Chapter – 13 Scopes

Contents

Chapter – 13 Scopes

13.1 Identifier (Name)

Name ဆိုတာ object တွေက ဘယ်သူ ဘယ်ဝါဖြစ်တယ်ဆိုတာကို ဖော်ပြဘို့(identify လုပ်ဘို့) ပေးထားတဲ့ နာမည်ပါ။ Object တွေ ဆိုတာကတော့ variable တွေ၊ string တွေ၊ list တွေ၊ function တွေ၊ method တွေ ပါ။ လူတိုင်းမှာ identify လုပ်ဘို့ နာမည်ကိုယ်စီ ရှိကြသလိုမျိုး Python object တိုင်းမှာလည်း identify လုပ်ဘို့ identifier(name) ကိုယ်စီရှိကြသည်။ ထို့ကြောင့် Python မှာ name တွေကို identifier ဟု ခေါ်သည်။

13.2 Name Space

Namespace ဆိုသည်မှာ program အတွင်းရှိ နာမည်များအားလုံးသည် တစ်ခုနှင့်တစ်ခု မတူသည့် (unique) နာမည်များ ဖြစ်နေအောင်, space တစ်ခုအတွင်း နာမည်တူ(၂)ခု မရှိအောင် ပြုလုပ်ပေးသည့် system တစ်ခု ဖြစ်သည်။

Python တွင် namespace သည် dictionary ကို အခြေခံထားသည်။ Identifier(name) များကို key အဖြစ်နှင့် object များကို value အဖြစ် name-to-object mapping လုပ်သည့်နည်း ဖြစ်သည်။

Python namespace များမှာ local namespace ၊ global namespace နှင့် built-in namespace တို့ ဖြစ်သည်။

13.3 Scope ဆိုတာ

Namespace သည် ပရိဂရမ်အတွင်းရှိ နာမည်အားလုံးကို တစ်ခုနှင့်တစ်ခုမတူသည့်(uniquely identify) နာမည်များ ဖြစ်အောင်ကူညီသည်။ သို့သော် variable name များကို ကြိုက်သလို အသုံးပြုခွင့် ရှိသည်။ Namespace ကဲ့သို့ပင် program တစ်ခုတွင် scope နယ်ပယ်များစွာရှိသည်။

Program တစ်ခုကို execution လုပ်သည့်အချိန်တွင် ရှိနိုင်သည့် scope နယ်ပယ်(၄) မျိုးရှိမှာ

(၁) Local scope (အတွင်းဘက်အကျဆုံး ဖြစ်သည်။)

(၂) Global scope (module level scope ဖြစ်သည်။ Current module ၏ global name များ ပါဝင်သည်။)

(၃) Enclosed (enclosing function များအားလုံးနှင့် သက်ဆိုင်သည်။)

(၄) Built_in (အပြင်ဘက် အကျဆုံး outermost scope ဖြစ်သည်။ Built-in name များ အားလုံး ပါဝင်သည်။)

```
dir()
```

```
['__builtins__', '__doc__', '__loader__', '__name__', '__package__',
'__spec__']
```

dir() function သည် လက်ရှိနယ်ပယ်(current scope)အတွင်းရှိအမည်များ စာရင်းကို ထုတ်ပေးသည်။

Variable တစ်ခုကို define လုပ်ပြီး **dir()** function ခေါ်ကြည့်လျှင် current scope တွင် ထို variable၏ နာမည် `'a_num'` ပါဝင်နေလိမ့်မည်။

```
a_num = 10
dir()
```

```
'__builtins__' .... '__spec__', 'a_num']
```

13.4 Local Scope

Function တစ်ခုအတွင်းမှာ variable တွေကို define လုပ်ထားရင် ၄င်း variable တွေရဲ့ scope က ထို function အတွင်းမှာပဲ တည်ရှိသည်။ Function ပြီးဆုံးသွားရင် function ထဲမှာ define လုပ်ထားတဲ့ variable တွေရဲ့ သက်တမ်းကုန်ဆုံးသွားပြီ(ends the availability) ဖြစ်သည်။

13.5 Global Scope

Module အဆင့်မှာ ရှိတဲ့ variable တွေ သို့မဟုတ် ပရိုဂရမ်ရဲ့က အစနေရာမှ ရေးထားတဲ့ variable တွေက global scope များဖြစ်ကြသည်။ ပရိုဂရမ်တစ်ခုလုံးနှင့် သက်ဆိုင်သည်။ ပရိုဂရမ် ရပ်သည့်အခါမှာ global variable များ၏ သက်တမ်းကုန်ဆုံးသည်။

Local scope အတွင်း၌ ရှိသည့် variable များ အားလုံးကို local variable ဟု ခေါ်သည်။ Local scope အတွင်း၌ ရှိသည့် function များ အားလုံးကို local function ဟု ခေါ်သည်။

Global scope အတွင်း၌ ရှိသည့် variable များ အားလုံးကို global variable ဟုခေါ်သည်။ Global scope အတွင်း၌ ရှိသည့် function များ အားလုံးကို global function ဟုခေါ်သည်။

13.6 Enclosed Scope

Enclosing function များ အတွင်း၌ define လုပ်ထားသည့် name များ အားလုံး ဖြစ်သည်။ Nested function သည် enclosed scope ဖြစ်သည်။ Nested function ဆိုသည်မှာ function တစ်ခုအတွင်း၌ တခြား function များ ပါဝင်နေခြင်း ဖြစ်သည်။ Nested function အတွင်း declare လုပ်ထားသည့် variable များ အားလုံးသည် enclosed variable ဖြစ်သည်။

13.7 Built in scope

Python language အတွင်း၌ ထည့်သွင်းတည်ဆောက်ထားသည့် name နာမည်များအားလုံးသည် built-in function ထဲတွင်ပါဝင်သည်။ special built_in module များထဲတွင် built-in name များပါဝင်သည်။ ဥပမာဖြင့် လေ့လာကြရအောင်

```
def inner():              # function name is inner
    x = 4                 # variable x is define inside inner function
    print ('x is', x)

inner()
```

```
x is 4
```

Variable x ကို inner function အတွင်း၌ define လုပ်ထားသည်။ Variable x သည် local variable ဖြစ်သည်။ Inner function ထဲက ထွက်လိုက်သည်နှင့် တစ်ပြိုင်နက် variable x သက်တမ်းကုန်ဆုံးသွားသည်။ တစ်နည်းအားဖြင့် inner function ၏ အပြင်ဘက်တွင် variable x မတည်ရှိပါ။

Inner function ၏ အပြင်ဘက်တွင် variable x မတည်ရှိပုံကို ဖော်ပြရန် အောက်က ဥပမာကို ဆက်လက်လေ့လာပါ။ Inner function ကို မခေါ်ဘဲ function အပြင်ဘက်တွင် variable x ကို print လုပ်ကြည့် လျှင် `name 'x'` ကို `define` မလုပ်ထားပါဟု NameError ပေးပါလိမ့်မည်။

```
def inner():   # function name is inner
    x = 4      # variable x is define inside inner function
```

```
    print ('x is', x)

print (x)
```

NameError: name 'x' is not defined

X သည် function အတွင်း၌သာ တည်ရှိသောကြောင့် function အပြင်ဘက်၌ သုံး၍ မရနိုင်ပါ။ Access လုပ်၍ မရနိုင်ပါ။

```
y = 10    # inner function ၏ အပြင်ဘက်တွင် variable y ကို define လုပ်သည်။
def inner():    # function name is inner
    x = 4       # variable x is define inside inner function
    print('x is', x)
    print('inside the function y:', x)

print ("y:", y)
inner()
```

y: 10 → သည် print("y:", y) မှ print ထုတ်ပေးသည့် တန်ဖိုးဖြစ်သည်။

x is 4 → inner function မှ print ထုတ်ပေးသည်။

inside the function y: 10 → inner function မှ print ထုတ်ပေးသည်။

Inner function အတွင်း၌ local variable y တန်ဖိုးက 5 ဟု assign လုပ်ပြီး print ထုတ်ကြည့်မည်။

```
y = 10
def inner():    # function name is inner
    x = 4       # variable x is define inside inner function
    y = 5
    print('x is', x)
    print('inside the function y:', y)

print ("y:" , y)
inner()
```

y: 10
x is 4
inside the function y: 5

Global variable y တန်ဖိုးသည် 10 ဖြစ်ပြီး local variable y တန်ဖိုးသည် 5 ဟု print ထုတ်ပေးသည်။

13.8 Modify the Global Variable in Local Scope

“y” ကို function ၏ အပြင်ဘက်တွင် assign (declare) လုပ်ထားပါသည်။ ဥပမာ အဖြစ် “y” ကို local scope အတွင်း သို့မဟုတ် function အတွင်း၌ modify လုပ်ကြရအောင်။

```
y = 10          # variable y ကို function အပြင်ဘက်တွင် assign လုပ်ထားပါသည်။
def inner():    # function name is inner
```

```
    x = 4       # variable x is define inside inner function
    y = y+1     # variable y ကို function အတွင်းဘက်တွင် modify လုပ်မည်။
    print('x is', x)
    print('inside the function y:', y)

print ("y:" , y)
inner()
```

```
UnboundLocalError: local variable 'y' referenced before assignment
```

Global value ကို local scope အတွင်းမှာ ပြောင်း(modify လုပ်)ဘို့ ကြိုးစားသည့်အခါ unbound local error ထုတ်ပေးပါလိမ့်မည်။ Global variable ကို local scope အတွင်းမှာ modify လုပ်လို့မရပါ။ ထိုကဲ့သို့ ပြုလုပ်ချင်ပါက **global** Keyword ကို အသုံးပြုရမည်။ global y ဟု ရေးလိုက်လျှင် အဆင်ပြေသွားပါလိမ့်မည်။

```
y = 10
def inner():      # function name is inner
    x = 4         # variable x is define inside inner function
    global y      # variable y သည် global ဖြစ်ကြောင်း ကြေငြာသည်။
    y = y+1       # variable ကို function အတွင်းဘက်တွင် modify လုပ်မည်။
    print('x is', x)
    print('inside the function y:', y)

print ("y:", y)
inner()
```

```
y: 10
x is 4
inside the function y: 11
```

Y တန်ဖိုးသည် function အပြင်ဘက်တွင် 10 ဖြစ်သည်။ Function အတွင်းဘက်တွင် (၁)ပေါင်းလိုက်သောကြောင့် 11 ဖြစ်သည်။ အခု global y ဆိုသည့် ကုဒ်တစ်ကြောင်းကို function အတွင်း၌ ထည့်ရေးလိုက်သည့်အခါ အဆင်ပြေသွားသည်။

13.9 Enclosed Scope

Function တစ်ခုအတွင်း၌ နောက်ထပ် function တစ်ခု ရှိနေလျှင် အပြင်က outer function သည် **enclosing function** ဖြစ်သည်။ အတွင်းက inner function သည် **nested function** ဖြစ်သည်။ အောက်က ဥပမာတွင် variable z သည် outer function အတွက် local variable ဖြစ်သည်။ inner function အတွက် non-local variable ဖြစ်သည်။ Inner function အတွက် variable z သည် local variable မဟုတ်သလို global variable မဟုတ်ပေ။ ထို့ကဲ့သို့များကို non-local variable ဟုခေါ်သည်။ non-local variable ဆိုသည်မှာ local variable လည်းမဟုတ်သည့် global variable လည်းမဟုတ်သည့် variable တစ်မျိုး ဖြစ်သည်။

```
y = 10                          # → global scope
def outer():                                # enclosing function
```

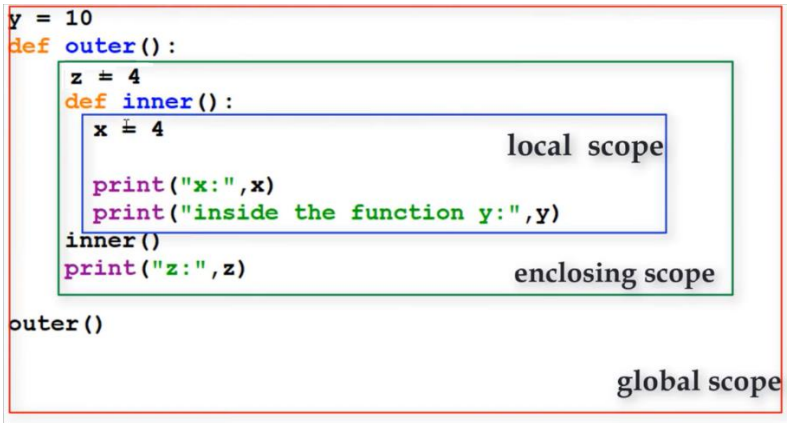
```
    z = 4                           # → enclosed scope
    def inner():                                # nested function
        x = 4                       # → local scope
        print('x is', x)
        print('inside the function y:', y)

    print ("y:" , y)
    inner()
    print("z:" , z)

outer()
```

```
y: 10
x is 4
inside the function y: 10
z: 4
```

အောက်တွင် ပုံနှင့်ဖော်ပြထားသည်။



Variable z ကို outer function ၏ အပြင်ဘက် နေပြီးတော့ access လုပ်လို့ မရသလို inner function အတွင်းကလည်း လှမ်းပြီးတော့ access လုပ်လို့ မရပါ။ နှစ်ဘက်မှာ ပိတ်မိနေသလိုမျိုး ဖြစ်လို့ enclosing scope ဟု ခေါ်ခြင်း ဖြစ်သည်။

Enclosing scope အပြင်ဘက်မှ လှမ်းပြီးတော့ access လုပ်ချင်ရင် **nonlocal** keyword ကို အသုံးပြုရသည်။

Variable z သည် outer function အတွက် local variable ဖြစ်သည်။ Inner function အတွက် non-local variable ဖြစ်သည်။ Inner function အတွက် variable z သည် local variable ဖြစ်သလို global variable မဟုတ်ပေ။ ထို့ကဲ့သို့ variable များကို non-local variable ဟုခေါ်သည်။

Global variable ကို local scope အတွင်း၌ modity လုပ်လိုပါက **global** keyword ကို သုံးရသည်။

Enclosed variable ကို local scope အတွင်း၌ modity လုပ်လိုပါက **nonlocal** keyword ကို သုံးရသည်။

13.10 Build in Scope

Build-in scope တွင် built-in name များ ပါဝင်သည်။ ကြိုက်သည့်နေရာက ကြိုက်သလို အသုံးပြုနိုင်သည်။ ဘာမှ define လုပ်စရာ မလိုပါ။ တိုက်ရိုက်အသုံးပြုနိုင်သည်။ **print()** ဆိုသည့် function ကို define လုပ်စရာမလိုပါ။ ရောက်သည့် နေရာအတွင် အသုံးပြုခွင့်ရှိသည်။

အောက်က ကုဒ် print('x is', x) တွင် x တန်ဖိုးသည် မည်မျှ ဖြစ်မည်နည်း?

```
x = 5
def my_function():
    x = 10
    def inner():
        x = 15
        print('x is', x)
    inner()

my_function()
```

```
x is 15
```

ထိုကဲ့သို့ variable x သည် နေရာတိုင်းတွင် ရှိနေသည့်အခါ မည်သည့် x ကို အသုံးပြုရမည်နည်း?

13.11 LEGB rules

L → Local scope

E → Enclosed scope

G → Global scope

B → Built in scope

Python သည် variable တစ်ခုကို အသုံးပြုရန် ညွှန်ကြားချက်(instruction)ရသည့်နှင့် တစ်ပြိုင်နက် ပထမဦးစွာ local scope အတွင်း၌ လိုက်ရှာမည်။ မတွေ့လျှင် enclosed scope အတွင်း၌ လိုက်ရှာမည်။ မတွေ့လျှင် global scope အတွင်း၌ လိုက်ရှာမည်။ မတွေ့လျှင် built in scope တွင် ရှာမည်။ ထိုအားလုံးတွင် မတွေ့လျှင် NameError ပေးလိမ့်မည်။

-End-