PRINCIPLES OF PROGRAMMING LANGUAGES

Scope and Bindings

Prafullata Kiran Auradkar

Computer Science and Engineering

PRINCIPLES OF PROGRAMMING LANGUAGES

Variable – Scope and Bindings

Prafullata Kiran Auradkar

Computer Science and Engineering

Scope - Bindings

- Scope – is the extent of statements in a program where the variable is accessible
- Lifetime is the time duration in which the variable is alive or accessible.

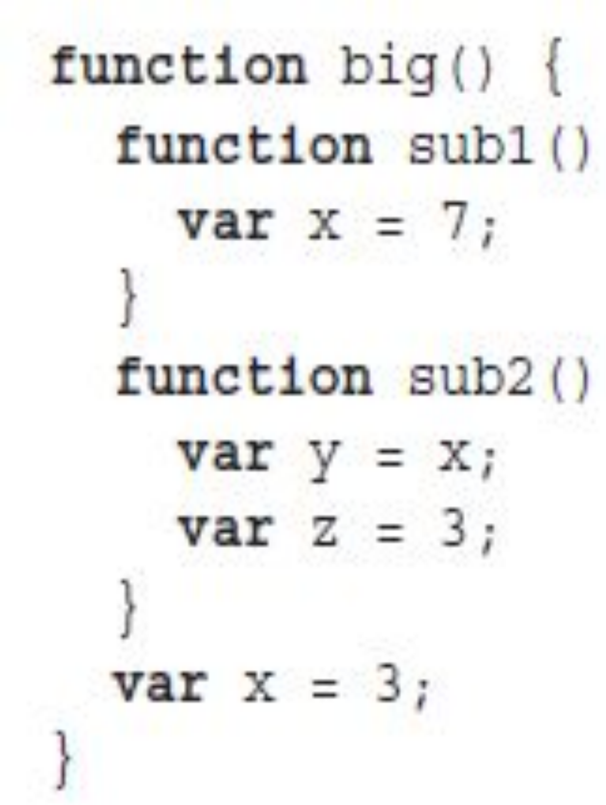
- Are these SAME?

Scoping Rules - Static scoping

- **Static scoping**
  - C scope rules (Check with blk.c)
  - C++ scope rules (check with blk.cpp)
  - Compare with blk.py
  - What about Java?
- Is determined at compile time,
- Easy to manage and runtime efficiency is high as binding of access to the definition is resolved before runtime.

Scoping Rules - Dynamic scoping

- Based on calling sequences of program units, not their textual layout (temporal versus spatial)



```
function big() {
  function sub1() {
    var x = 7;
  }
  function sub2() {
    var y = x;
    var z = 3;
  }
  var x = 3;
}
```

Scoping Rules - Dynamic scoping

- References to variables are connected to declarations by searching back through the chain of subprogram calls that forced execution to this point

(**Perl** variable scopes can be restricted to lexical scope or dynamic scope – using **my** or **local** keyword usage.

Ex: Scope1.pl)

Static Scoping Vs. Dynamic Scoping

- Variable access availability is more than what is needed.
- Designers are encouraged to use more No. of global variables, than needed.
- Reliability is higher as type checks can happen statically.
- Time taken for the non local access is lesser

- No information available on the access of variables at compile time.
- Less reliable due to extended use of variables in the calling sequence.
- No static type checking is possible
- Difficult to read
- Time taken for non local access is higher.
- As the local variables of calling function are available to the called function, there is no need of passing the parameters.

Referencing Environments :

The ***referencing environment*** of a statement is the collection of all names that are visible in the statement

In a **static-scoped language**, it is the local variables plus all of the visible variables in all of the enclosing scopes

A subprogram is **active** if its execution has begun but has not yet terminated

In a **dynamic-scoped language**, the referencing environment is the local variables plus all visible variables in all active subprograms

Do it Yourself

Python Example

```
 g=3

def sub1():
    a=5
    b=7                    ----------(1)
   def sub2():
        global g
        c=9                ------------(2)
        def sub3():
            nonlocal c
            g=11                   --------------------(3)
```

THANK YOU

Prafullata Kiran Auradkar
Computer Science and Engineering

prafullatak@pes.edu