**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Topic 49 - Functions: Local vs. Global Variables**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**What**

* **Global Variables**: Defined outside of any function, they’re accessible **throughout the entire program**.
* **Local Variables**: Defined inside functions, they’re accessible **only within that function**.
* **Scope**: Determines where a variable can be accessed—**global scope** means it’s accessible everywhere, while **local scope** restricts it to within the function.

**Why**

* **Avoiding Errors**: By understanding scope, you prevent **NameError** issues when variables are inaccessible.
* **Code Clarity**: Keeping variables local to functions when possible makes code more **predictable and easier to understand**.

**How**

1. **Creating and Using Global Variables**  
   Define a variable in the main body of your code:

python

Copy code

what\_to\_say = "Hello" # Global variable

def greet():

print(what\_to\_say) # Accessible in functions

greet() # Output: Hello

1. **Creating and Using Local Variables**  
   Define variables inside functions for local scope:

python

Copy code

def say\_something():

what\_to\_say = "Hi" # Local variable

print(what\_to\_say)

say\_something() # Output: Hi

print(what\_to\_say) # Error: NameError: 'what\_to\_say' is not defined

1. **Avoiding Global Variable Use in Functions**  
   Use arguments to pass values into functions, avoiding global variables:

python

Copy code

greeting = "Hello" # Global variable

def greet(message): # Use parameters instead of global variables

print(message)

greet(greeting) # Output: Hello

1. **Local and Global Variables with Same Name**  
   When a function and the main code use the same variable name, **they’re treated as separate variables**:

python

Copy code

x = 10 # Global variable

def modify\_x():

x = 5 # Local variable in this function

print(x) # Output: 5

modify\_x()

print(x) # Output: 10 (global x remains unaffected)

**Things to Remember**

* **Global variables are accessible inside functions** but are **better avoided** inside functions to keep code modular and predictable.
* Variables with the **same name** inside and outside functions don’t interfere with each other.
* Local variables **expire** after the function call ends—they’re no longer accessible.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**