Global Variables

A variable created outside the function is called global variable. Global variables are used by one more than one function (OR) in order to share data between number of functions we declare global variables.

The life time global variables are until execution program. Once the program execution is completed these variables deleted from memory.

Global variable memory is allocated when program is executed. Global variable memory is allocated with in global namespace.

Example:	**Output**

x=100 # Global Variable
y=200 # Global Variable
 	
def fun1():
  print("inside fun1")
  print(x,y,sep=",")
 	
def fun2():
  print("inside fun2")
  print(x,y,sep=",")
 	
fun1()
fun2()
	inside fun1
	100,200
	inside fun2
	100,200

Example	**Output**

def add():
  print(f'sum of {n1}+{n2}={n1+n2}')
def sub():
  print(f'diff of {n1}-{n2}={n1-n2}')
def multiply():
  print(f'product of
	Enter first number 5
	Enter second number 2
	sum of 5+2=7
	diff of 5-2=3
	product of 5*2=10
	division of 5/2=2.5

{n1}*{n2}={n1*n2}') def div(): print(f'division of {n1}/{n2}={n1/n2}') n1=int(input("Enter first number ")) n2=int(input("Enter second number ")) add() sub() multiply() div()	

Example
def fun1():
  print(x)
 	
fun1()
x=100
def fun2():
  print(x)
 	
fun2()
	Output
	Traceback (most recent call last):
	  File "E:/python5pmjun/test202.py", line 4, in <module>
	  fun1()
	  File "E:/python5pmjun/test202.py", line 2, in fun1
	  print(x)
	NameError: name 'x' is not defined

Example
x=100 # G.V
 	
def fun1():
  x=200 # L.V
  print(f'local variable x={x}')
 	
def fun2():
  print(f'global variable x={x}')
 	
fun1()
	Output
	local variable x=200
	global variable x=100

fun2()	

A function can access global variable directly but cannot modify or assign value directly.

global keyword

The global statement is a declaration which holds for the entire current code block. It means that the listed identifiers are to be interpreted as globals. It would be impossible to assign to a global variable without global, although free variables may refer to globals without being declared global.

Names listed in a global statement must not be used in the same code block textually preceding that global statement.

Syntax: global variable-name,variable-name

Example	**Output**

x=100 # Global Variable
 	
def fun1():
  global x
  x=500
 	
def fun2():
  print(x)
 	
fun2()
fun1()
fun2()
	100
	500

Example	**Output**

x=100
def fun1():
  global x,y
  x=500
  y=600
	500 600

```
def fun2():
    print(x,y)


fun1()
fun2()
```

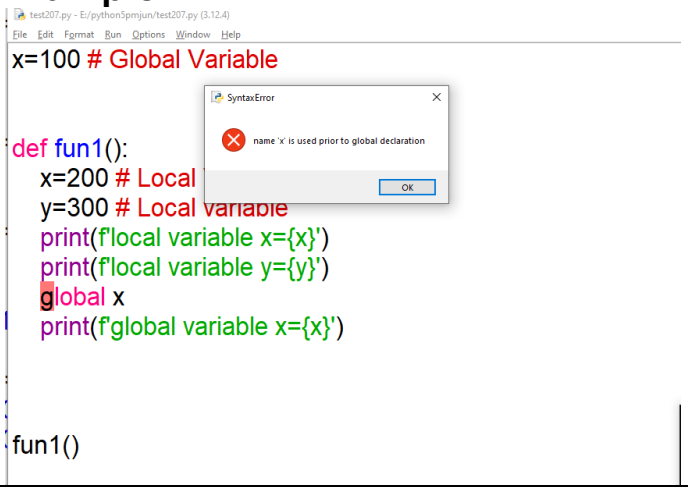
Example

```
# find area of triangle

base,height=0,0 # Global Variables
def read_dim():
    global base,height
    base=float(input("Enter Base "))
    height=float(input("Enter Height "))

def find_area():
    area=base*height
    print(f'Area of triangle is {area:.2f}')


read_dim()
find_area()
```

Output

```
Enter Base 1.2
Enter Height 1.5
Area of triangle is 1.80
```

Example



Error:

Names listed in a global statement must not be used in the same code block textually preceding that global statement.

globals()
Return the dictionary implementing the current module namespace. All global names are stored inside a dictionary called global (namespace name).

Example	**Output**

x=100 # Global Variable
def fun1():
  x=200 # Local Variable
  y=300 # Local variable
  print(f'local variable x={x}')
  print(f'local variable y={y}')
  a=globals()
  print(f'global variable x={a['x']}')
 	
def fun2():
  g=globals()
  g['x']=700
  print(g['x'])
  x=900 # L.V
  print(x)
  print(g['x'])
 	
fun1()
fun2()
fun1()
	local variable x=200
	local variable y=300
	global variable x=100
	700
	900
	700
	local variable x=200
	local variable y=300
	global variable x=700

Function with parameters or arguments

Function with parameters receives values from caller (OR) if a function required input to perform operation, it must be defined with parameters.
Parameters are local variables which receive values.

Python allows defining function with 3 types of parameters/arguments

1. Function with required arguments
    a. Function with required positional arguments
    b. Function with required keyword arguments
2. Function with default arguments
3. Function with variable length arguments/arbitrary arguments
    a. Function with variable length positional arguments
    b. Function with variable length keyword arguments