When we create a variable in Python that name stored in a namespace. So, what is namespace in Python ??

A namespace is a system to have a unique name for each and every object in Python. An object might be a variable or a method.

Let's take an example of the Unix/Linux file system, we can create multiple directories and create multiple files under these directories. Also, it is possible, to have same file name under different directory.

/home/devops/dir1 → fileA , fileB

/home/devops/dir2 → fileA, fileC

To get directed to a file, we have to provide the absolute path for the file, such as:

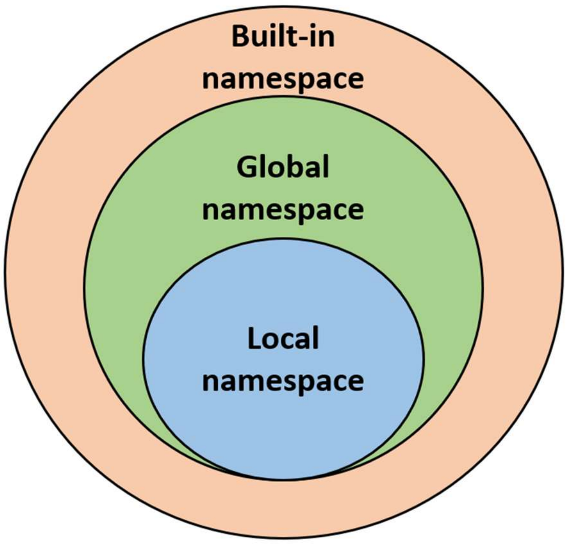
/home/devops/dir1/fileA and /home/devops/dir2/fileA

On the similar lines, Python interpreter understands what exact method or variable one is trying to point to in the code, depending upon the namespace.

Namespace = Name + space
Name: Unique Identifier
Space: Scope

Here, a name might be of any Python method or variable and space depends upon the location from where is trying to access a variable or a method.

Types of Namespace:
1: Built-in-namespace: Like print() function, that is available with Python installation.
2: Global-namespace: When user creates a module
3: Local namespace: Create of local function



Type of Namespaces

A lifetime of a namespace depends upon the scope of objects, if the scope of an object ends, the lifetime of that namespace comes to an end.

Example1: What will be the value of x?

```
x = 100

def my_func():
    x = 10
    return x
```

```
print(x)
```

```
100
```

```
print(my_func())
```

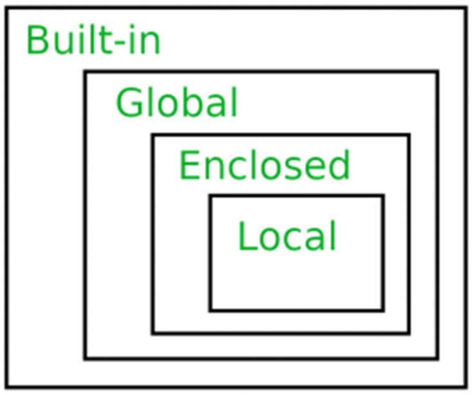
```
10
```

How does Python know which x I am referring here?
To understand this, let's first understand the LEGB rule that Python follows to assign values to these variables.

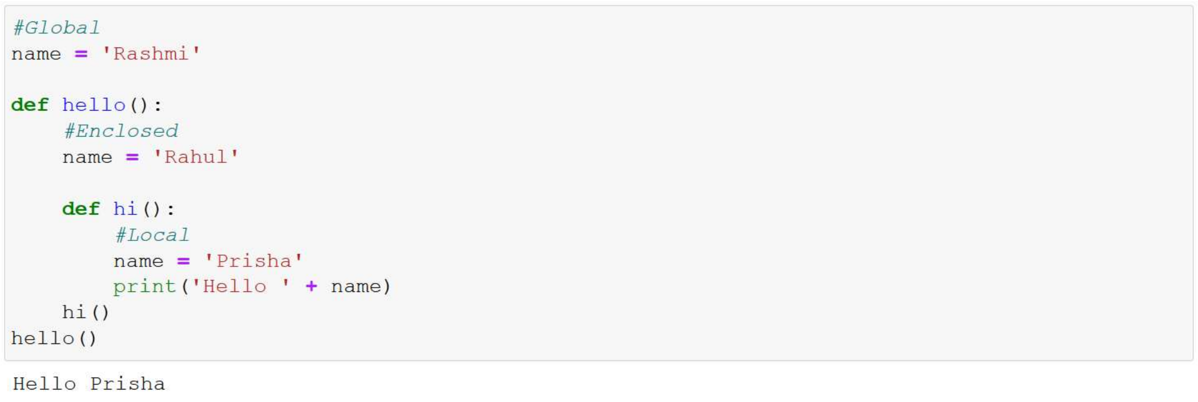
Ok, so far we understand that Namespace is a container where names are mapped to objects like modules, functions and classes etc. Every namespace has its scope.

In Python, the LEGB rule is used to decide the order in which the namespaces are to be searched for scope resolution. The scopes are listed below:

Local(L): Defined inside function/class
Enclosed(E): Defined inside enclosing functions (Nested function concept)
Global(G): Defined at the uppermost level
Built-in(B): Reserved names in Python built-in modules



Let’s understand this with an example



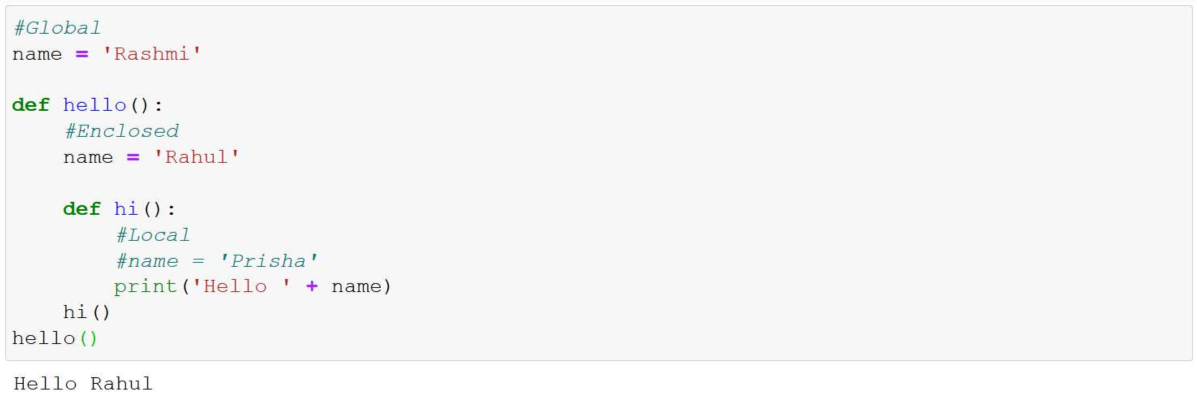
```
#Global
name = 'Rashmi'

def hello():
    #Enclosed
    name = 'Rahul'

    def hi():
        #Local
        name = 'Prisha'
        print('Hello ' + name)
    hi()
hello()
```

```
Hello Prisha
```

As you can see the local variable get the precedence:



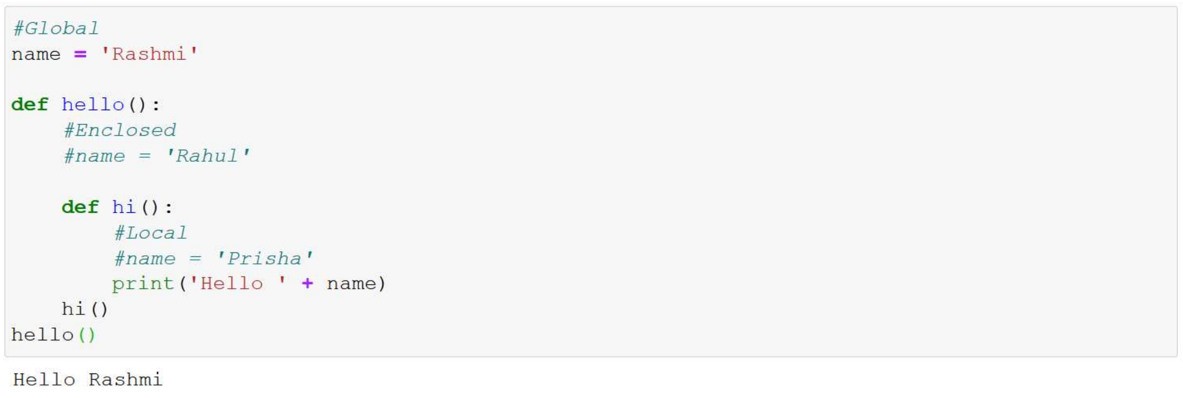
```
#Global
name = 'Rashmi'

def hello():
    #Enclosed
    name = 'Rahul'

    def hi():
        #Local
        #name = 'Prisha'
        print('Hello ' + name)
    hi()
hello()
```

```
Hello Rahul
```

As I have commented the local, next it moves to Enclosed one.



```
#Global
name = 'Rashmi'

def hello():
    #Enclosed
    #name = 'Rahul'

    def hi():
        #Local
        #name = 'Prisha'
        print('Hello ' + name)
    hi()
hello()
```

```
Hello Rashmi
```

Here it picks Global