Python 3.13 Keywords:
by Atomic Badger
1-27-2025

On 1-17-2025, I asked **ChatGPT** the following:

“please provide a numbered list in alphabetical order of all keywords in the computer language python 3.13. please provide a short definition of what keyword does.”

This was the answer given, word-for-word.

“In Python 3.13, there are 35 reserved keywords that have special meanings and cannot be used as identifiers (e.g., variable names, function names). These keywords are fundamental to the language's syntax and structure. Here's an alphabetically ordered list of these keywords, along with brief descriptions:

1. **and**: Logical operator that returns True if both operands are true.
2. **as**: Used to create an alias, commonly in the context of importing modules.
3. **assert**: Used for debugging; tests if a condition is true, and if not, raises an AssertionError.
4. **async**: Declares an asynchronous function, used with await for asynchronous programming.
5. **await**: Pauses the execution of an async function until the awaited coroutine completes.
6. **break**: Exits the nearest enclosing loop prematurely.
7. **class**: Defines a new user-defined class.
8. **continue**: Skips the rest of the code inside a loop for the current iteration and moves to the next iteration.
9. **def**: Defines a new function or method.
10. **del**: Deletes objects, variables, or items from a list or dictionary.

11. **elif**: Stands for "else if"; used in conditional statements to check multiple expressions.
12. **else**: Specifies a block of code to execute if the preceding if or elif conditions are false.
13. **except**: Catches exceptions that occur in the try block.
14. **False**: Boolean value representing falsehood
15. **finally**: Specifies a block of code to execute after the try and except blocks, regardless of whether an exception was raised.
16. **for**: Begins a for loop, which iterates over a sequence.
17. **from**: Specifies the module from which to import specific attributes or functions.
18. **global**: Declares that a variable inside a function refers to a global variable.
19. **if**: Begins a conditional statement.
20. **import**: Includes modules into the current namespace.
21. **in**: Checks if a value exists within a sequence or iterable.
22. **is**: Tests for object identity; checks if two references point to the same object.
23. **lambda**: Creates an anonymous function.
24. **None**: Represents the absence of a value or a null value.
25. **nonlocal**: Declares that a variable inside a nested function refers to a variable in the nearest enclosing scope that is not global.
26. **not**: Logical operator that inverts the truth value of its operand.
27. **or**: Logical operator that returns True if at least one of the operands is true.
28. **pass**: A null statement; a placeholder in loops, functions, or classes that does nothing.
29. **raise**: Triggers an exception.

30. **return**: Exits a function and optionally passes back a value.
31. **True**: Boolean value representing truth.
32. **try**: Begins a block of code that will be tested for exceptions.
33. **while**: Begins a while loop, which continues as long as a specified condition is true.
34. **with**: Simplifies exception handling by encapsulating common preparation and cleanup tasks.
35. **yield**: Pauses a generator function and returns a value; can be resumed later.

These keywords form the core of Python's syntax and are integral to writing effective Python code."

If one intends to truly learn to write Python code, one should start with learning the keywords.