Namespaces and Scope in Python

What is namespace:

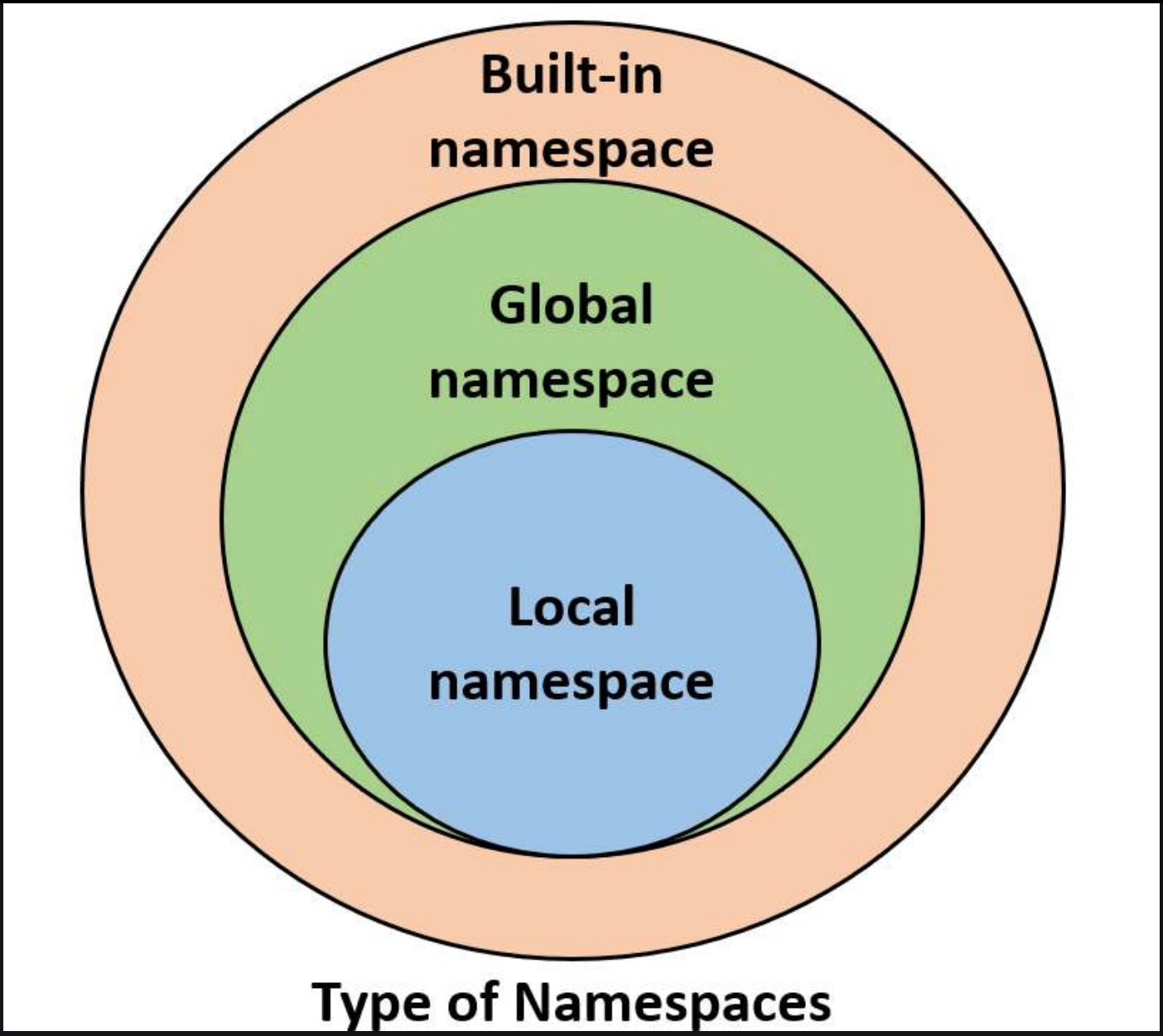
A namespace is a system that has a unique name for each and every object in Python. An object might be a variable or a method. Python itself maintains a namespace in the form of a Python dictionary. Let's go through an example, a directory-file system structure in computers. Needless to say, that one can have multiple directories having a file with the same name inside every directory. But one can get directed to the file, one wishes, just by specifying the absolute path to the file.

Real-time example, the role of a namespace is like a surname. One might not find a single "Alice" in the class there might be multiple "Alice" but when you particularly ask for "Alice Lee" or "Alice Clark" (with a surname), there will be only one (time being don't think of both first name and surname are same for multiple students).

On similar lines, the Python interpreter understands what exact method or variable one is trying to point to in the code, depending upon the namespace. So, the division of the word itself gives a little more information. Its **Name** (which means name, a unique identifier) + **Space**(which talks something related to 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 namespaces :

When Python interpreter runs solely without any user-defined modules, methods, classes, etc. Some functions like print(), id() are always present, these are built-in namespaces. When a user creates a module, a global namespace gets created, later the creation of local functions creates the local namespace. The **built-in namespace** encompasses the **global namespace** and the global namespace encompasses the **local namespace**.



The lifetime of a namespace :

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. Hence, it is not possible to access the inner namespace's objects from an outer namespace.

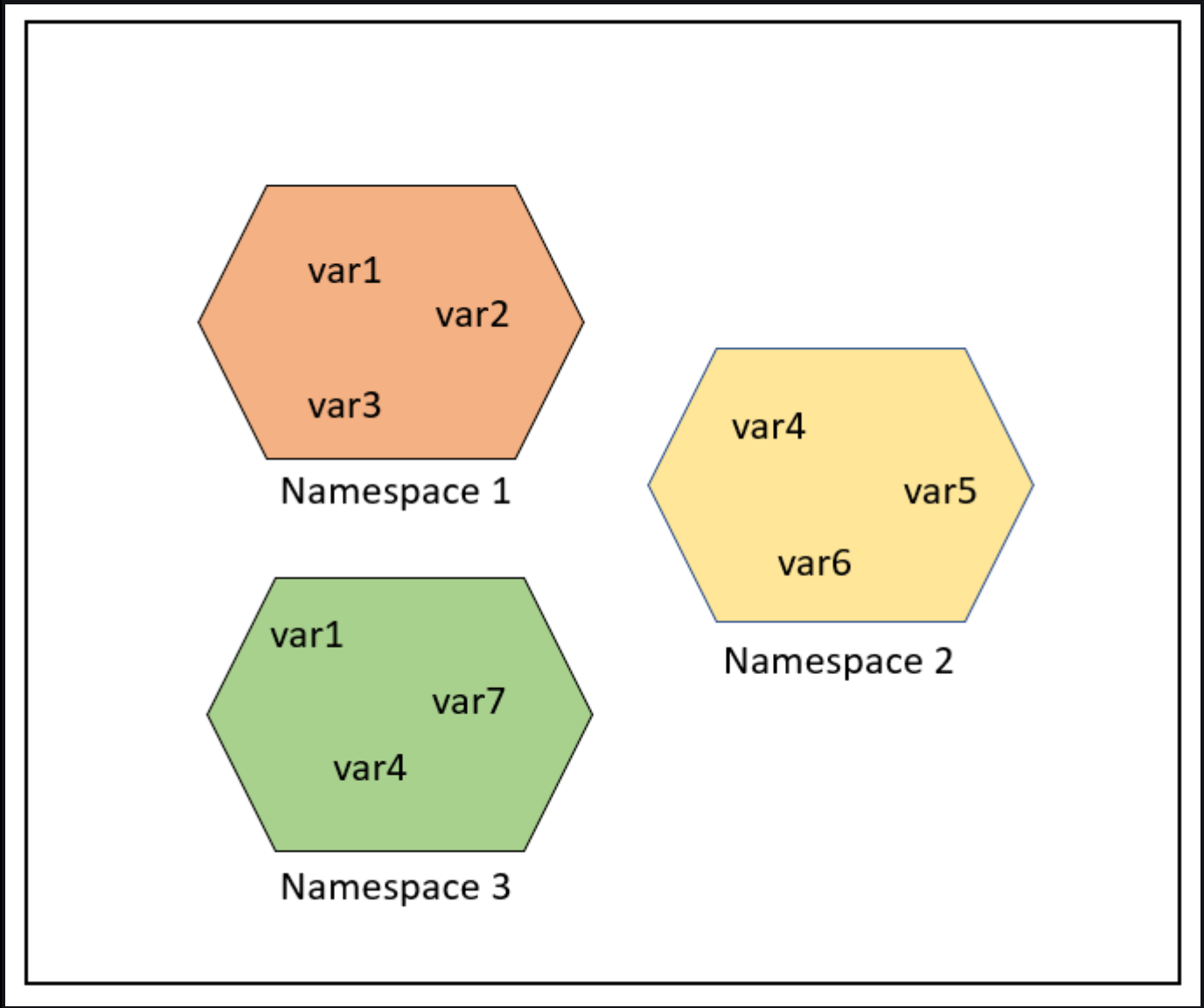
Example:

- Python3

```
# var1 is in the global namespace

var1 = 5

def some_func():
```

```
    # var2 is in the local namespace
    var2 = 6
    def some_inner_func():


        # var3 is in the nested local
        # namespace
        var3 = 7
```

As shown in the following figure, the same object name can be present in multiple namespaces as isolation between the same name is maintained by their namespace.



But in some cases, one might be interested in updating or processing global variables only, as shown in the following example, one should mark it explicitly as global and the update or process. Note that the line “count = count +1” references the global variable and therefore uses the global variable, but compare this to the same line written “count = 1”. Then the line “global count” is absolutely needed according to scope rules.

- Python3

```
# Python program processing
# global variable


count = 5
def some_method():
    global count
    count = count + 1
    print(count)
some_method()
```

Output:

6

Scope of Objects in Python :

Scope refers to the coding region from which a particular Python object is accessible. Hence one cannot access any particular object from anywhere from the code, the accessing has to be allowed by the scope of the object.
Let's take an example to have a detailed understanding of the same:

Example 1:

- Python3

```
# Python program showing
# a scope of object


def some_func():
    print("Inside some_func")
    def some_inner_func():
        var = 10
        print("Inside inner function, value of var:",var)
    some_inner_func()
    print("Try printing var from outer function: ",var)
some_func()
```

Output:

```
Inside some_func
Inside inner function, value of var: 10

Traceback (most recent call last):
  File "/home/1eb47bb3eac2fa36d6bfe5d349dfcb84.py", line 8, in
    some_func()
  File "/home/1eb47bb3eac2fa36d6bfe5d349dfcb84.py", line 7, in some_func
    print("Try printing var from outer function: ",var)
NameError: name 'var' is not defined
```