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

**Topic 50 - Functions within Functions**

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

**What**

* **Functions can call other functions**. This is useful for breaking complex tasks into simpler, reusable tasks.
* If a **function calls another function**, it must be **defined before** the function that calls it.

**Why**

* **Readability**: Breaking down code into functions that call each other makes it **easier to read and debug**.
* **Avoiding Scope Issues**: Passing arguments between functions **ensures data is shared without causing scope issues**.

**How**

1. **Basic Nested Function Example**  
   Define a function that calls another function:

python

Copy code

def now\_say\_it(message):

print(message)

def say\_something():

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

now\_say\_it(what\_to\_say) # Pass the local variable as an argument

say\_something() # Output: Hi

1. **Using Parameters and Arguments to Avoid Scope Errors**  
   When a variable is local to one function, pass it as an argument to share its value:

python

Copy code

def now\_say\_it(content): # Accepting parameter

print(content) # Output the received parameter

def say\_something():

what\_to\_say = "Hello"

now\_say\_it(what\_to\_say) # Passing as argument

say\_something() # Output: Hello

**Things to Remember**

* **Define helper functions before the main function** that calls them.
* Use **parameters** when calling functions within other functions to prevent **scope-related errors**.

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