Scope of Variable in Python

Scope of Variable - Global and Local Variables

Every variable you create in Python lives in a specific area of your program — this area is called its scope.
Understanding scope is crucial to avoid bugs and ensure proper variable access and modification.

Why We Use Variables in Python

- Store data for later use (e.g., numbers, text, lists).
- Reuse values without repeating the same data.
- Update or change values during program execution.
- Improve code readability with meaningful names.
- Perform calculations and operations easily.

Two Main Types of Variables:

1. **Local Variables**

- These are **declared inside a function.**
- They **exist only while the function is running.**
- You **cannot use them outside** the function.

Example:

```
def show_message():
    msg = "Hello from inside the function!"   # local variable
    print(msg)

show_message()
print(msg)   # ❌ This will cause an error because 'msg' is local
```

Output:

```
Hello from inside the function!
NameError: name 'msg' is not defined
```

Explanation:

- msg is created **inside** the function.
- Once the function finishes, msg is **destroyed.**
- So, you cannot use it outside.

Solved Example: Local Variable

```
def greet():
    name = "Abhyas"   # Local variable
    print("Hello,", name)
```

Scope of Variable in Python

```
greet()
print(name)   # ❌ Error: name not defined
```

Output:
Hello, Abhyas
NameError: name 'name' is not defined

Explanation:
- name is local — it lives only inside greet().
- Once the function ends, the variable is **destroyed**.

2. Global Variables
- These are **declared outside all functions**.
- They can be used **anywhere in the program** (inside or outside functions).
- You can **read** a global variable inside a function,

but to **modify it**, you must use the global keyword.

Example:

```
name = "XYZABC"   # global variable

def greet():
    print("Hello,", name)   # can access global variable

greet()
print("Welcome,", name)
```

Output:
Hello, XYZABC
Welcome, XYZABC

Explanation:
- The variable name is global.
- It is accessible both inside and outside the function.

Solved Example: Global Variable

```
city = "Ambala"   # Global variable

def show_city():
    print("Inside function:", city)

show_city()
print("Outside function:", city)
```

Scope of Variable in Python

Output:
Inside function: Ambala
Outside function: Ambala

Explanation:
- The variable city is global.
- Both the function and the rest of the program can use it.

Global variables are useful for:
- Storing data that needs to be shared between multiple functions.
- Keeping track of global states (like counters or settings).

Both Global and Local Variables with the Same Name
If a local and global variable have the **same name**, Python will use the **local variable** inside the function.

Example:
```
x = 100  # global variable

def demo():
    x = 50   # local variable
    print("Inside function:", x)

demo()
print("Outside function:", x)
```

Output:
Inside function: 50
Outside function: 100

Explanation:
• Inside the function → local variable x = 50
• Outside the function → global variable x = 100
They are **different** variables even though they have the same name.

The "global" Keyword
If you want to **change a global variable inside a function**, you must use global.

Example:
```
total = 0

def add():
    global total
    total += 5
```

Scope of Variable in Python

```
    print("Total inside:", total)

add()
print("Total outside:", total)
```

Output:
Total inside: 5
Total outside: 5

Explanation:
1. **total = 0** → This creates a **global variable** named total.
It lives outside the function and is accessible anywhere in the program.
2. **def add():** → You define a function.
3. **global total** →
This line tells Python:
"I want to use the global total variable inside this function, not create a new one."
Without this line, Python would treat total inside the function as a **new local variable**, leading to an error.
4. **total += 5** →
Adds 5 to the global variable total.
Now total becomes 5.
5. **print("Total inside:", total)** →
Prints the value of total inside the function → **5**
6. **add()** →
Calls the function and executes the above steps.
7. **print("Total outside:", total)** →
Prints the same global variable from outside → **5**

Assignment

Ques 1: Modify a Global Variable
Write a Python program with a global variable count = 0.
Create a function increment() that adds 1 to count each time it is called using the global keyword.
Call the function 3 times and print the final value of count.

Ques 2: Global vs Local Variable
Create a program where:
- You have a global variable x = 50
- Define a function show() that has a local variable x = 20
- Print x inside and outside the function

Observe how Python treats local and global variables.

Scope of Variable in Python

Ques 3: Update Global Variable Inside Function
Write a Python program that:
- Has a global variable total = 100
- A function update_total() increases total by a number entered by the user
- Use global total inside the function
- Print the value of total before and after calling the function.