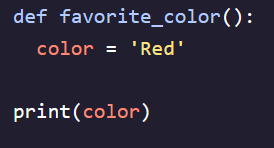
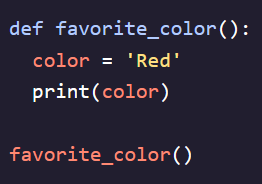
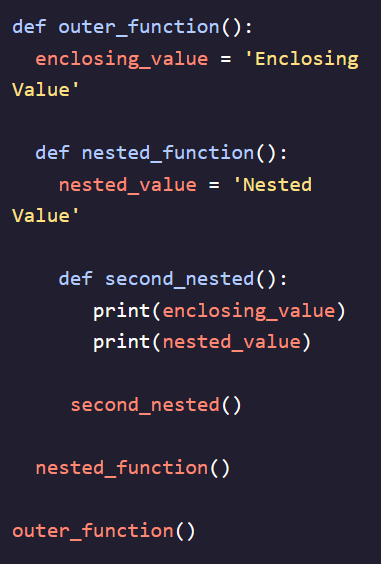
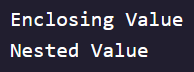
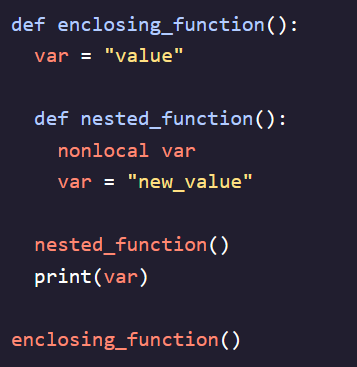
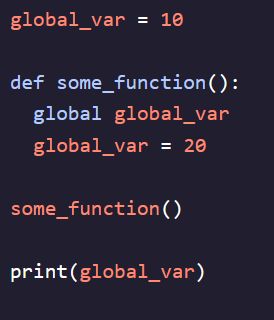
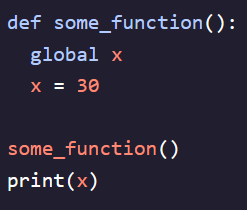
**Local Scope:**

- *scope* – Refers to areas where variables are visible and accessible, defines which namespaces a program will search within and in what order.  
- When we call a function a local scope is generated, each subsequent function call generates a new local scope  
- Names in a local scope cannot be accessed or modified by any code called in outer scopes  
- The code on left returns an error because *print()* is trying to access a local scope  
- The code on the right works properly because *print()* is now nested within the local scope  
 

**Enclosing/Nonlocal Scope:**

- Special rules apply for accessing nested values within nested functions  
- Enclosing scope allows any value defined in an enclosing function (top level) to be accessed in nested functions below it  
- Only works from the bottom up, top most function does **NOT** have access to bottom most variables  
- Immutable objects can be accessed in nested functions but cannot be modified   
- If you use the *nonlocal* statement you can tell Python that you want to access a nonlocal version of the variable instead of the local one in the nested function  
  

**Modifying Global Scope:**

- Similarly to local scope, global scope assignments cannot be modified from within a nested function  
- Using the *global* statement you could modify a global statement from within a nested function  
- Can even be used if variable name has not yet been defined as *global* will create a new variable in the global namespace  
 

**Scope Resolution**:

- A term used to describe the search procedure for a name in the various namespaces determined by a set of rules that dictate the order the search needs to follow  
- **LEGB** (**L**ocal, **E**nclosing, **G**lobal, **B**uilt-In)– the unofficial rule dictating the search parameters   
- The order that Python searches in always begins at the lowest level and flows upwards to higher scopes   
- In the example below, Python begins looking for the variable name at the local scope level (inner\_func()), since it doesn’t find it there it moves to the Enclosing Scope (func(), since it doesn’t find it there either it moves to the Global Scope which is where age is  
- In the second example, Python finds the “age = 42” variable first, stops looking for more, and outputs that. To fix this you would need to add “global age” before the age variable to make the global scope variable mutable  
