In Python, the term "scope" refers to the region of a program where a particular variable is accessible. Python has four main scopes:

1. **Local scope: Variables defined within a function have local scope. They can only be accessed within that function.**
2. **Enclosing scope: If a function is defined within another function, the inner function has access to variables in the outer (enclosing) function's scope.**
3. **Global scope: Variables defined at the top level of a module or explicitly declared as global have global scope. They can be accessed from anywhere within the module.**
4. **Built-in scope: Python has a set of built-in functions and types that are always available.**

Global variables in Python are variables that are declared outside of any function or in global scope. They can be accessed and modified from any part of the program.

python Copy code

```python
# Global variable
global_var = 10

def func():
    # Local variable
    local_var = 20

    # Accessing global variable
    print("Inside func(): global_var is", global_var)

    # Modifying global variable
    global global_var
    global_var = 30

    print("Inside func(): global_var modified to", global_var)

func()

print("Outside func(): global_var is", global_var)
```

In this example:

- **`global_var` is a global variable accessible from within the function `func()`.**
- **`local_var` is a local variable only accessible within `func()`.**
- **We use the `global` keyword inside `func()` to modify the value of `global_var`.**

Output:

```
go                                                    Copy code

Inside func(): global_var is 10
Inside func(): global_var modified to 30
Outside func(): global_var is 30
```

Using the `global` keyword allows us to modify global variables from within a function. However, it's generally considered a good practice to avoid global variables when possible, as they can make code harder to understand and maintain.

A global variable cannot be changed in function as a global variable.

If we want to change global then define that variable in function as global l.

Code:

```
# Global variable
global_var = 10


def funcA():
    # Accessing global variable
    print("Inside funcA(): global_var is",
global_var)


def funcB():
```


```
    # Trying to modify global variable without
'global' keyword
    global_var = 20
    print("Inside funcB(): global_var is",
global_var)

def funcC():
    # Modifying global variable using 'global'
keyword
    global global_var
    global_var = 30
    print("Inside funcC(): global_var modified
to", global_var)


# Function with nested scope
def outer_function():
    outer_var = "I am outer"

    def inner_function():
        nonlocal outer_var
        outer_var = "I am inner"
        print("Inside inner_func(): outer_var
is", outer_var)
    inner_function()
    print("Inside outer_func(): outer_var is",
outer_var)

# Demonstrate scopes and global variable
modification
funcA()
funcB()  # This won't modify the global_var
```


```
funcC()
print("Outside all functions: global_var is",
global_var)

# Demonstrate nested scope
outer_function()
```

Output:

```
/root/PycharmProjects/pythonProject/.venv/bin/python /root/PycharmProjects/pythonProject/21.py
Inside funcA(): global_var is 10
Inside funcB(): global_var is 20
Inside funcC(): global_var modified to 30
Outside all functions: global_var is 30
Inside inner_func(): outer_var is I am inner
Inside outer_func(): outer_var is I am inner
```