Local & Global Variable

In [1]:
```
# global
a=100
def f1():
    print("Function 1:",a)
def f2():
    print("Function 2:",a)
def f3():
    print("Function 3:",a)

f1()
f2()
f3()
```

```
Function 1: 100
Function 2: 100
Function 3: 100
```

In [4]:
```
# Local
a=100
def f1():
    print("Function 1:",a)
def f2():
    a=666
    print("Function 2:",a)
def f3():
    a=555
    print("Function 3:",a)

f1()
f2()
f3()
```

```
Function 1: 100
Function 2: 666
Function 3: 555
```

In [1]:
```
def f2():
    a=768
    print("Function 2:",a)
def f3():
    print("Function 3:",a)

f2()
f3()
```

```
Function 2: 768

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 8
      5     print("Function 3:",a)
      7 f2()
----> 8 f3()

Cell In[1], line 5, in f3()
      4 def f3():
----> 5     print("Function 3:",a)

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

In [2]:
```
a=99  # global variable
def f2():
    a=768   # f() call is gives the priority to local variable
    print("Function 2:",a)
def f3():
    print("Function 3:",a)

f2()
f3()
```

```
Function 2: 768
Function 3: 99
```

In [4]:
```
a=99  # global variable
def f2():
    a=768
    print("Function 2:",a)
def f3():
    print("Function 3:",a)

a=200  # updating global variable --> gglobally

f2()
f3()
```

```
Function 2: 768
Function 3: 200
```

In [5]:
```
# Global - Local variable concept works when the both variable has same identi
```

In [2]:
```
def f2():
    a=768   # f() call is gives the priority to local variable
    print("Function 2:",a)
def f3():
    print("Function 3:",a)
a=200    # global variable
f2()
f3()
```

```
Function 2: 768
Function 3: 200
```

In [1]:
```
def f1():
    a=100
    print("Function 1:",a)
def f2():
    a=987
    print("Function 2:",a)
def f3():
    print("Function 3:",a)

f1()
f2()
f3()  # throws error
```

```
Function 1: 100
Function 2: 987

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 12
     10 f1()
     11 f2()
---> 12 f3()

Cell In[1], line 8, in f3()
      7 def f3():
----> 8     print("Function 3:",a)

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

In [2]:
```
# global variable inside f()
def f1():
    global a
    a=100
    print("Function 1:",a)
def f2():
    print("Function 2:",a)
def f3():
    print("Function 3:",a)

f1()
f2()
f3()
```

```
Function 1: 100
Function 2: 100
Function 3: 100
```

In [3]:
```
# global and local variable inside f()
def f1():
    global a
    a=100
    print("Function 1:",a)
def f2():
    a=125
    print("Function 2:",a)
def f3():
    print("Function 3:",a)

f1()
f2()
f3()
```

```
Function 1: 100
Function 2: 125
Function 3: 100
```

In [5]:
```
# Accesing global variable in presence of local variable inside f()
def f1():
    global a
    a=100
    print("Function 1:",a)
def f2():
    a=125
    print("Function 2:",a)
def f3():
    a=659
    print("Function 3 - Local:",a)
    print("Function 3 - Global:",globals()["a"])  # printing global variable

f1()
f2()
f3()
```

```
Function 1: 100
Function 2: 125
Function 3 - Local: 659
Function 3 - Global: 100
```

In [7]:
```
# updating global variable inside f() in presence of local variable
def f1():
    global a
    a=100
    print("Function 1:",a)
def f2():
    a=125
    print("Function 2:",a)
def f3():
    a=659
    print("Function 3 - Local:",a)
    print("Function 3 - Global:",globals()["a"])
    a=333
    print("Function 3 - Global:",globals()["a"])
    globals()["a"]=414
    print("Function 3 - Global:",globals()["a"])


f1()
f2()
f3()
```

```
Function 1: 100
Function 2: 125
Function 3 - Local: 659
Function 3 - Global: 100
Function 3 - Global: 100
Function 3 - Global: 414
```

In [10]:
```
# updating global variable inside f() in presence of local variable
def f1():
    global a
    a=100
    print("Function 1:",a)
def f3():
    a=655
    globals()["a"]=414
    print("Function 3 - Local:",a)
    print("Function 3 - Global:",globals()["a"])


f1()
f3()
print(a)
```

```
Function 1: 100
Function 3 - Local: 655
Function 3 - Global: 414
414
```

In [12]:
```
# order of f() calling and chane in output
def f1():
    global a
    a=100
    print("Function 1:",a)
def f3():
    a=655
    globals()["a"]=414
    print("Function 3 - Local:",a)
    print("Function 3 - Global:",globals()["a"])


f3()
f1()
print(a)
```

```
Function 3 - Local: 655
Function 3 - Global: 414
Function 1: 100
100
```

Lambda / Anonymous Function

In [13]:
```
def square(n):
    return n*n

print(square(6))
```

```
36
```

In [14]:
```
# (lambda_keyword parameter:operation)(calling)
(lambda n:n*n)(7)
```

Out[14]:
```
49
```

In [15]:
```
(lambda a,b,c:(a+b+c)-(a-b-c))(1,2,3)
```

Out[15]:
```
10
```

In [16]:
```
(lambda a,b:a if a>b else b)(10,12)
```

Out[16]:
```
12
```

In [17]:
```
lambda a,b:a if a>b else b  # definition of lambda f()
```

Out[17]:
```
<function __main__.<lambda>(a, b)>
```

In [18]:
```
# storing lambda f() for calling
a=lambda a,b:a if a>b else b
print(a)
print(a(45,56))
```

```
<function <lambda> at 0x0000022047F172E0>
56
```

In [21]:
```
# lambda f() for variable length argument
x=lambda *arg:print(arg)
x(4,7,2,4,6)
(x)(4,7,2,4,6)
```

```
(4, 7, 2, 4, 6)
(4, 7, 2, 4, 6)
```

In [22]:
```
x=lambda *arg:print(arg)
l=[1,5,4,6,2,3,6,9,8,7]
x(l)  # passing complete list
x(*l) # passing list elements
```

```
([1, 5, 4, 6, 2, 3, 6, 9, 8, 7],)
(1, 5, 4, 6, 2, 3, 6, 9, 8, 7)
```

Programs of Lambda Function

In [30]:
```
# 1) Write a Lambda f() to print even no. in a given list
even=(lambda *t:print([i for i in t if i%2==0]))
l=[1,2,4,5,15,7,8,9,6,54,46,1,2,3,6,5,4]
even(*l)
```

```
[2, 4, 8, 6, 54, 46, 2, 6, 4]
```

In [36]:
```
# 2) write a lambda expression to calculate area of a circle
r=float(input("Enter Radius of Circle: "))
(lambda r,p=3.14:print(f"Area of Circle: {p*r*r}"))(r)
```

```
Enter Radius of Circle: 7
Area of Circle: 153.86
```

In [7]:
```
# 3) write a lambda f() to find HCF of two no.
l=[int(i) for i in input("Enter 2  no. to find HCF by giving space: ").split()
hcf=(lambda *t:[i for i in range(min(t),0,-1) if t[0]%i==0 and t[1]%i==0])(*l)
print(hcf[0])
```

```
Enter 2  no. to find HCF by giving space: 36 27
9
```

In [33]:
```
# 4) write a lambda f() to count words in a given text
cw=(lambda s:print(f"Your text has {len(s.split())} words"))
txt=input("Enter your text to count words: ")
cw(txt)
```

```
Enter your text to count words: nikhil vishwakarma cse ai&ds third sem sistec
gandhi nagar
Your text has 9 words
```

In [ ]: