Strict Rules (must be followed for your code to run):

1. Allowed Characters: Variable names can only contain:
   - Letters (a-z, A-Z)
   - Numbers (0-9)
   - Underscores (_)
1. Cannot Start with a Digit: A variable name cannot begin with a number.
   - 1variable is invalid
   - variable1 is valid
2. Case-Sensitive: Python variable names are case-sensitive. This means:
   - MyVar, myvar and MyVar are all treated as different variables.
3. No Spaces: Variable names cannot contain spaces.
   - my variable is invalid
   - my_variable is valid
4. No Keywords: You cannot use Python's reserved keywords (words that have special meaning in Python) as variable names. Examples include:

if	else
for	while
def	class
True	False
None	yield

- There are many more. If you try to use a keyword, Python will give you a SyntaxError.)

Conventions/Best Practices (highly recommended for good, readable code):

1. Descriptive Names: Choose names that clearly indicate the purpose or content of the variable.
   - Instead of x = 10, use age = 10 or user_count = 10.
2. Snake Case for Multi-Word Names: For variable names consisting of multiple words, use underscores to separate them. This is the official Python convention (PEP 8).
   - my_variable_name (snake case, recommended)
   - Avoid myVariableName (camelCase, common in some other languages) or MyVariableName (PascalCase, typically used for class names in Python).
3. Avoid Single Letters (unless for specific cases): Generally, avoid single-letter variable names unless they are commonly understood loop counters (i, j, k) or represent a clear mathematical variable in a short context.
4. Constants: For variables that are intended to be constant (their value shouldn't change during program execution), use all uppercase letters with underscores.
   - MAX_CONNECTIONS
   - PI
5. Avoid Leading/Trailing Underscores (mostly):
   - Single leading underscore (_variable): By convention, indicates a "private" internal variable (though not strictly enforced by Python).
   - Double leading underscore (__variable): Triggers "name mangling" within classes, making the variable less directly accessible from outside the class.
   - Leading and trailing double underscores (__variable__): Reserved for special "magic methods" or attributes in Python (e.g., __init__, __str__). Avoid naming your own variables this way.