

Global & Local Variable :

1. Local Variable

- A variable that is created inside a function.
- It can be used only inside that function.
- Once the function ends, the variable is gone.

2. Global Variable

- Declared outside any function.
- Can be used anywhere in the program.
- Stored in the program's global memory.

```
In [1]: a = 10 # global variable

def something():
    b = 15 # local variable
    print("In Function:", b) # b is local → works fine
    print("Out Function:", a) # a is global → can be accessed inside function

# Right now ✗ no output because you only defined the function. not call the Function
```

```
In [2]: a = 10 # global variable

def something():
    b = 15 # local variable
    print("In Function:", b) # b is local → works fine

print("Out Function:", a) # a is global → can be accessed anywhere
```

Out Function: 10

```
In [3]: a = 10 # global variable

def something():
    a = 15 # local variable (inside function, but function not called)

print("In Function:", a)
print("Out Function:", a)
```

In Function: 10

Out Function: 10

```
In [4]: a = 10 # global variable

def something():
    a = 15 # local variable (inside function)
    b = 8 # local variable
```

```
print(b)
print(a)
```

*# If you don't call the function:
Nothing runs. No output.*

```
In [5]: a = 10 # global variable

def something():
    a = 15 # local variable (inside function)
    b = 8 # local variable
    print("In Function",b)

something()

print("Out Function",a)
```

In Function 8
Out Function 10

```
In [6]: a = 10 # global variable

def something():
    print("In Function", a)

something()

print("Out Function", a)

# If you should not have local variable then Global variable acts as a local variable.
```

In Function 10
Out Function 10

```
In [7]: a = 10 # global variable
        b = 25 # global variable

def something():
    b = 15 # local variable (only inside function)
    # if we remove this variable, then Python will look for global b
    print("In Function", b)

something()

print("Out Function", a)
```

In Function 15
Out Function 10

1. What is global?

- The global keyword is used inside a function to tell Python that you want to use the global variable instead of creating a new local one.

- Without global, if you assign a value to a variable inside a function, Python will treat it as local.

[OR]

- Global Function is a built in function that returns dictionary representing the current global symbol table. it allows you to access and modify global variables programmatically.

2. Why use global? - To read and modify a global variable inside a function.

3. Important Notes

- global works only inside functions.
- You can declare multiple globals:

```
In [8]: a = 10 # global variable

def something():
    global a # use the global 'a'
    b = 15 # local variable
    print("In Function", b) # local b
    print("Global Variable:", a) # global a

something()

print("Out Function", a)
```

In Function 15
Global Variable: 10
Out Function 10

```
In [9]: x = 10 # global variable

def update_x():
    global x # Declare that we are using global variable x .
    x += 10 # adds 10 to the global x

update_x() # calls the function
print(x) # prints the value of x
```

20

```
In [10]: x = 10

def Update_x():
    globals()['x'] += 20

Update_x()
print(x)
```

30

```
In [11]: # import keyword
# keyword.kwlist
```

```
In [12]: # len(keyword.kwlist)
```

How to pass the LIST to a FUNCTION:

```
In [13]: def count(lst):
    even = 0
    odd = 0

    for i in lst:
        if i % 2 == 0: # checks if number is divisible by 2
            even += 1 # increments even count
        else:
            odd += 1 # increments odd count
    return even, odd # returns both counts

lst = [10, 9, 8, 23, 50, 8, 9, 100]

even, odd = count(lst) # unpack the returned tuple

print("Even Number:", even)
print("Odd Number:", odd)
```

Even Number: 5

Odd Number: 3

```
In [14]: def fib(n):
    a = 0
    b = 1

    print(a)
    print(b)

    for i in range(0,n):
        c = a + b
        a = b
        b = c

        print(c)

fib(10)
```

0
1
1
2
3
5
8
13
21
34
55
89

In [15]: *# Factorial of a number in python:*

```
def fact(n):  
    f = 1  
    for i in range(1, n+1):  
        f = f+1  
  
    return f  
x = 5  
result = fact(x)  
print(result)
```

6

In [16]:

```
def wish():  
    print("Hello")  
    print("Hii")  
  
wish()
```

Hello

Hii

In [17]:

```
def wish():  
    print('Hello')  
    print('Hii')  
    wish()  
wish()
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Hello
Hii
Hello
Hii
Hello
Hii
Hello
Hii
Hello
Hii
Hello


```

-----
RecursionError                                Traceback (most recent call last)
Cell In[17], line 5
      3     print('Hii')
      4     wish()
----> 5     wish()

Cell In[17], line 4, in wish()
      2     print('Hello')
      3     print('Hii')
----> 4     wish()

Cell In[17], line 4, in wish()
      2     print('Hello')
      3     print('Hii')
----> 4     wish()

[... skipping similar frames: wish at line 4 (2970 times)]

Cell In[17], line 4, in wish()
      2     print('Hello')
      3     print('Hii')
----> 4     wish()

Cell In[17], line 2, in wish()
      1     def wish():
----> 2         print( )
      3         print('Hii')
      4         wish()

File ~\AppData\Roaming\Python\Python313\site-packages\IPython\core\interactiveshell.p
y:3056, in InteractiveShell._tee.<locals>.write(data, *args, **kwargs)
    3054     if not data:
    3055         return result
-> 3056     execution_count = self.execution_count
    3057     output_stream = None
    3058     outputs_by_counter = self.history_manager.outputs

File ~\AppData\Roaming\Python\Python313\site-packages\traitlets\traitlets.py:687, in T
raitType.__get__(self, obj, cls)
    685         return self
    686     else:
--> 687         return t.cast(G, self.get(obj, cls))

File ~\AppData\Roaming\Python\Python313\site-packages\traitlets\traitlets.py:666, in T
raitType.get(self, obj, cls)
    664         raise TraitError("Unexpected error in TraitType: default value not set pro
perly") from e
    665     else:
--> 666         return t.cast(G, value)

RecursionError: maximum recursion depth exceeded

```

```

In [ ]: import sys
        sys.setrecursionlimit()

```

```
In [ ]: import sys
        sys.setrecursionlimit(200)
        print(sys.getrecursionlimit())
```

```
In [ ]: import sys
        sys.getrecursionlimit()
```

```
In [ ]: import sys
        sys.setrecursionlimit(150)

        i = 0

        def wish():
            global i
            i += 1
            print('Hello',i)
            wish()

        wish()
```

Factorial using Recursion

```
In [ ]: def fact (n):
        if n==0:
            return 1
        return n * fact(n-1)
        result = fact(5)

        result
```