**Def:** A **definition** is anything that establishes a possible binding to a name.

**Def:** **Scope** is a programming language tool to limit the visibility of definitions.

**Example:** Early dialects of Basic did not have **scoping rules**, all definitions of all variables were visible in the **global scope**.

```
$A = "This is a global string"
...
Function Foo ()
   $A = "This is a local string"
   ...
End
Call Foo ...
$B = $A
```

What is the content of $B?

**Problem:** When everything is visible everywhere then it is up to the programmer to control visibility of definitions (e.g. globally unique names).
Def: A namespace is a zone in a programming language which is populated by names. In a namespace, each name must be unique.

The most common namespace in programming languages is the block.
Scoping with Blocks

Def: A block is any language construct that contains definitions and delineates the region of the program where those definitions apply.

Example: Java

```java
if (cond) {
    int q = ...;
}
else {
    int r = ...;
}
```

Example: ML

```ml
let val q = ...;
in ...
end
```
Nested Blocks

In most modern programming languages blocks can be nested.

Example: Java

```java
if (cond) {
    bool q = ...;
    while (q) {
        int r = ...;
    }
}
```
This can lead to interesting anomalies, consider;

**Example:** ML

```
let
  val n = 1
in
  let
    val n = 2
  in
    n
  end
end;
```

What is the value of this expression?
Implicit Blocks

Def: A block is any language construct that contains definitions and delineates the region of the program where those definitions apply.

Blocks can also be defined implicitly by some language construct.

Example: ML

- fun add (a,b) = a + b;
**Def:** A labeled namespace is any language construct that contains definitions and delineates a region of the program where those definitions apply; and also have a name that can be used to access those definitions.

**Example: Java**

```java
class MyInt {
    public static int min = -32000;
    public static int max = 32000;
}
```

to access definitions in the labeled namespace:

```java
int i = MyInt.min;
```

**Other labeled namespaces:**
- Java: packages, class
- C++: class, namespace
- C: struct
- ML: structure
Def: A primitive namespace is a language construct that contains definitions and delineates a region of the program where those definitions apply; but the region was defined at language design time (similar to primitive data types, you can use them but not define them).

Most modern programming languages define two primitive namespaces – one for user defined variable names and one for type names (both primitive and constructed).
Primitive Namespaces

**Example: ML**
- `val int = 3;`
- `val int = 3 : int`

**Example: Java**
```
class Foo { ... };  
myFunc (Foo Foo) {  
    Foo g = Foo;  
}
```

**Observation:** Because of the primitive namespaces modern programming languages never get confused about whether a name is a type or a variable – they simply look up the name in the corresponding primitive namespace.