In this post, I will explore Python functions using my own original examples. These examples are designed to illustrate the role of arguments, parameters, local variables, and scope rules. The concepts are inspired by our readings in *Think Python*.

**Example 1: Defining a Function with Parameters**

def combine\_names(first\_name, last\_name):

print(f"{first\_name} {last\_name}")

combine\_names("Merhawit", "Kahsay")

* first\_name and last\_name are **parameters** (placeholders in the function).
* "Merhawit" and "Kahsay" are **arguments** (actual values passed).

**Example 2: Calling the Function with Different Argument Types**

# Value arguments

combine\_names("Liya", "Tesfaye")

# Variable arguments

first = "Daniel"

last = "Berhe"

combine\_names(first, last)

# Expression arguments

combine\_names("Meri".upper(), "kahsay".capitalize())

* "Liya" is a value; first is a variable; "Meri".upper() is an expression.

**Example 3: Local Variable Scope**

def multiply(a, b):

c = a \* b

print(c)

multiply(2, 3)

print(c) # NameError: c is not defined

c is local. Accessing it outside the function causes a **NameError**.

**Example 4: Parameter Scope**

def greet\_user(special\_name):

print(f"Hello, {special\_name}!")

greet\_user("Merhawit")

print(special\_name) # NameError

specific\_name exists only inside greet\_user.

**Example 5: Global vs. Local Variable**

x = 10

def show\_value():

x = 5

print("Inside function, x =", x)

show\_value()

print("Outside function, x =", x)

Local and global variables with the same name do **not interfere** with each other.

**Question to My Classmates:**  
Why do you think Python doesn’t allow access to function parameters or local variables outside the function? Could this be a security feature?

Reference  
  
Downey, A. (2015). Think Python: How to think like a computer scientist. Green Tree Press. <https://greenteapress.com/thinkpython2/thinkpython2.pdf>