# >> USER: # Global Scope

player\_health = 10 # Global Scope avialable anywhere in the file

|  |
| --- |
| def game():  def drink\_potion():  potion\_strength = 2  print(potion\_strength)  print(player\_health)  drink\_potion()  print(player\_health) |

## **explain namespace scope of the code above.why is it such that drink\_potion() outside of function definition game() returns Name Error?**

>> **AI\_AISTANT :** In the given code, there are three levels of scope: global scope, function scope (game function), and nested function scope (drink\_potion function). Let's explain each scope:

## **1. Global Scope:**

- `player\_health` is defined in the global scope and has a value of 10.

- Global variables are accessible from anywhere in the file, including inside functions.

## **2. Function Scope (game function):**

- The `game` function defines a nested function called `drink\_potion`.

- `drink\_potion` is only accessible within the `game` function scope and cannot be accessed from outside the function.

## 3. **Nested Function Scope (drink\_potion function):**

- The `drink\_potion` function has its own scope, nested within the `game` function.

- It defines a local variable called `potion\_strength` with a value of 2.

- The `drink\_potion` function also prints the values of `potion\_strength` and `player\_health`.

Now, let's address why calling `drink\_potion()` outside the `game()` function results in a NameError:

When `drink\_potion()` is called outside of the `game()` function, it is not defined in the global scope. Since `drink\_potion` is only accessible within the `game` function, attempting to call it outside of the function scope will raise a NameError. The function is not visible or accessible in the global scope because it is defined within the `game` function.

|  |
| --- |
| **player\_health = 10 # Global Scope: variable defined outside any functions**  **def game():**  **def drink\_potion():**  **potion\_strength = 2 # Nested Function Scope: variable defined inside drink\_potion()**  **print(potion\_strength)**  **print(player\_health) # Accessing global variable inside nested function**  **drink\_potion() # Calling drink\_potion() from within game() function**  **print(player\_health) # Accessing global variable outside any functions** |

To resolve the NameError, you would need to call `drink\_potion()` from within the `game()` function or make it accessible from the global scope by defining it outside the `game` function.

## **Code:**

In the code above, we have a global variable **player\_health** defined in the global scope. It has a value of 10. This variable is accessible from anywhere within the file, including inside functions.

The **game()** function defines a nested function called **drink\_potion()**. The **drink\_potion()** function has its own scope, known as the nested function scope. It defines a local variable **potion\_strength** with a value of 2.

Inside the **drink\_potion()** function, we can access both the local variable **potion\_strength** and the global variable **player\_health**. The local variable **potion\_strength** is specific to the **drink\_potion()** function and is only accessible within that function.

When we call **drink\_potion()** from within the **game()** function, it executes the code inside the **drink\_potion()** function and prints the values of **potion\_strength** and **player\_health**. Since **player\_health** is a global variable, it is accessible within the nested function.

Finally, outside of any functions, we print the value of **player\_health** directly in the global scope.

Now, let's address the issue of calling **drink\_potion()** outside of the **game()** function:

|  |
| --- |
| **drink\_potion()** |

If we call **drink\_potion()** directly in the global scope, without being inside the **game()** function, it will result in a NameError. This is because **drink\_potion()** is defined within the **game()** function and is not accessible in the global scope.

## **Resolve Name Error:**

To resolve the NameError, we need to call **drink\_potion()** from within the **game()** function or make it accessible from the global scope by defining it outside the **game()** function. Here's an example:

|  |
| --- |
| **player\_health = 10**  **def drink\_potion():**  **potion\_strength = 2**  **print(potion\_strength)**  **print(player\_health)**  **drink\_potion()**  **print(player\_health)** |

In the modified code, we define **drink\_potion()** in the global scope. Now, we can call it directly without any NameError.

# **Here's another example to illustrate namespace and scope in Python:**

|  |
| --- |
| **def outer\_function():**  **outer\_variable = 10**  **def inner\_function():**  **inner\_variable = 5**  **print("Inner variable:", inner\_variable)**  **print("Outer variable:", outer\_variable)**  **inner\_function()**  **# Calling the outer function**  **outer\_function()** |

In this example, we have an **outer\_function()** that defines an **outer\_variable** within its scope. Inside the **outer\_function()**, there is an **inner\_function()** defined, which has its own scope (nested within the **outer\_function()** scope). The **inner\_function()** defines an **inner\_variable** within its own scope.

When we call **inner\_function()** from within **outer\_function()**, the code inside the **inner\_function()** is executed. It can access both the **inner\_variable** and the **outer\_variable** defined in their respective scopes.

## **Output:**

|  |
| --- |
| **Inner variable: 5**  **Outer variable: 10** |

In this example, the **inner\_function()** can access the **inner\_variable** defined within its scope and the **outer\_variable** defined in the outer scope of **outer\_function()**. However, the variables defined within the **inner\_function()** are not accessible outside of its scope.

This example demonstrates the concept of nested scopes, where inner scopes can access variables from outer scopes. It also highlights the concept of namespaces, where variables defined within a function's scope are only accessible within that function or its nested functions.

Remember, the scope determines the visibility and accessibility of variables, and the namespaces keep track of the names and objects within a particular scope.