Local variable and Global variable

Local var:Var which is defined inside fun body called local var,purpose is to store temp data which is comming from fun call Global var:Var which is common for all fun call is Global var,must defined fun call otherwise we get NameError.

In [17]:
```
#program for global var
lang='Python'
def learnAi():
    sub1=('AI')
    print('To develop {} application ,we use {} language'.format(sub1,lang))
def DS():
    sub2='Data Science'
    print('To work in {} ,you should use {} language'.format(sub2,lang))
def ML():
    sub3='Machine Language'
    print('To develop {} applications,we use {} language'.format(sub3,lang))
#main
lang='Python'  #here lang is global var

learnAi()
DS()
ML()
```

```
To develop AI application ,we use Python language
To work in Data Science ,you should use Python language
To develop Machine Language applications,we use Python language
```

In [29]:
```
#program for global var
lang='Python'
def learnAi():
    sub1=('AI')
    print('To develop {} application ,we use {} language'.format(sub1,lang))
def DS():
    sub2='Data Science'
    print('To work in {} ,you should use {} language'.format(sub2,lang))
def ML():
    sub3='Machine Language'
    print('To develop {} applications,we use {} language'.format(sub3,lang))
#main
learnAi()

#lang='Python'  #here lang is global var

DS()
ML()
```

```
To develop AI application ,we use Python language
To work in Data Science ,you should use Python language
To develop Machine Language applications,we use Python language
```

In [5]:
```
#program for global var
lang='Python'
def learnAi():
    sub1=('AI')
    print('To develop {} application ,we use {} language'.format(sub1,lang))
def DS():
    sub2='Data Science'
    print('To work in {} ,you should use {} language'.format(sub2,lang))
def ML():
    sub3='Machine Language'
    print('To develop {} applications,we use {} language'.format(sub3,lang))
#main
learnAi()
DS()
ML()

lang='Python'  #here lang is global var
```

```
To develop AI application ,we use Python language
To work in Data Science ,you should use Python language
To develop Machine Language applications,we use Python language
```

In [1]:
```
#program for global var
#lang='Python'
def learnAi():
    sub1=('AI')
    print('To develop {} application ,we use {} language'.format(sub1,lang))
def DS():
    sub2='Data Science'
    print('To work in {} ,you should use {} language'.format(sub2,lang))
def ML():
    sub3='Machine Language'
```

```
    print('To develop {} applications,we use {} language'.format(sub3,lang))
#main
learnAi()
DS()
ML()

lang='Python'  #here lang is global var
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 13
     11     print('To develop {} applications,we use {} language'.format(sub3,lang))
     12 #main
---> 13 learnAi()
     14 DS()
     15 ML()

Cell In[1], line 5, in learnAi()
      3 def learnAi():
      4     sub1=('AI')
----> 5     print('To develop {} application ,we use {} language'.format(sub1,lang))

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

In [1]:
```
#program for global var

def learnAi():
    sub1=('AI')
    print('To develop {} application ,we use {} language'.format(sub1,lang))
def DS():
    sub2='Data Science'
    print('To work in {} ,you should use {} language'.format(sub2,lang))
def ML():
    sub3='Machine Language'
    print('To develop {} applications,we use {} language'.format(sub3,lang))
#main
learnAi()
DS()
ML()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 13
     11     print('To develop {} applications,we use {} language'.format(sub3,lang))
     12 #main
---> 13 learnAi()
     14 DS()
     15 ML()

Cell In[1], line 5, in learnAi()
      3 def learnAi():
      4     sub1=('AI')
----> 5     print('To develop {} application ,we use {} language'.format(sub1,lang))

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

Global Keyword

When we want to modify global var inside fun definition we use global keyword otherwise we get UnboundLocalError.

In [6]:
```
a=10
def modify1():
    a=a+1
    print(a)
modify1()
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
Cell In[6], line 5
      3     a=a+1
      4     print(a)
----> 5 modify1()

Cell In[6], line 3, in modify1()
      2 def modify1():
----> 3     a=a+1
      4     print(a)

UnboundLocalError: cannot access local variable 'a' where it is not associated with a value
```

In [8]:
```
a=10
def modify1():
    global a
```

```
    a=a+1
    print(a)
modify1()
```

```
11
```

In [10]:
```
a=10
b=20
def modify():  #here we are not modifying ,we are just assigning value so no need to use global kwd
    c=a+b
    print(c)
modify()
```

```
30
```

In [14]:
```
a=10
def modify1():
    global a
    a=a+1
    print(a)
def modify2(): #compulsary to mentioned global kwd inside fun definition
    a=a*2
    print(a)
modify1()
modify2()
```

```
11
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
Cell In[14], line 10
      8     print(a)
      9 modify1()
---> 10 modify2()

Cell In[14], line 7, in modify2()
      6 def modify2():
----> 7     a=a*2
      8     print(a)

UnboundLocalError: cannot access local variable 'a' where it is not associated with a value
```

In [16]:
```
a=10
def modify1():
    global a
    a=a+1
    print(a)
def modify2():  #compulsary to mentioned global kwd inside fun definition
    global a
    a=a*2
    print(a)
modify1()
modify2()
```

```
11
22
```

Global var and Local var and globals()

When we come across same global var and local var in same fun ,PVM gives preference to local var only ,not for global var. To retrive values from global var name and local var values we used globals(). globals() print data in the form of dict , key and value pair, where key is a global var and value is a local var.

In [27]:
```
#program for global var and local var without globals()
a=10
b=20
c=30
def oper():

    a=100
    b=200
    c=300

    print(a,b,c)
oper()
```

```
100 200 300
```

In [29]:
```
#program for globals()
a=10
b=20
```

```
c=30
def oper():
    obj=globals()
    a=100
    b=200
    c=300
    res=obj ['a'],obj ['b'],obj['c']
    print(res)
oper()
```

```
(10, 20, 30)
```

In [35]:
```
#program for globals()
a=10
b=20
c=30
def oper():
    obj=globals()
    a=100
    b=200
    c=300
    res=a+b+c,obj ['a'],obj ['b'],obj['c']

    print(res)
oper()
```

```
(600, 10, 20, 30)
```

In [37]:
```
#program for globals()
a=10
b=20
c=30
def oper():
    obj=globals()
    a=100
    b=200
    c=300
    res=a+b+c,obj ['a'],obj ['b'],obj['c']
    for val in res:
        print(val)

oper()
```

```
600
10
20
30
```

In [ ]:
```
#program for types of defining methods in globals()
a=10
b=20
c=30
d=40
def oper():
    obj=globals()
    for gn,gv in obj.items():
        print(gn,gv)
        print('='*50)
        print('globals var type 1')
        print('='*50)
        print('a is ',obj['a'])
        print('b is ', obj['b'])
        print('c is ', obj['c'])
        print('d is ',obj['d'])

        print('='*50)
        print('globals var type 2')
        print('='*50)
        print('a is',globals()['a'])
        print('b is ',globals()['b'])
        print('c is ',globals()['c'])
        print('d is ',globals()['d'])

        print('='*50)
        print('globals var type 3')
        print('='*50)
        print('a is ',obj.get('a'))
        print('b is ',obj.get('b'))
        print('c is ',obj.get('c'))
        print('d is ',obj.get('d'))
```

```
        print('='*50)
        print('globals var type 4')
        print('='*50)
        print('a is ',globals().get('a'))
        print('b is ',globals().get('b'))
        print('c is ',globals().get('c'))
        print('d is ',globals().get('d'))

oper()
```

In [102…

```
a=10           #here both global var and local var is different so no need to use globals()
b=20
c=20
d=40
def oper():
    x=100
    y=200
    z=300
    res1=a,b,c,d,x,y,z
    res2=a+b+c+d+x+y+z
    print('\tWITHOUT ADDITION')
    print(res1)
    print('\nADDITION')
    print(res2)
oper()
```

```
       WITHOUT ADDITION
(10, 20, 20, 40, 100, 200, 300)

ADDITION
690
```