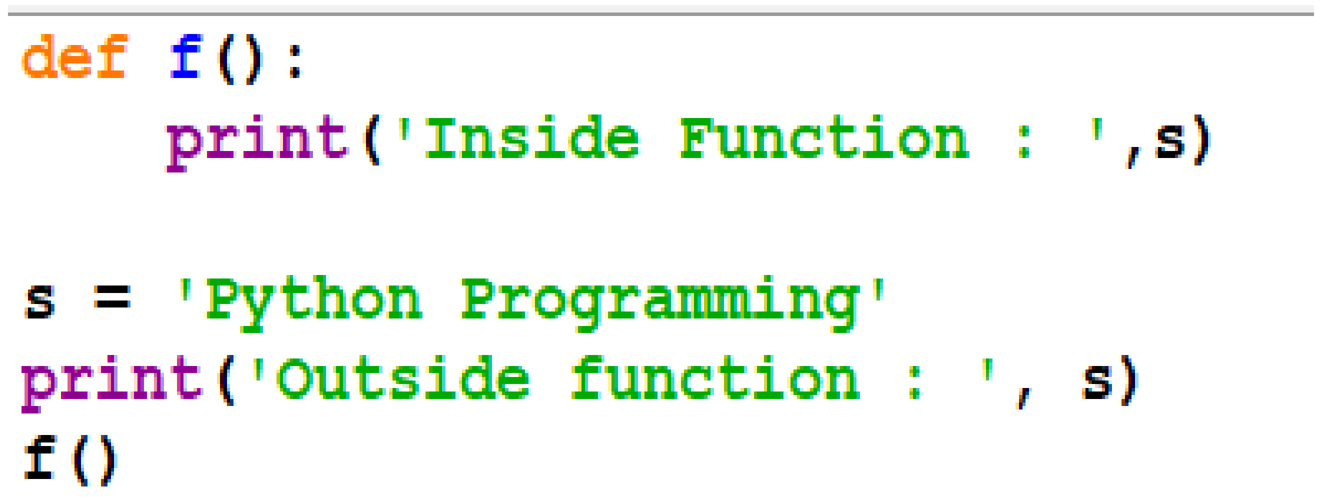
SCOPE OF VARIABLES

Scope of variable

- The location where we can find a variable and also access it if required is called the **scope of a variable**.

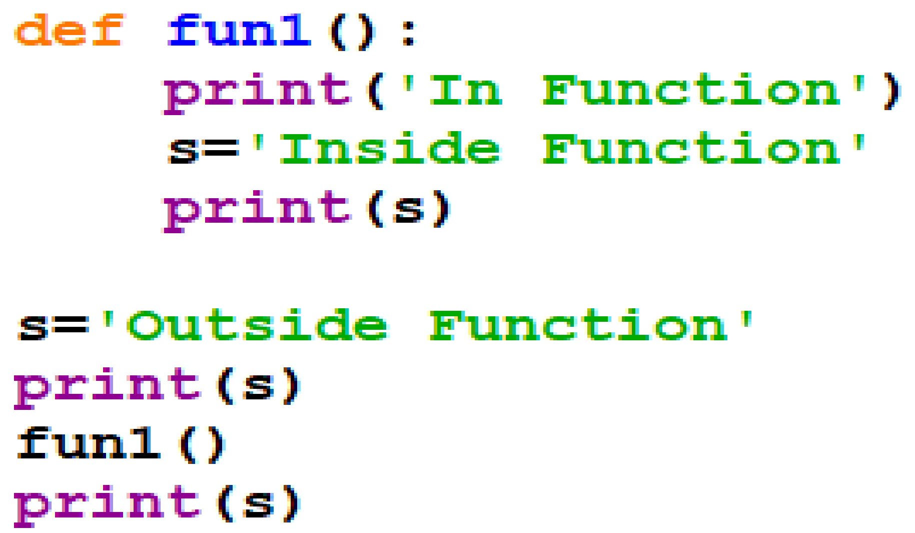
Global and local variables

- Global variables are the ones that are defined and declared outside any function and are not specified to any function. They can be used by any part of the program.



```python
def f():
    print('Inside Function : ',s)

s = 'Python Programming'
print('Outside function : ', s)
f()
```

- Now suppose a variable with the same name is defined inside the scope of function as well then it will print the value given inside the function only and not the global value.



```
def fun1():
    print('In Function')
    s='Inside Function'
    print(s)

s='Outside Function'
print(s)
fun1()
print(s)
```

- The question is, what will happen if we change the value of s inside of the function f()? Will it affect the global s as well?

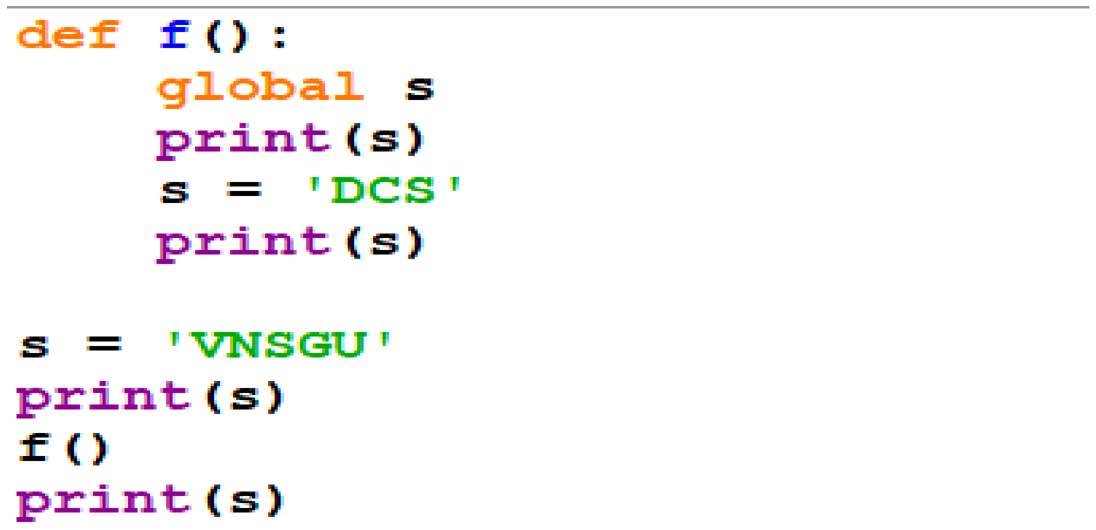
```
def f():
    print(s)
    s = 'DCS'
    print(s)

s='VNSGU'
print(s)
f()
```

Python "assumes" that we want a local variable due to the assignment to s inside of f(), so the first print statement throws this error message.

Output :

```
VNSGU
Traceback (most recent call last):
  File "G:/VNSGU/MCA 2/2022-23 Python/Programs/p33_Scope3.py", line 8, in <module>
    f()
  File "G:/VNSGU/MCA 2/2022-23 Python/Programs/p33_Scope3.py", line 2, in f
    print(s)
UnboundLocalError: local variable 's' referenced before assignment
>>>
```

Any variable which is changed or created inside of a function is local, if it hasn't been declared as a global variable. To tell Python, that we want to use the global variable, we have to use the keyword global



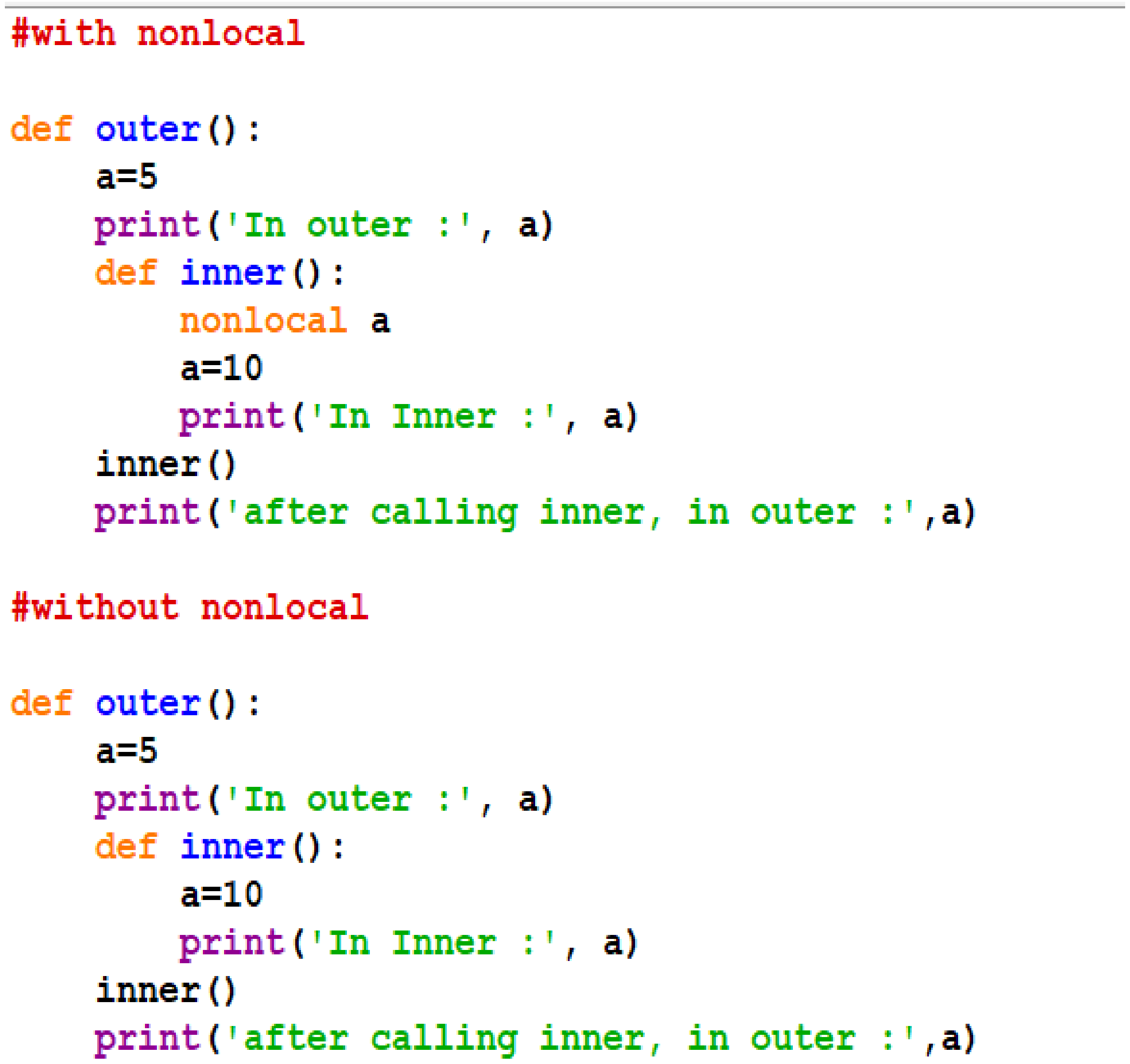
```
def f():
    global s
    print(s)
    s = 'DCS'
    print(s)

s = 'VNSGU'
print(s)
f()
print(s)
```

Example :

```
def f():
    print('Inside f :', a)

def g():
    a=2
    print('Inside g :', a)

def h():
    global a
    a=3
    print('Inside h :', a)

a=1
print('global :', a)
f()
print('global :', a)
g()
print('global :', a)
h()
print('global :', a)
```

Nonlocal keyword

- In Python, nonlocal keyword is used in the case of nested functions. This keyword works similar to the global, but rather than global, this keyword declares a variable to point to the variable of outside enclosing function, in case of nested functions.



```python
#with nonlocal

def outer():
    a=5
    print('In outer :', a)
    def inner():
        nonlocal a
        a=10
        print('In Inner :', a)
    inner()
    print('after calling inner, in outer :',a)

#without nonlocal

def outer():
    a=5
    print('In outer :', a)
    def inner():
        a=10
        print('In Inner :', a)
    inner()
    print('after calling inner, in outer :',a)
```

```
#with global

def outer():
    a=5
    print('In outer :', a)
    def inner():
        global a
        a=10
        print('In Inner :', a)
    inner()
    print('after calling inner, in outer :',a)


a=15
print('outside function:', a)
outer()
print('outside function:', a)
```