Housekeeping

- Grace is gone this week, but
  - Her labs (AC and AD) will meet as usual
  - Her Monday morning (9am) office hours have been moved to 2:30-4:30pm
- Project 2B is due Wednesday
- Project 3 will be handed out on Wednesday
- The next quiz is this Friday (covering FIT 11-15)

Iteration: Once Is Not Enough

People don’t usually like to repeat themselves, but in computers repetition is one of the most valuable things a program can do. Computers can repeat steps systematically without tiring. If program instructions are to be performed more than once, the computer can be programmed to repeat instructions without the programmer explicitly writing them out each time.

The Idea of Iteration

- Concept: Iteration is the repeated execution of a series of statements in programming
- There are two key components to iteration:
  - The repetition of a bunch of steps...
  - A way to stop the repetition at some point and continue with the rest of the program
- To perform iteration, programming languages include special statements often called iteration statements

Key Components of Iteration

- Iteration Component # 1
  - The statements that will be repeated are called the loop body
- Iteration Component # 2
  - A test specifying when the repetition stops is called the stop condition
- In addition to the components above, loops typically have at least one variable that is explicitly changed "inside" the loop – this is called the iteration variable
- When the iteration variable contains a certain value (defined by the program), then the loop stops
- Some value must change at some point between consecutive iterations, or else the loop will never terminate… it is an infinite loop
Programming languages, like VB 6, usually have more than one form of iteration as part of their notation. But we will only study one of them here – the Do-While loop.

```
Do While <stop condition>
<code statements>
Loop
```

The meaning is as follows:
- The stop condition is tested. If it is false, all the statements are skipped. Execution of the code continues at the point just after the Loop statement.
- If the stop condition is true, the code statements are performed once.
- The stop condition is tested again. If it is false the loop is over and the code statements are skipped; code execution continues after the Loop statement.
- If the stop condition is true, the code statements are performed a second time.
- ...
Exercise #2
What does this code print?

Dim i As Integer
i = 10
Do While i <= 4
    Print i
    i = i + 1
Loop

It doesn't print anything!

Exercise #3
What does this code print?

Dim i As Integer
i = 1
Do While i >= 1
    Print i
    i = i + 1
Loop

This is an infinite loop!

Hmmmm, How Is It Done?
Think about writing a program to do the following:

10 seconds
9 seconds
8 seconds
7 seconds
6 seconds
5 seconds
4 seconds
3 seconds
2 seconds
1 second
Blast Off!!!!!!

Blastoff
Private Sub Form_Click()
    Dim x As Integer
    x = 10
    Do While x > 0
        Print x & " seconds"
        x = x - 1
    Loop
    Print "Blastoff!!!!!!"
End Sub
**Blastoff**

```
Private Sub Form_Click()
    Dim x As Integer
    x = 10
    Do While x > 0
        Print x & " seconds."
        Call Pause()
        x = x - 1
    Loop
    Print "Blastoff"
End Sub
```

```
Private Sub Pause()
    Dim i As Integer, j As Integer
    i = 0
    Do While i < 1000
        j = 0
        Do While j < 100
            frmBlastoff.Visible = True
            j = j + 1
        Loop
        i = i + 1
    Loop
End Sub
```

---

**A Little More About Infinite Loops**

- If you don't properly change your iteration variable—so that the stop condition eventually evaluates to false—then you will never exit the loop.
- This is called an infinite loop.
- The only way out of the infinite loop is by stopping the program from outside of the program itself.
- In VB 6, press the CTRL + Break keys to end an infinite loop.

---

**Random Numbers**

- CONCEPT: Random numbers are numbers that are independent or unrelated to each other.
  - Coin flipping can produce random bits... heads (0), tails (1)
  - Rolling a die can produce random digits 1 through 6
  - Drawing cards from a shuffled deck can produce
    - Random bits... red or black
    - Random digits... 1 through 4 (Suit)
    - Random digits... 1 through 13 (Value)

- Rnd is VB6’s random number generator.

---

**Using Rnd**

- Rnd gives a double between 0 and 1
  - 0.467827363

- To choose randomly among x number of things, multiply Rnd by x and truncate... the result is a random integer between 0 and x-1
  - Pick among 255 things:
    - Rnd * 255 -> 0.467827363 * 255 -> 199.295977565
    - Rnd * 255 -> 0.747238767 * 255 -> 190.545885585

  - To truncate, or convert to Integer, the expression above:
    - Int (Rnd * 255)
### Using Rnd

- This procedure will produce a random number between 1 and range:

```vbc
Private Sub randomNumber(range As Integer, result As Integer)
    result = Int(range * Rnd) + 1
End Sub
```

- After executing

```vbc
Call randomNumber(4, x)
```

x will be an integer between 1 and 4

### Is Random Really, Truly Random?

- What is randomness?
  - Generation of some value or thing that is NOT predictable

- But, computers are deterministic – they exactly follow instructions and do exactly what is asked...how can they do something random?

- CONCEPT: Rnd is “Pseudo-random”, a deterministic computation that produces numbers that appear to be random and pass standard tests for randomness

- Random numbers generated by a computer can be thought of as a very, very long list of values. The computer always returns the values in the same order, but it doesn’t always start at the same place in the list. If the list contained a million values, sometimes the computer would return values starting at the first value, but sometimes it might start at the 105,768th value. Because the starting point is unknown and the list is so long, a repeatable pattern is not discernable.

### Initializing Rnd

- Use `Randomize` to initialize VB’s random number generator. This is called seeding.

- To avoid generating the same sequence of psuedo-random values, call `Randomize` before you call `Rnd`; but this only has to be done once. A good place to seed the psuedo-random number generator is in the Form’s Load event:

```vbc
Private Sub Form_Load() ' Load event
    Randomize
End Sub
```

- If `Randomize` is not called before `Rnd` is used, the same series of psuedo-random values will be returned.

### Experiment with Rnd

```vbc
Private Sub spin(choices as Integer, result as Integer)
    result = Int(choices * Rnd)
End Sub
```

```vbc
Private Sub Form_Click()
    Call experiment
End Sub
Private Sub experiment()
    Dim roll As Integer
    Dim outcome0 As String
    Dim outcome1 As String
    Dim outcome2 As String
    Dim outcome3 As String
    Dim reps As Integer
    Dim endResult As String
    reps = 0
    outcome0 = "0 : "
    outcome1 = "1 : "
    outcome2 = "2 : "
    outcome3 = "3 : "
    Do While reps < 100
        Call spin(4, roll)
        If roll = 0 Then
            outcome0 = outcome0 & "*"
        ElseIf roll = 1 Then
            outcome1 = outcome1 & "*"
        ElseIf roll = 2 Then
            outcome2 = outcome2 & "*"
        ElseIf roll = 3 Then
            outcome3 = outcome3 & "*"
        End If
        reps = reps + 1
    Loop
    lblBar0.Caption = outcome0
    lblBar1.Caption = outcome1
    lblBar2.Caption = outcome2
    lblBar3.Caption = outcome3
End Sub
```
Experiment with Rnd

Private Sub cmdPick_Click()
    Dim npicks As Integer, curpick As Integer
    Dim cardsuit As Integer, cardnumber As Integer
    Dim cardname As String
    Dim resultstring As String
    npicks = txtncards.Text
    curpick = 0
    resultstring = "My picks are:
    Do While curpick < npicks
        Call randomNumber(4, cardsuit)
        Call randomNumber(13, cardnumber)
        Call getCardName(cardsuit, cardnumber, cardname)
        resultstring = resultstring & vbCrLf & cardname
        lblResults.Caption = resultstring
        lblResults.Visible = True
        curpick = curpick + 1
    Loop
    End Sub

Private Sub randomNumber(range As Integer, result As Integer)
    result = Int(range * Rnd) + 1
End Sub

Private Sub getCardName(suit As Integer, number As Integer, name As String)
    If number = 1 Then
        name = "Ace"
    ElseIf number <= 10 Then
        name = number
    ElseIf number = 11 Then
        name = "Jack"
    ElseIf number = 12 Then
        name = "Queen"
    ElseIf number = 13 Then
        name = "King"
    End If
    name = name & " of ">
    If suit = 1 Then
        name = name & "Spades"
    ElseIf suit = 2 Then
        name = name & "Clubs"
    ElseIf suit = 3 Then
        name = name & "Hearts"
    ElseIf suit = 4 Then
        name = name & "Diamonds"
    End If
End Sub

Summary

Iteration is very useful when you want the program to repeat a sequence of steps.
Iteration requires 2 components:
- Loop body – the steps to be repeated
- Stop Condition – a way to exit the loop
When the Loop ends, the execution of code continues at the point where the Loop ended.
You have been introduced to one iterations statement, the Do-While Loop, but there are many.
With Conditional(If-Then) and Iteration (Do-While) you can accomplish almost any programming needed.
Computers generate random numbers using "pseudo random" techniques.
Random numbers are handy for making computers less boring.
You will LOVE this function for Project 3!