Global variables are created to share data between number of functions within program/module.

```
n1=int(input("Enter First Number"))  # G.V
n2=int(input("Enter Second Number")) # G.V

def add():
    print(f'Sum of {n1} and {n2} is {n1+n2}')

def sub():
    print(f'Diff of {n1} and {n2} is {n1-n2}')

add()
sub()
```

Output:

```
Enter First Number20
Enter Second Number10
Sum of 20 and 10 is 30
Diff of 20 and 10 is 10
```

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

```
def fun1():
    x=100 # L.V
    print(f'Local Variable x={x}')

y=20 # G.V
def fun2():
    print(f'Global variable y={y}')

def fun3():
    z=50 # L.V
    print(f'Local Variable z={z}')
    y=40 # L.V
    print(f'Local Variable y={y}')
```

```
fun1()
fun2()
fun3()
print(f'Global Variable y={y}')
```

Output

```
Local Variable x=100
Global variable y=20
Local Variable z=50
Local Variable y=40
Global Variable y=20
```

global keyword

This keyword is used to perform two operations.

1. Updating value of global variable within function
2. Creating global variable within function

Syntax: global variable,variable,variable

Example:

```
x=100 # G,V
def fun1():
    print(f'Global variable x={x}')

def fun2():
    y=200 # L.V
    print(f'Local Variable y={y}')
    global x
    x=500
    print(f'Global Variable x={x}')

fun1()
fun2()
fun1()
```

Output:

Global variable x=100
Local Variable y=200
Global Variable x=500
Global variable x=500

Example:

```
# A Program to find area of triangle

base=0.0 # G.V
height=0.0 # G.V

def read():
    global base,height
    base=float(input("Enter Base "))
    height=float(input("Enter Height "))

def find_area():
    area=0.5*base*height
    print(f'Area of triangle is {area:.2f}')


read()
find_area()
```

Output

```
Enter Base 1.2
Enter Height 1.3
Area of triangle is 0.78
```

globals()

it is a predefined function of python. This function returns reference/address of global dictionary. Python virtual machine store current program/module global names inside a dictionary.

Example:

```
x=100 # G.V
```

```
def fun1():
    a=200 # L.V
    b=300 # L.V
    print(f'Local variable a={a}')
    print(f'Local variable b={b}')
    x=500 # L.V
    print(f'Local variable x={x}')
    gd=globals()
    print(f'Global variable x={gd["x"]}')
    gd['x']=500

fun1()
print(f'Global variable x={x}')
```

Output:

```
Local variable a=200
Local variable b=300
Local variable x=500
Global variable x=100
Global variable x=500
```

Example

```
def fun1():
    gd=globals()
    gd['x']=100 # GV
    gd['y']=200 # GV
    print(f'Global variable x={gd["x"]}')
    print(f'Global variable y={gd["y"]}')
    x=100 # Local variable
    print(f'Local variable x={x}')


fun1()
print(f'Global variable x={x}')
```

Output

```
Global variable x=100
Global variable y=200
Local variable x=100
```

Global variable x=100

Function with parameters or arguments
Function with parameters or arguments receive values at the time of invoking or calling function.
If function required input to perform operation, it is defined with parameters or arguments.
Python allows writing function with 4 types of arguments.

1. Required Positional arguments
2. Default or Optional arguments
3. Variable length arguments
4. Keyword arguments

Parameters or arguments are called local variables. Local variables memory is allocated when function is called and memory is deleted or de-allocated after execution of function.

Function required or required positional arguments
Required arguments required values at the time invoking or calling function.
If values of required arguments are not given python translator generate TypeError.

Syntax:
def <function-name>(arg-name,arg-name,arg-name,…):
    statement-1
    statement-2