1. best pratices

* **Clean Code**:
  + Code should be easy to read, understand, and maintain[**1**](https://www.freecodecamp.org/news/how-to-write-clean-code/). It should be simple, concise, and expressive, following a set of conventions, standards, and practices that make it easy to read and follow[**1**](https://www.freecodecamp.org/news/how-to-write-clean-code/).
* **Consistent Formatting**:
  + Good documentation, consistent formatting, and a well-organized codebase are all indicators of clean code[**1**](https://www.freecodecamp.org/news/how-to-write-clean-code/).
* **Code Reviews**:
  + Code reviews can help to identify potential issues and ensure that code follows best practices and conventions[**1**](https://www.freecodecamp.org/news/how-to-write-clean-code/).
* **Testing**:
  + Testing is an important aspect of clean code. It helps to ensure that code is functioning as expected and can catch errors early[**1**](https://www.freecodecamp.org/news/how-to-write-clean-code/).
* **Industry-Specific Coding Standards**:
  + Coding best practices and standards vary depending on the industry a specific product is being built for[**2**](https://www.browserstack.com/guide/coding-standards-best-practices).
* **Code Readability**:
  + Focus on making your code as readable as possible[**2**](https://www.browserstack.com/guide/coding-standards-best-practices).
* **Standardize Headers for Different Modules**:
  + This helps in understanding the purpose of different parts of your code[**2**](https://www.browserstack.com/guide/coding-standards-best-practices).
  + **Don’t use a Single Identifier for multiple purposes**:
    - This can lead to confusion and make the code harder to understand[**2**](https://www.browserstack.com/guide/coding-standards-best-practices).
* **Leave Comments and Prioritize Documentation**:
  + This helps other developers understand your code better[**2**](https://www.browserstack.com/guide/coding-standards-best-practices).
* **Try to formalize Exception Handling**:
  + This helps in dealing with unexpected situations in your code[**2**](https://www.browserstack.com/guide/coding-standards-best-practices).

Source(s)

1. [How to Write Clean Code – Tips and Best Practices (Full Handbook)](https://www.freecodecamp.org/news/how-to-write-clean-code/)

2. [Coding Standards and Best Practices to Follow | BrowserStack](https://www.browserstack.com/guide/coding-standards-best-practices)

3. [Best Practices for Writing Clean and Maintainable Code - DZone](https://dzone.com/articles/best-practices-for-writing-clean-and-maintainable-code)

4. [How to Write Good Code: 10 Beginner-friendly Techniques for Instant ...](https://www.sitepoint.com/how-to-write-good-code/)

1. empty lines

* **Improves Readability**:
  + Empty lines can make the code more readable by visually separating logical blocks of code[**1**](https://stackoverflow.com/questions/5661444/when-should-we-insert-blank-lines-in-source-code).
* **Logical Separation**:
  + They can be used to separate functions, classes, and other code elements[**1**](https://stackoverflow.com/questions/5661444/when-should-we-insert-blank-lines-in-source-code).
* **Reduces Noise in Version Control**:
  + Having an empty line at the end of a file can reduce unnecessary changes in source control systems[**3**](https://stackoverflow.com/questions/2287967/why-is-it-recommended-to-have-empty-line-in-the-end-of-a-source-file).
* **Compatibility with Tools**:
  + Some tools may misbehave if the last line of data in a text file is not terminated with a newline[**2**](https://stackoverflow.com/questions/2402593/whats-the-reason-for-leaving-an-extra-blank-line-at-the-end-of-a-code-file).
* **Concatenation**:
  + If you try to concatenate two text files together, you will be much happier if the first one ends with a newline character[**2**](https://stackoverflow.com/questions/2402593/whats-the-reason-for-leaving-an-extra-blank-line-at-the-end-of-a-code-file).
* Empty lines can vary by programming language
  + For example, in Python, the interpreter uses blank lines to detect the end of blocks of code[**5**](https://stackoverflow.com/questions/50987645/whats-wrong-to-have-empty-lines-in-java-class). In Java, empty lines do not matter at runtime, as the Java compiler turns your source code into bytecode[**4**](https://stackoverflow.com/questions/60143061/meaning-of-an-empty-line-in-python-source-code-file).

Source(s)

1. [When should we insert blank line (s) in source code?](https://stackoverflow.com/questions/5661444/when-should-we-insert-blank-lines-in-source-code)

2. [What's the reason for leaving an extra blank line at the end of a code ...](https://stackoverflow.com/questions/2402593/whats-the-reason-for-leaving-an-extra-blank-line-at-the-end-of-a-code-file)

3. [Why is it recommended to have empty line in the end of a source file?](https://stackoverflow.com/questions/2287967/why-is-it-recommended-to-have-empty-line-in-the-end-of-a-source-file)

4. [Meaning of an empty line in Python source code file](https://stackoverflow.com/questions/60143061/meaning-of-an-empty-line-in-python-source-code-file)

5. [What's wrong to have empty lines in Java class? [closed]](https://stackoverflow.com/questions/50987645/whats-wrong-to-have-empty-lines-in-java-class)

1. every instruction in a different line

* **Readability**:
  + Placing each instruction on a separate line makes the code easier to read[**1**](https://www.turing.com/blog/source-code-management-tools-best-practices/).
  + It allows developers to quickly scan the code and understand the flow of logic[**1**](https://www.turing.com/blog/source-code-management-tools-best-practices/).
* **Debugging**:
  + It's easier to identify errors when each instruction is on its own line[**1**](https://www.turing.com/blog/source-code-management-tools-best-practices/).
  + Many debugging tools point to the line number where an error occurred[**1**](https://www.turing.com/blog/source-code-management-tools-best-practices/).
* **Version Control**:
  + When changes are made, version control systems like Git show differences line by line[**1**](https://www.turing.com/blog/source-code-management-tools-best-practices/).
  + If multiple instructions are on the same line, a change in one instruction will show the whole line as changed[**1**](https://www.turing.com/blog/source-code-management-tools-best-practices/).
* **Code Reviews**:
  + Code reviews are simpler when each instruction is on a separate line[**1**](https://www.turing.com/blog/source-code-management-tools-best-practices/).
  + Reviewers can provide feedback for specific lines, making the review process more efficient[**1**](https://www.turing.com/blog/source-code-management-tools-best-practices/).
* Depend on the programming language.
  + Some languages or style guides may have different conventions[**2**](https://www.freecodecamp.org/news/how-to-write-clean-code/).

Source(s)

1. [Source Code Management, Tools, and Best Practices in 2023](https://www.turing.com/blog/source-code-management-tools-best-practices/)

2. [How to Write Clean Code – Tips and Best Practices (Full Handbook)](https://www.freecodecamp.org/news/how-to-write-clean-code/)

3. [Source Code Documentation Best Practices | Eastern Peak](https://easternpeak.com/blog/source-code-documentation-best-practices/)

4. [Coding Standards and Best Practices to Follow | BrowserStack](https://www.browserstack.com/guide/coding-standards-best-practices)

5. [Documentation Best Practices | styleguide](https://google.github.io/styleguide/docguide/best_practices.html)

1. variable names

* **Meaningful Names**:
  + Choose names that accurately describe the entity the variable represents[**1**](https://programmingduck.com/articles/naming)[**2**](https://www.brainstobytes.com/writing-good-variable-names/).
  + For example, a variable that holds a collection of users could be named users[**1**](https://programmingduck.com/articles/naming).
* **Domain Knowledge**:
  + Embed meaningful domain knowledge into the name. It should be clear what entity from the domain a variable represents[**2**](https://www.brainstobytes.com/writing-good-variable-names/).
* **Use Software Knowledge**:
  + Words for programming concepts let you express complex ideas in a few words that are easily understandable by fellow programmers[**2**](https://www.brainstobytes.com/writing-good-variable-names/).
* **Avoid Type Information**:
  + Don't put type information in the name. Modern statically typed languages made names like stringName obsolete[**2**](https://www.brainstobytes.com/writing-good-variable-names/).
* **Avoid Useless Context**:
  + Don't put into a variable name information like the class, package, or module it belongs to[**2**](https://www.brainstobytes.com/writing-good-variable-names/).
* **Length**:
  + Use short enough and long enough variable names in each scope of code[**4**](https://www.bbc.co.uk/bitesize/guides/zwmbgk7/revision/3).
  + Generally, length may be 1 char for loop counters, 1 word for condition/loop variables, 1-2 words for methods, 2-3 words for classes, 3-4 words for globals[**4**](https://www.bbc.co.uk/bitesize/guides/zwmbgk7/revision/3).
* **Specific Names**:
  + Use specific names for variables, for example, "value", "equals", "data", are not valid names for any case[**4**](https://www.bbc.co.uk/bitesize/guides/zwmbgk7/revision/3).
* **Consistency**:
  + Variable names should be consistent.
  + For example, 'name' is not the same as 'Name' or 'NAME'[**5**](https://medium.com/swlh/how-to-better-name-your-functions-and-variables-e962a4ef335b).

Source(s)

1. [Naming (in code) - The ultimate guide and reference - Programming Duck](https://programmingduck.com/articles/naming)

2. [Writing good variable names](https://www.brainstobytes.com/writing-good-variable-names/)

3. [Best Practices for Variable and Method Naming - DZone](https://dzone.com/articles/best-practices-variable-and)

4. [Naming variables - Programming basics - KS3 Computer Science ... - BBC](https://www.bbc.co.uk/bitesize/guides/zwmbgk7/revision/3)

5. [How to Better Name Your Functions and Variables | The Startup - Medium](https://medium.com/swlh/how-to-better-name-your-functions-and-variables-e962a4ef335b)

1. return value of a function

* **Explicit Return Statements**:
  + Use explicit return statements in your functions. This makes it clear what value the function is intended to return[**2**](https://realpython.com/lessons/python-return-conditionals/).
* **Returning Single or Multiple Values**:
  + Depending on the requirements of your function, you can return a single value or multiple values[**2**](https://realpython.com/lessons/python-return-conditionals/).
* **Returning None Explicitly**:
  + If your function doesn't need to return a value, you can return None explicitly to make it clear that the function is not intended to return a value[**2**](https://realpython.com/lessons/python-return-conditionals/).
* **Avoiding Complex Expressions**:
  + Try to avoid returning complex expressions directly. Instead, consider breaking the expression down into smaller parts and return a simple expression[**2**](https://realpython.com/lessons/python-return-conditionals/).
* **Returning Values vs Modifying Globals**:
  + It's generally better to return values from a function rather than modifying global variables[**2**](https://realpython.com/lessons/python-return-conditionals/).
* **Using Return with Conditionals**:
  + You can use return statements with conditional statements. This can be useful for returning different values based on certain conditions[**3**](https://stackoverflow.com/questions/42575331/best-practice-for-compute-the-function-return-value).
* **Returning True or False**:
  + If your function is intended to check a condition, consider returning True or False[**2**](https://realpython.com/lessons/python-return-conditionals/).
* **Short-Circuiting Loops**:
  + You can use return statements to short-circuit loops. This can be useful for stopping the execution of a loop as soon as a certain condition is met[**2**](https://realpython.com/lessons/python-return-conditionals/).
* **Recognizing Dead Code**:
  + Be aware of "dead code", or code that can never be executed. If you have a return statement in a loop, any code after that return statement will not be executed[**2**](https://realpython.com/lessons/python-return-conditionals/).
* **Returning Multiple Named-Objects**:
  + If your function needs to return multiple values, consider returning a named object or a data structure like a dictionary or a tuple[**2**](https://realpython.com/lessons/python-return-conditionals/).

Source(s)

1. [The Python return Statement: Usage and Best Practices](https://realpython.com/python-return-statement/)

2. [Using return Statements With Conditionals – Real Python](https://realpython.com/lessons/python-return-conditionals/)

3. [Best practice for compute the function return value](https://stackoverflow.com/questions/42575331/best-practice-for-compute-the-function-return-value)

4. [PowerShell function return best practices - Stack Overflow](https://stackoverflow.com/questions/34251843/powershell-function-return-best-practices)

5. [Golang - best practices to pass and return variables](https://stackoverflow.com/questions/40822745/golang-best-practices-to-pass-and-return-variables)

1. declare variables in the beginning

* The best practice for declaring variables can depend on the programming language and the specific coding style guide you're following. However, here are some general guidelines:
* **Scope**: It is good practice to restrict the scope of your variables to the minimum needed[**4**](https://www.geeksforgeeks.org/variables-in-c/). A loop counter is only needed in a loop so declare it at the top of a loop[**4**](https://www.geeksforgeeks.org/variables-in-c/). A variable used in a whole function, declare at the top of the function[**4**](https://www.geeksforgeeks.org/variables-in-c/). A variable only used in a small block, declare it at the top of the block[**4**](https://www.geeksforgeeks.org/variables-in-c/).
* **Declare When Needed**: In many modern programming languages, it's often recommended to declare variables as close as possible to where they are first used[**1**](https://cboard.cprogramming.com/c-programming/172041-variable-declaration-what-considered-good-practice.html)[**2**](https://stackoverflow.com/questions/7959573/declaring-variables-inside-loops-good-practice-or-bad-practice). This can make the code easier to understand and maintain[**1**](https://cboard.cprogramming.com/c-programming/172041-variable-declaration-what-considered-good-practice.html)[**2**](https://stackoverflow.com/questions/7959573/declaring-variables-inside-loops-good-practice-or-bad-practice).
* **Inside Loops**: Declaring variables inside loops is generally considered good practice[**1**](https://cboard.cprogramming.com/c-programming/172041-variable-declaration-what-considered-good-practice.html). By creating variables inside loops, you ensure their scope is restricted to inside the loop[**1**](https://cboard.cprogramming.com/c-programming/172041-variable-declaration-what-considered-good-practice.html). It cannot be referenced nor called outside of the loop[**1**](https://cboard.cprogramming.com/c-programming/172041-variable-declaration-what-considered-good-practice.html).
* **Initialization**: It's also a good practice to initialize variables when you declare them, if possible[**2**](https://stackoverflow.com/questions/7959573/declaring-variables-inside-loops-good-practice-or-bad-practice). This can help prevent bugs related to uninitialized variables[**2**](https://stackoverflow.com/questions/7959573/declaring-variables-inside-loops-good-practice-or-bad-practice).
* Remember, these are general guidelines and the specifics may vary based on the programming language and the project you are working on.

Source(s)

1. [Variable declaration - what is considered good practice? - C++ Programming](https://cboard.cprogramming.com/c-programming/172041-variable-declaration-what-considered-good-practice.html)

2. [Declaring variables inside loops, good practice or bad practice?](https://stackoverflow.com/questions/7959573/declaring-variables-inside-loops-good-practice-or-bad-practice)

3. [Best Practice when Declaring and Initializing variables in c++](https://stackoverflow.com/questions/42302857/best-practice-when-declaring-and-initializing-variables-in-c)

4. [C Variables - GeeksforGeeks](https://www.geeksforgeeks.org/variables-in-c/)

5. [Java variable declaration best practices - Online Tutorials Library](https://www.tutorialspoint.com/Java-variable-declaration-best-practices)

1. empty lines in the begin and end

* **Beginning of a Function**:
  + It's generally not common to start a function with an empty line[**1**](https://stackoverflow.com/questions/1998656/empty-lines-in-functions-methods)[**2**](https://stackoverflow.com/questions/70132648/eslint-disallow-empty-lines-inside-function-body).
  + The first line of a function is typically where you start writing your code[**1**](https://stackoverflow.com/questions/1998656/empty-lines-in-functions-methods)[**2**](https://stackoverflow.com/questions/70132648/eslint-disallow-empty-lines-inside-function-body).
* **End of a Function**:
  + Similarly, it's not common to end a function with an empty line[**1**](https://stackoverflow.com/questions/1998656/empty-lines-in-functions-methods)[**2**](https://stackoverflow.com/questions/70132648/eslint-disallow-empty-lines-inside-function-body).
  + The last line of a function is typically where you return a value or end the function[**1**](https://stackoverflow.com/questions/1998656/empty-lines-in-functions-methods)[**2**](https://stackoverflow.com/questions/70132648/eslint-disallow-empty-lines-inside-function-body).
* **Between Functions**:
  + It's common to put empty lines between functions to separate them visually[**3**](https://peps.python.org/pep-0008/).
  + This can make the code easier to read and understand[**3**](https://peps.python.org/pep-0008/).
* **Inside a Function**:
  + Some developers use empty lines inside a function to separate logical blocks of code[**1**](https://stackoverflow.com/questions/1998656/empty-lines-in-functions-methods)[**2**](https://stackoverflow.com/questions/70132648/eslint-disallow-empty-lines-inside-function-body).
  + However, others argue that if a function needs to be separated into different sections with empty lines, it might be doing too much and could be refactored into smaller functions[**1**](https://stackoverflow.com/questions/1998656/empty-lines-in-functions-methods)[**2**](https://stackoverflow.com/questions/70132648/eslint-disallow-empty-lines-inside-function-body).

Source(s)

1. [coding style - empty lines in functions/methods - Stack Overflow](https://stackoverflow.com/questions/1998656/empty-lines-in-functions-methods)

2. [ESLint: disallow empty lines inside function body](https://stackoverflow.com/questions/70132648/eslint-disallow-empty-lines-inside-function-body)

3. [PEP 8 – Style Guide for Python Code | peps.python.org](https://peps.python.org/pep-0008/)

4. [Using clang-format - keep empty braces on the same line](https://stackoverflow.com/questions/39858266/using-clang-format-keep-empty-braces-on-the-same-line)