Formalize Exception Handling

'Exception' refers to problems, issues, or uncommon events that occur when code is run and disrupt the normal flow of execution. This either pauses or terminates program execution, which is a scenario that must be avoided.

When the Exceptions occur, use the following techniques to minimize damage to overall execution in terms of both time and dev effort:

- ✓ Keep the code in a try-catch block.
- ✓ Ensure that auto recovery has been activated and can be used.
- ✓ Consider that it might be an issue of software/network slowness. Wait a few seconds for the required elements to show up.
- ✓ Use real-time log analysis.

---