Understanding `nonlocal`

- The `nonlocal` keyword is used within nested functions to declare that a variable is not local to the inner function but resides in a closer enclosing scope (beyond the local scope).
- It allows the nested function to modify the value of the variable in the enclosing scope.

Syntax

```
nonlocal variable_name

# Rest of the function's code
```

- The `nonlocal` keyword must appear at the beginning of the inner function, before any assignments or operations on the variable.
- You can declare multiple variables as `nonlocal` in a single statement.

Example

```
def outer_function():
    count = 0

    def inner_function():
        nonlocal count  # Declare 'count' as nonlocal
        count += 1
        return count

    return inner_function

increment = outer_function()

print(increment())  # Output: 1
print(increment())  # Output: 2
```

In Summary

The `nonlocal` keyword provides a mechanism for nested functions to access and modify variables in their enclosing scopes, but it's important to use it thoughtfully and consider alternative approaches when appropriate. By following these guidelines, you can write cleaner and more maintainable Python code.

Understanding `global`

- The `global` keyword is used within functions to declare that a variable belongs to the module's global scope (the entire Python file).
- It allows the function to modify the value of the global variable, even though the variable is not defined within the function itself.

  Syntax

  ```
  global variable_name

  # Rest of the function's code
  ```
- The `global` keyword appears at the beginning of the function, before any assignments or operations on the variable.
- You can declare multiple variables as `global` in a single statement.

  Example

  ```
  count = 0  # Global variable

  def increment():
      global count  # Declare 'count' as global
      count += 1

  increment()
  print(count)  # Output: 1
  ```

In Conclusion

The `global` keyword provides a way for functions to access and modify global variables, but it's essential to use it with caution and explore alternative approaches for better code organization.