Using Subroutines & Functions

GEO 590

Tim Sennott

Subroutines

- Procedures with a list of instructions that carry out a task
- Can be thought of as a miniature program
- One subroutine can call another subroutine
- Runs when it is "called" by another procedure as opposed to being triggered by an "event"

Subroutine Example

```
Sub TestSub()
   MsgBox "Code in TestSub()"
End Sub

Private Sub Form_Load()
   MsgBox "Code in Form_Load()"
   TestSub
   MsgBox "Back in Form_Load()"
End Sub
```

Subroutine Example Using “Call” keyword

```
Sub TestSub()
   MsgBox "Code in TestSub()"
End Sub

Private Sub Form_Load()
   MsgBox "Code in Form_Load()"
   'This line is functionally equal as the line in the
   previous example
   Call TestSub
   MsgBox "Back in Form_Load()"
End Sub
```

Why Use Subroutines ?

- Organization of code into separate blocks allows for reuse in other modules
- Better organization of code
- Ease of de-bugging smaller blocks of code

Subroutine Scope

- "Private" Sub - can only be called from the source file from where they were defined
- "Public" Sub - can be called from anywhere in your program.
- subroutine with a Private scope in a form can not be called from another form, whereas it can if it has a Public scope
- should declare the subroutine to be Private if you know that your subroutine should never be called from a different source file

Functions

- Subroutines that return a value
- Message boxes, Input boxes, etc
- Uses the "Function" keyword in place of the "Sub" keyword

Functions

- Keyword "Call" not used as in Subroutines
- Instead two lines of code:
    1. Declare Variable
    2. Sets variable = Function Name

Ex. To call Function "Test":

```
Dim x As Integer
x = Test
```

Function Example

```
Private Function Add(ByVal x As Integer, ByVal y As Integer) As
    Integer
    Dim Res as integer
    Res = x + y
    Add = Res
End Function

Private Sub Form_Load()
    Dim a As Intege
    Dim b As Integer
    Dim c As Integer
    a = 32
    b = 64
    c = Add(a, b)
    MsgBox ("Sum is : " & c)
End Sub
```

Sources:

Programming ArcObjects with VBA
A Task Oriented Approach
Kang-Tsung Chang

Getting to Know ArcObjects
Programming ArcGIS with VBA
Robert Burke