Types of Variables

In [ ]:
```
Functional Programming :
    Two Types of Variables are there:
        1.Local Variables
        2.Global Variable
```

Global Variable

In [ ]:
```
Global Variable--> are those variable that are declared and intilized outside any function.
```

Example

In [1]:
```
a=10     #Global Variable
def check():
    print(a)
    return ""
print(check())
```

```
10
```

Local Variable

In [ ]:
```
Local Variable --> are those variable that are declared and initilized inside the function
```

Example

In [2]:
```
def check():
    a=10   #Local Variable
    b=20     #local Variable
    print(a)
    print(b)
    return ""
print(check())
```

```
10
20
```

Scopes of the variables:

In [ ]:
```
Scopes of the variables:

    a.Global variable --> global variables can be accessed any where in the program.
    (either outside or inside)

    b.Local variable can only be accessed within the function.You cannot access local variable
      outside
```

Local Variable Scope

In [3]:
```
def check():
    abc=10   #Local Variable
    bd=20    #local Variable
    return ""
print(abc)   #Error

#Local variable you cannot access outside the function
#.The scope of local variable is within a function.
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [3], in <cell line: 5>()
      3     bd=20    #local Variable
      4     return ""
----> 5 print(abc)

NameError: name 'abc' is not defined
```

Global Variable Scope

In [7]:
```
abc=200
def check():
    print("Inside function",abc)
    print(abc)
    return ""

def checking():
    print("Inside second function",abc)
    print(abc)
    return ""
print("Outside Function ",abc)
print(check())
print(checking())
print(abc)


Note --> global variable can be assessed anywhere in the program.The scope of global variable is
         within a program.
```

```
Outside Function  200
Inside function 200
200

Inside second function 200
200

200
```

Priority of local and global variable

In [ ]:
```
--> if the name of local variable and global variable both are same then which variable will be
    considered inside the scope and which variable  will be worked outside the scope.

--> If local and global variable name are same then PVM will check the variable first in local scope
    if the variable is present at local scope then that variable will be printed and if the variable
    is not present in local scope then it will check at global scope if the variable is present at
    global scope that that will be printed if the variable is not present in global as well as
    local then error will be there
```

In [8]:
```
x=10   #Global
z=20   #global
def f1():
    x=1000   #local variable
    print(x)
    print(z)
    return ""
print(f1())  #1000
print(x)    #10
```

```
1000
20

10
```

Global keyword

In [ ]:
```
Global Keyword is used to declare global variable inside the function.
```

Example

In [9]:
```
az=10
def f1():
    print(az)
    return ""
def f2():
    global az
    az=2777
    print(az)
    return ""
print(f2())
print(f1())
```

```
777

777
```

In [10]:
```
abc=999
def f1():
    print(abc+100)
    return ""

def f2():
    global abc
    abc=900
    print(abc+800)
    return ""

def f3():
    print(abc)
    return ""

print(f1())
print(f3())

#Global keyword variable will be first statement of the function
```

```
1099

999
```

Nested Functions

In [ ]:
```
--> Nested Functions are also possible in Python
--> Function inside the function is known as Nested Function.
--> Inner function will be executed first.then outer function
--> You cannot call inner function directly
--> Inner function is always local to the outer function
```

Example

In [11]:
```
def add(x,y):
    def sub(x,y):
        return x-y   #-10
    return sub(10,20)  #return -10

print(add(10,20))
```

```
-10
```

In [12]:
```
def outer():
    def inner():
        print("Inner function")
        return ""
    return inner()

print(outer())
```

```
Inner function
```

In [13]:
```
def outer():
    print("Outer Function")   #1
    def inner():
        print("Inner function")  #3
        return ""
    return inner()
    print("Inner function outside")  #2


print(outer())
```

```
Outer Function
Inner function
```

In [14]:
```
def add(a,v):
    return a-v,a+v
    return
    print(b)
    print(c)
print(add(10,20))
```

```
(-10, 30)
```

In [15]:
```
def add(a,b):
    def sub(c,d):
        return c-d  #5-10  -->-5
    return sub(a,b)   #sub(5,10)   return -5
    return a
res=add(5,10)
print(res)
```

```
-5
```

In [16]:
```
def div(a,b,c):   #10,15,20
    def mul(c,d,e):     #10,15,20
        return c*d*e     #return 10*15*20 -->3000
    return mul(a,b,c)       #return mul(10,15,20)   #return 3000
print(div(10,15,20))
```

```
3000
```

In [17]:
```
def outer():
    print("Outer function")
    def add(c,d,e):
        return c+d+e   #45
    print("Outer add function")
    return add(10,15,20)   #return 45
    return "Hello world"
print(outer()) #Outer Function , Outer add function , 45
```

```
Outer function
Outer add function
45
```

Note --> Inside a function after return statement nothing will be executed. No any statement after return statement will be considered

Recursive Function

In [ ]:
```
Recursive Function  --> When a function call itself then such types of funcitons are known as
                        recursive function
```

In [ ]:
```
factorial(3):
    3*factorial(2)
    3*2*factorial(1)
    3*2*1*factorial(0)
    3*2*1*1
    3*2*1
    6
```

Example

In [18]:
```
def factorial(num):
    if num==0:
        return 1
    return num*factorial(num-1)
print(factorial(5))
```

```
120
```