1. best pratices

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

Source(s)

1. How to Write Clean Code – Tips and Best Practices (Full Handbook)
2. Coding Standards and Best Practices to Follow | BrowserStack
3. Best Practices for Writing Clean and Maintainable Code - DZone
4. How to Write Good Code: 10 Beginner-friendly Techniques for Instant ...

2. empty lines

- **Improves Readability**:
    - Empty lines can make the code more readable by visually separating logical blocks of code[1].
- **Logical Separation**:
    - They can be used to separate functions, classes, and other code elements[1].
- **Reduces Noise in Version Control**:
    - Having an empty line at the end of a file can reduce unnecessary changes in source control systems[3].
- **Compatibility with Tools**:
    - Some tools may misbehave if the last line of data in a text file is not terminated with a newline[2].
- **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].
- 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]. In Java, empty lines do not matter at runtime, as the Java compiler turns your source code into bytecode[4].

Source(s)

1. When should we insert blank line (s) in source code?
2. What's the reason for leaving an extra blank line at the end of a code ...
3. 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
5. What's wrong to have empty lines in Java class? [closed]

3. every instruction in a different line

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

Source(s)

1. Source Code Management, Tools, and Best Practices in 2023
2. How to Write Clean Code – Tips and Best Practices (Full Handbook)
3. Source Code Documentation Best Practices | Eastern Peak
4. Coding Standards and Best Practices to Follow | BrowserStack
5. Documentation Best Practices | styleguide

4. variable names

- **Meaningful Names**:
    - Choose names that accurately describe the entity the variable represents[1][2].
    - For example, a variable that holds a collection of users could be named `users`[1].
- **Domain Knowledge**:
    - Embed meaningful domain knowledge into the name. It should be clear what entity from the domain a variable represents[2].
- **Use Software Knowledge**:
    - Words for programming concepts let you express complex ideas in a few words that are easily understandable by fellow programmers[2].
- **Avoid Type Information**:
    - Don't put type information in the name. Modern statically typed languages made names like `stringName` obsolete[2].
- **Avoid Useless Context**:
    - Don't put into a variable name information like the class, package, or module it belongs to[2].
- **Length**:
    - Use short enough and long enough variable names in each scope of code[4].
    - 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].
- **Specific Names**:
    - Use specific names for variables, for example, "value", "equals", "data", are not valid names for any case[4].
- **Consistency**:
    - Variable names should be consistent.
    - For example, 'name' is not the same as 'Name' or 'NAME'[5].

Source(s)

1. Naming (in code) - The ultimate guide and reference - Programming Duck
2. Writing good variable names
3. Best Practices for Variable and Method Naming - DZone
4. Naming variables - Programming basics - KS3 Computer Science ... - BBC
5. How to Better Name Your Functions and Variables | The Startup - Medium

5. 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].
- Returning Single or Multiple Values:
    - Depending on the requirements of your function, you can return a single value or multiple values[2].
- 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].
- 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].
- Returning Values vs Modifying Globals:
    - It's generally better to return values from a function rather than modifying global variables[2].
- 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].
- Returning True or False:
    - If your function is intended to check a condition, consider returning True or False[2].
- 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].
- 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].
- 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].

Source(s)

1. The Python return Statement: Usage and Best Practices
2. Using return Statements With Conditionals – Real Python
3. Best practice for compute the function return value
4. PowerShell function return best practices - Stack Overflow
5. Golang - best practices to pass and return variables

6. 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]. A loop counter is only needed in a loop so declare it at the top of a loop[4]. A variable used in a whole function, declare at the top of the function[4]. A variable only used in a small block, declare it at the top of the block[4].
- 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[12]. This can make the code easier to understand and maintain[12].
- Inside Loops: Declaring variables inside loops is generally considered good practice[1]. By creating variables inside loops, you ensure their scope is restricted to inside the loop[1]. It cannot be referenced nor called outside of the loop[1].
- Initialization: It's also a good practice to initialize variables when you declare them, if possible[2]. This can help prevent bugs related to uninitialized variables[2].
- 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
2. Declaring variables inside loops, good practice or bad practice?
3. Best Practice when Declaring and Initializing variables in c++
4. C Variables - GeeksforGeeks
5. Java variable declaration best practices - Online Tutorials Library

7. 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][2].
    - The first line of a function is typically where you start writing your code[1][2].
- End of a Function:
    - Similarly, it's not common to end a function with an empty line[1][2].
    - The last line of a function is typically where you return a value or end the function[1][2].
- Between Functions:
    - It's common to put empty lines between functions to separate them visually[3].
    - This can make the code easier to read and understand[3].
- Inside a Function:
    - Some developers use empty lines inside a function to separate logical blocks of code[1][2].
    - 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][2].

Source(s)

1. coding style - empty lines in functions/methods - Stack Overflow
2. ESLint: disallow empty lines inside function body
3. PEP 8 – Style Guide for Python Code | peps.python.org
4. Using clang-format - keep empty braces on the same line