Course: <http://www.homeandlearn.org/index.html>

Object

Property

Method

Types

# Getting Started

* An object is the thing you’re trying to manipulate (such as a worksheet)
* A method is the thing that you want to do with the object/what you want to manipulate
* A property, is literally a property of an object
* Methods come with parameters, syntax is weird tho
  + Ex: tv.buy PaymentType:=Cash
    - tv is the object
    - buy is method
    - PaymentType is a parameter
    - Cash is the value of the parameter
  + Ex: tv.buy PaymentType:=Cash Discount:=No DeliveryCharge:=No
    - Multiple parameters – PaymentType, Discount
* ActiveCell is an object where your cursor currently is
  + You can point to another cell by using Row and Column numbers typed between round Brackets
    - ActiveCell(1,1) points to the current ActiveCell
    - Ex ActiveCell(1,0) points to the cell to the left of the ActiveCell
* Range is a property
  + Syntax: Range(“A1:B7”)
    - First cell reference is top-let, bottom right is the second cell
* .Select is a method
* Worksheets is an object
  + Syntax: Worksheets(“Sheet1”) or Worksheets(1)
* Offset is a property which shifts your current selection to a new location
  + Used with Range property and ends with .Select
  + Syntax: .Offset(RowOffSet:=1, ColumnOffset:=1) or .Offset(1,1)
* Resize is a property which selects a range of cells based on a starting cell
  + Used with Range property and ends with .Select
  + Syntax: .Resize(RowResize:=1, ColumnResize:=1) or .Resize(1,1)
    - Very similar to offset

# VBA Programming Variables

* Variables are usually declared with the “Dim” keyword
* Integer type: As Integer
  + Can only stores integers values from -32 768 to 32 767
* You can’t declare and define a variable in one line, you must first declare, then assign
  + You can’t do Dim MyNumber As Integer = 10
  + You must
    - Dim MyNumber As Integer
    - MyNumber = 10
* Value is a property of Range
  + Can be used to get or set value of cell(s)
* If you don’t do the whole “Dim … As Integer” thingy, but still do the “MyNumber = 10”, VBA automatically makes the type of MyNumber as a variant – will technically still work
  + But bad practice and will be extremely slow
* Variants can be any type that the program chooses
* You can put an “Option Explicit” at the top of the program, and it’ll prevent any variables without proper declarations (i.e. using “Dim” and specifying the type) to be set-up
  + When you put Option Explicit into your window – there will be a new section above that will allow you to declare globally-scoped variables
* Mathematical operations generally still follow the order of operations, and goes from left to right
  + Use brackets if you want to clarify stuff
  + / is integer division whereas \ is regular division -> pay attention the variable type (ie integer)
* Other Types
  + As Long – can hold integer numbers up to values from -2,147, 483, 648 to 2,147, 483, 647
  + As Single – holds 4 bytes of data, can be decimal values
  + As Double – holds 8 bytes of data, can be decimal values
  + As Currency
  + As String

# Conditional Logic

* Syntax:
  + If Condition\_To\_Test Then

'CODE HERE

ElseIf (Else + If) Condition\_To\_Test Then

'CODE HERE

Else

'CODE HERE

End If

* Conditional operators

|  |  |
| --- | --- |
| Operator | Meaning |
| = | Equal |
| < | Less than |
| > | Greater than |
| <= | Less than or equal to |
| >= | Greater than or equal to |
| <> | Not equal to |
| Not | Test if value is NOT something |
| AND | Test for more than one condition |
| OR | Test if the value is either or something |
| XOR | Test if one and ONLY one value is true |

* To join a variable into a string, you use **&** 
  + Ex: MsgBox MyNumber & “is less than 20”
* Boolean type
  + Value can be True or False
* “Not” syntax
  + If Not Condition\_To\_Test
* “And” syntax
  + If Condition\_To\_Test And Condition\_To\_Test
  + Same thing with “Or” operator
* Nested conditionals are also a thing you can use
* Another property is Interior.Color and Interior.ColorIndex
  + For Interior.Color you set it equal to an RGB Color – (RED, GREEN, BLUE)
    - Ex: ActiveCell(1,2).Interior.Color = RGB(0,255,0)
  + For Interior.ColorIndex you set it equal to a single number which corresponds to the color position in the Excel colour pallet
    - Ex: ActiveCell(1,2).Interior.ColorIndex = 4
  + If you want to clear a background color, you can use the constant xlColorIndexNone
    - Ex ActiveCell(1,2).Interior.ColorIndex = xlColorIndexNone
    - Works with .ColorIndex and .Color
* You can align data with HorizontalAlignment and VerticalAlignment properties
  + Constants for HorizontalAlignment are
    - xlCenter
    - xlLeft
    - xlRight
  + Constants for VerticalAlignment are
    - xlBottom
    - xlCenter
    - xlTop
* Select Case is another way to work with conditionals
  + Syntax:

Select Case User\_Choice

Case "R"

MsgBox "Red"

Case "G"

MsgBox "Green"

Case "B"

MsgBox "Blue"

Case Else

MsgBox "None"

End Select

* + User\_Choice can be anything and will be evaluated to True or False in each case
  + You can check for more than one value if you use the word “To”, or separate values with commas
    - Ex Case 0 to 35 / Case 10,20,30,40
* With…end is a way to reduce redundancy of the same object
  + Ex:
    - ActiveCell.Font.Bold = True

ActiveCell.Font.Color = vbBlue

ActiveCell.Font.Name = “Arial”

ActiveCell.Font.Size = 22

ActiveCell.Font.Italic = True

* + - Is the same thing as
    - With ActiveCell.Font

.Bold = True

.Color = vbBlue

.Name = “Arial”

.Size = 22

.Italic = True

End With

* + Hence can be used on an object, and an object’s property and perhaps other stuff as well

# Strings and String Functions

* Strings must be surrounded by double quotes “”, including numbers if you want them to be recognized as text
* UCase and LCase – Uppercase and Lowercase
  + Found to the right of equal sign
  + Syntax: Range(“A2”).Offset(, 2).Value = UCase(FullName)
* Application is a top-level object – meaning the whole of excel
  + WorkSheetFunction is a function used to access Excel’s built-in functions
    - One of these functions is the Proper function – changes everything to proper case
* Len is a function that provides the number of characters in a string
  + Syntax: LengthFullName = Len(FullName)
* Trim is a function to remove unwanted spaces before and after a string
  + Syntax: FullName = Trim(“ word “)
* Space is a function that pads a string with spaces – uses parenthesis and a number to convey how many spaces
  + Syntax: FullName = Space(5) & FullName
* Replace is a function that changes characters in a string with something else
  + Syntax: CorrectedText = Replace(OriginalText, “a”, “o”)
  + Generally – Replace(string\_to\_search, string\_to\_replace, replace\_with)
    - Also includes optional things –

Replace(string\_to\_search, string\_to\_replace, replace\_with, start, count, compare)

* + - * Start – where in the string you want to start the search from – default Is character 1 which is the first character in the string
      * Count – how many occurrences you want to replace – ex if you only want to replace the first two occurrences, write 2 here
      * Compare – rarely used, dw about it
* InStr is a function that tells the location of a certain character in a string – if the character doesn’t exist in a string, it returns 0
  + Syntax: Location = InStr(“myaddress@myisp.com”, @)
    - Location should become 10, since @ is the tenth character
  + Two optional parameters – InStr(start, Text\_To\_Search, Find, Compare)
    - Start tells InStr which character to start
    - Compare is used rarely, dw about it
* Left and Right are functions which chops characters from a string
  + Left chops characters from start
  + Right chops characters from end
  + Syntax: MsgBox Left(“myaddress@myisp.com”, 9)
* Mid is a function used to grab characters from a string of text
  + Syntax: Mid(string\_to\_search, start\_position, number\_of\_characters\_to\_grab)
  + Ex: GrabbedChars = Mid(“myaddress@myisp.com”, 16, 4) will end up assigning GrabbedChars to .com

# Programming Loops

* For Loop is a general kind of loop
  + Syntax: For StartNumber = 1 To EndNumber
    - The ending is also known as the condition
  + VBA needs you to assign a value to the variable as the starting point for the loop. If you do it earlier, it will error
* After the condition in a For loop, you can add an optional Step value
  + Basically, the value of the increment
  + Default is for VBA to go through the start number to the end number in steps of 1 each time round the loop
  + Ex: For StartNumber = 10 To EndNumber Step -1
    - This will tell it to increment the value by -1
* The Final line of a loop is the Next line
  + Ex:

For StartNumber = 1 To EndNumber

[insert code]

Next StartNumber

* + This tells the loop to add 1 to the variable StartNumber
* For Each loops are normally used with collections and arrays – you have more items to cycle through
  + Syntax: For Each variable\_name In collection\_name
  + The last line is still the same “Next” line
  + Can be used for a range of cells
  + Ex:
  + For Each MyCell In Range(“A2:A6”)

MyCell.Value=Replace(MyCell.Value, “-“,””)

Next MyCell

* + - MyCell is the variable name that will store individual cells from the range
    - The collection would be the Range(“A2:A6”)
* Cells is a property, similar to Range to refer to cells on a spreadsheet
  + Has an optional Item property that you use to reference cells on spreadsheet
  + Syntax: Cells.Item(Row, Column)
    - The Column here can be both a letter or a number, whereas Row can only be number
  + Syntax: Without the Item property, you can just do Cells(Row, Column)
  + Ex: Cells.Item(1, “A”) is referring to cell A1 and is the same thing as Cells(1, “A”)
  + This property is useful in loops to help move through different cells
* Do While Loop is a loop that doesn’t need to set a start condition, just the end condition
  + Syntax: Do While [Condition]

Loop

* + Ex: Do While x < y
  + Be careful of infinite loops
  + You can also add the condition at the end, after the word “Loop”
    - This will execute the code first, and then check the condition
* Do Until loop is similar to Do While loop, except it tells the condition to continue Until
  + Syntax: Do Until [Condition]

Loop

* If you need to bail out early from any loop, then you need to use the word “Exit” followed by the loop name
  + Ex:

…

If Counter = 0 Then

Exit Do

Else ….

# Programming Arrays

* Array is a way to store more than one value under the same name
  + Syntax: Dim MyArray(4) As Integer
    - The number of values stores is actually five, since arrays are zero indexed hence you can store a value at 0
    - If you don’t like that, you can declare it like this
    - Dim MyArray(1 to 5) As Integer
  + When defining, simply write ex: MyArray(0) = 10, MyArray(4) = 50
* UBound is the upper boundary – retrieves the highest index of the array
  + Syntax: UBound(MyArray)
  + There also exists LBound, the lower bound
* Multi-dimensional arrays are basically arrays with more than one column
  + Syntax: Dim MyArray(5,4) As Integers
    - Different dimensions are separated by a comma
  + For 2-D arrays – MyArray(row, column)
* Split function
  + Syntax: Split(text\_here, separator)
  + Must use the type Variant when working with split, other types won’t work
    - Ex: Dim FullName as Variant

FullName = Split(txt, “ “)

* + - * The pieces of the Split will then turn FullName into an array
* Join function - Puts pieces of an array back together
  + Syntax: Join(text\_here, concatenator)
  + Should also use the type Variant when working with join

# Subs and Functions

* Subs are small chunks of code also known as subroutines
* You can call subs from another sub
  + Syntax: Call SecondCode
    - “Call” is optional, you can just write the name of the sub instead
* You can make subs private, by typing “Private” before “Sub”
  + Syntax: Private Sub SecondCode
* You can create parameters for subs within the round brackets
  + Syntax:   
    Call SecondCode(True, “Arial”)  
    …  
    Sub SecondCode(BoldValue as Boolean, NameValue As String)
* The MsgBox has more options available after the default quotations
  + You can set a variable to equal to the MsgBox function, since the MsgBox function can return something
    - Syntax:   
      Response = MsgBox(“Number Required”, vbOKCancel + vbCritical, “Exiting Sub”)
    - The “+” is there to separate the button and the symbol
  + There are different values from different buttons
    - You can refer to them [here](http://www.homeandlearn.org/excel_vba_message_boxes.html)
* Functions are like subs, except they can return a value – subs just show msgbox or edit cells
  + Syntax:   
    Function function\_name() As variable\_type  
    ….  
    End Function
    - Whatever value you return from this function, will be stored to your function\_name
  + To return a value from your functions, you need this 🡪 function\_name = value
  + Can’t change anything on a worksheet in functions
  + You can set them as private, just like a sub
  + If you want access to another function you coded in another rworksheet, then you need to name the worksheet first
    - Syntax: Sheet1.CheckCell(“Ten”)
  + You can also Exit Function, just like Exit Sub
* .AutoFit helps to automatically adjust the column/row
  + Ex Range(“C1”).Columns.AutoFit
* As Range is a variable type – used to hold a range of cells from your spreadsheet
* Set is a keyword that can be used to create new object variables
* .Find is a method to search the cells in the new range

# Excel VBA and Text Files

* Open is a function which opens documents
  + Syntax: Open FilePath for Mode As FileNumber
    - Mode determines the nature of the file - can be any one of the following:
      * Append, Output, Input, Binary, Random
    - FileNumber can be any number between 1 and 511 🡪 must be preceded by a “#”
      * Basically just creates like a variable name thingy -> you must count up properly
        + Ex first file would be #1, second would be #2, etc.
* Once you’ve opened a file, you should close it – use the Close function
  + Ex: Close #1
* EOF() means End Of File
  + Put the file number in the brackets
  + Ex: Do Until EOF(1)