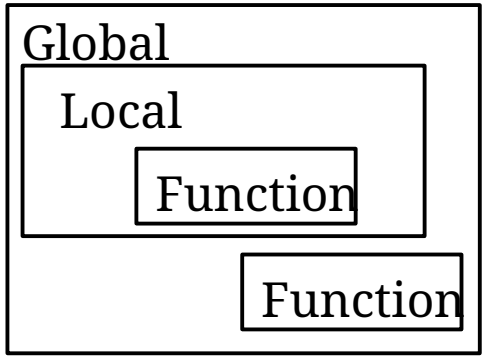
Scope of variables

In general you can distinguish between

Global
Local
Function

Scope



Bash (like Python) doesn't have block scope in conditionals.

It has local scope within functions, it is also possible to use the ‘local’ modifier which is a keyword to declare the local variables.
Local variables are visible only within the block of code.

Variable scope (visibility) is related mainly to the shell.
Exported variables are visible in all subshells.

Scope of variables

A variable exported is a global variable.

A variable defined in the main body of the script is called a local variable.
- It will be visible throughout the script,
- A variable which is defined inside a function is local to that function.
- It is accessible from the point at which it is defined until the end of the function, and exists for as long as the function is executing.

- Global variables can have unintended consequences because of their wide-ranging effects: we should almost never use them

Exercise: Scope of variables

```
#!/bin/bash
e=2
echo At beginning e = $e
function test1() {
  e=4
  echo "hello. Now in the function1 e = $e"
}
function test2() {
  local e=4
  echo "hello. Now in the function2 e = $e"
}
test1
echo "After calling the function1 e = $e"

e=2
echo In the file before to call func2 reassign e = $e
test2
echo "After calling the function2 e = $e"
```

Justify the result !