

# FIT1013 - Week 6 Resources

Variables and Selection Structures

# Week 6 Resources

1. Objectives .....	2
2. Creating and Using Object Variables in Excel .....	2
3. Data Types Used to Reserve Numeric Variables .....	13
4. Using an Assignment Statement to Assign a Value to a Numeric Variable .....	15
5. Assigning a Numeric Expression to a Numeric Variable .....	16
6. Example: Viewing the Paradise Electronics Price List.....	17
7. Using the Excel VLookup function .....	23
8. Program design – VBA control structures .....	27
9. Comparison Operators .....	29
10. UCase Function .....	31
11. Nested Selection Structure .....	37
12. Compiling your VBA code.....	45
13. Practice and Apply.....	51

## Reference:

New Perspectives Excel 2013 Appendix C “Enhancing Excel with VBA”

Note: the NP Excel 2016 Edition does not cover Excel VBA

Diane Zak, “Visual Basic for Applications” 2001

Available from Hargrave Andrews library

Useful reading!!!! (Covers Excel, Access, Powerpoint, Word).

Sections © 2017 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part, except for use as permitted in a license distributed with a certain product or service or otherwise on a password-protected website for classroom use.

# 1. Objectives

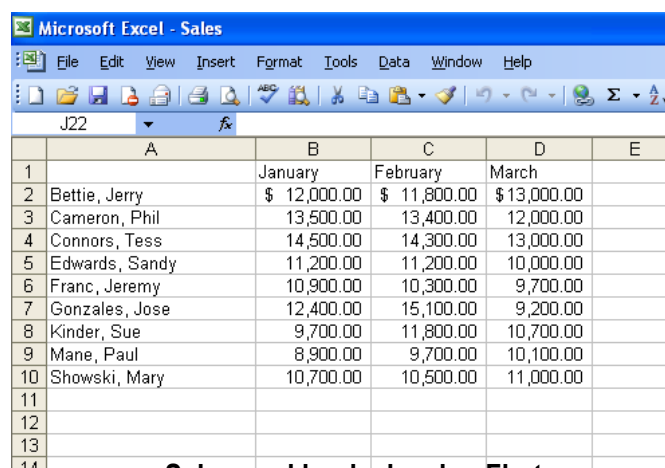
- Use object variables in Excel
- Reserve a numeric variable
- Use an assignment statement to assign a value to a numeric variable
- Perform calculations using arithmetic operators
- Add a list box to an Excel worksheet
- Use the Excel VLookup function in a procedure
- Perform selection using the **If...Then...Else** statement
- Write instructions that use comparison operators and logical operators
- Use the **UCase** function
- Use the nested **If...Then...Else** statement

## 2. Creating and Using Object Variables in Excel

- This example covers:
  - Creating a macro procedure called **FormatWorksheet** by using:
    - A **range object**
    - The **Autoformat** method of the range object to format the range object
    - The **PrintPreview** method of the range object
- This procedure formats a worksheet using the predefined format types in Excel and present the result in print preview mode

### Excel VBA E.g. : Creating the FormatWorksheet Macro Procedure

- Open Sales workbook
  - [Sales.xls](#)
- Insert a module
  - Click Insert on the menu bar
  - Click Module
- Insert a procedure
  - Click Insert on the menu bar
  - Click Procedure
  - Type **FormatWorkSheet** for the name of the procedure



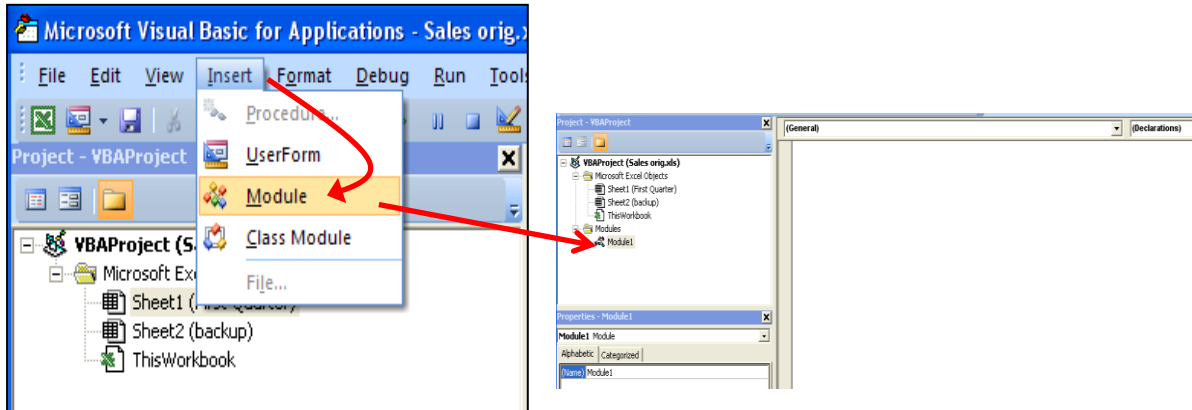
The screenshot shows the Microsoft Excel interface with the 'Sales' workbook open. The worksheet displays sales data for ten individuals across four quarters: January, February, and March. The data is formatted with currency symbols and commas for thousands. The formula bar shows 'J22' and the active cell is J22.

	A	B	C	D	E
1		January	February	March	
2	Bettie, Jerry	\$ 12,000.00	\$ 11,800.00	\$ 13,000.00	
3	Cameron, Phil	13,500.00	13,400.00	12,000.00	
4	Connors, Tess	14,500.00	14,300.00	13,000.00	
5	Edwards, Sandy	11,200.00	11,200.00	10,000.00	
6	Franc, Jeremy	10,900.00	10,300.00	9,700.00	
7	Gonzales, Jose	12,400.00	15,100.00	9,200.00	
8	Kinder, Sue	9,700.00	11,800.00	10,700.00	
9	Mane, Paul	8,900.00	9,700.00	10,100.00	
10	Showski, Mary	10,700.00	10,500.00	11,000.00	
11					
12					
13					
14					

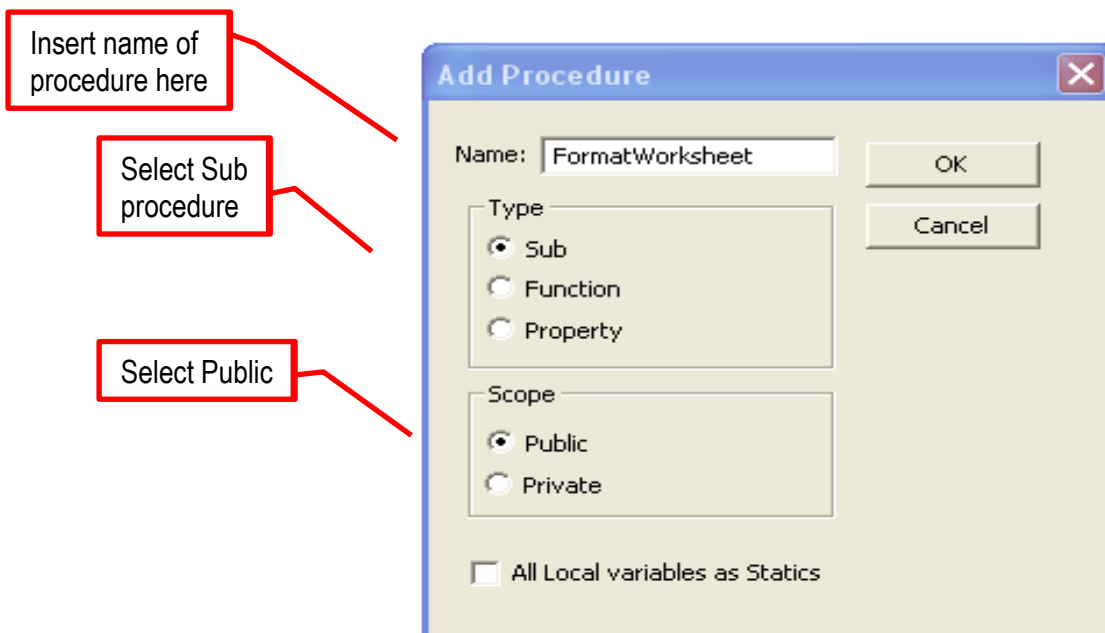
Sales workbook showing First Quarter worksheet

## Creating a module

- Remember modules contain procedures
- To create a module within the VBE



## Add Procedure Dialog Box



## Worksheet Format Desired by the District Sales Manager

Cell A1 contains the company name

2 new rows

Accounting2 format

	A	B	C	D
1	Paradise Electronics			
2				
3		January	February	March
4	Bettie, Jerry	\$ 12,000.00	\$ 11,800.00	\$ 13,000.00
5	Cameron, Phil	13,500.00	13,400.00	12,000.00
6	Connors, Tess	14,500.00	14,300.00	13,000.00
7	Edwards, Sandy	11,200.00	11,200.00	10,000.00
8	Franc, Jeremy	10,900.00	10,300.00	9,700.00
9	Gonzales, Jose	12,400.00	15,100.00	9,200.00
10	Kinder, Sue	9,700.00	11,800.00	10,700.00
11	Mane, Paul	8,900.00	9,700.00	10,100.00
12	Showski, Mary	10,700.00	10,500.00	11,000.00
13		\$ 103,800.00	\$ 108,100.00	\$ 98,700.00
14				
15				
16				

These cells contain formulas:  
 =Sum(B4:B12), =Sum(C4:C12), =Sum(D4:D12)  
 respectively

## Worksheet Format Desired by the Regional Sales Manager

Code		Add-Ins		Controls	
B6		Cell A1 contains the company name			
	A	B	C	D	E
1	Paradise Electronics				
2	2 new rows		Accounting2 format		
3		January	February	March	
4	Bettie, Jerry	\$ 12,000.00	\$ 11,800.00	\$ 13,000.00	
5	Cameron, Phil	13,500.00	13,400.00	12,000.00	
6	Connors, Tess	14,500.00	14,300.00	13,000.00	
7	Edwards, Sandy	11,200.00	11,200.00	10,000.00	
8	Franc, Jeremy	10,900.00	10,300.00	9,700.00	
9	Gonzales, Jose	12,400.00	15,100.00	9,200.00	
10	Kinder, Sue	9,700.00	11,800.00	10,700.00	
11	Mane, Paul	8,900.00	9,700.00	10,100.00	
12	Showski, Mary	10,700.00	10,500.00	11,000.00	
13		\$ 103,800.00	\$ 108,100.00	\$ 98,700.00	
14					

These cells contain formulas:  
 =Sum(B4:B12), =Sum(C4:C12), =Sum(D4:D12)  
 respectively

## Creating the FormatWorksheet Macro Procedure

- Pseudocode is composed of short English statements
- It is a tool programmers use to help them plan the steps that a procedure must take in order to perform an assigned task

1. Insert two rows at the top of the worksheet.
2. Enter Paradise Electronics in cell A1.
3. Enter formulas in cells B13 through D13 that add the contents of the January, February, and March columns.
4. Format cells A1 through D13 to the Accounting2 format for the district sales manager.
5. Print the worksheet for the district sales manager.
6. Format cells A1 through D13 to the Classic2 format for the regional sales manager.
7. Print the worksheet for the regional sales manager.

### Pseudocode for the FormatWorksheet procedure

## Pseudo code

1. Insert 2 rows at top of worksheet
2. Enter "Paradise Electronics" in cell A1
3. Enter formulas in cells B13 to D13 that add the contents of January, February and March columns
4. Format cells A1 to D13 in Accounting2 format (of the Autoformat method) for the district sales manager
5. Print the worksheet for the district sales manager
6. Format cells A1 to D13 in Classic2 format (of the Autoformat method) for the regional sales manager
7. Print the worksheet for the regional sales manager

	A	B	C	D	
1		January	February	March	
2	Bettie, Jerry	\$ 12,000.00	\$ 11,800.00	\$ 13,000.00	
3	Cameron, Phil	13,500.00	13,400.00	12,000.00	
4	Connors, Tess	14,500.00	14,300.00	13,000.00	
5	Edwards, Sandy	11,200.00	11,200.00	10,000.00	
6	Franc, Jeremy	10,900.00	10,300.00	9,700.00	
7	Gonzales, Jose	12,400.00	15,100.00	9,200.00	
8	Kinder, Sue	9,700.00	11,800.00	10,700.00	
9	Mane, Paul	8,900.00	9,700.00	10,100.00	
10	Showski, Mary	10,700.00	10,500.00	11,000.00	
11					
12					

## Inserting Rows Into a Worksheet

You can insert a row into a worksheet using the syntax:

***worksheetObject.Rows(rowNumber).Insert***

where *worksheetObject* is the name of a Worksheet object and *rowNumber* is the number of the row above which the new row will be inserted

– [Sales.xls](#)

e.g.

Without an object variable, you insert a row above row 1 and above row 5 in the First Quarter worksheet as follows:

**Application.Workbooks("sales.xlsx").Worksheets("first quarter").Rows(1).Insert**

**Application.Workbooks("sales.xlsx").Worksheets("first quarter").Rows(5).Insert**

Rows property of  
Worksheet object

Insert method

Once you create an object variable called **wksFirst** that points to the First Quarter worksheet, you can insert a row above row 1 and above row 5 in the First Quarter worksheet as follows:

**wksFirst.Rows(1).Insert**

**wksFirst.Rows(5).Insert**

The following code creates an object variable and then uses it to enter further code:

Public Sub FormatWorksheet()

'declare object variable and assign address

Dim wksFirstQ As Worksheet

Set wksFirstQ = Application.Workbooks("sales.xls").Worksheets(1)

'insert 2 rows above row 1

wksFirstQ.Rows(1).Insert

wksFirstQ.Rows(1).Insert

Declare a  
Worksheet  
object  
variable

Assign the address of the first worksheet  
in the Sales workbook to the Worksheet  
object variable

Insert 2 rows  
at the top of  
the worksheet



## Entering a Value Into a Range Object

- Recall that a row, a column, or a group of contiguous or non-contiguous cells in a worksheet also are Excel Range objects
- You need to enter the following formulas in cells B13 through D13 in the worksheet:

- B13 formula = SUM (B4:B12)
- C13 formula = SUM (C4:C12)
- D13 formula = SUM (D4:D12)

These formulas will add the contents of their respective columns

Using three instructions:

wksFirstQ.Range("b13").Formula = "=sum(b4:b12)"

wksFirstQ.Range("c13").Formula = "=sum(c4:c12)"

wksFirstQ.Range("d13").Formula = "=sum(d4:d12)"

Formula property of the Range object

Or using one instruction:

wksFirstQ.Range("b13:d13").Formula = "=sum(b4:b12)"

Cell references are relative, so will be adjusted for c13 and d13

## Entering a value in a range object

The following code will assign "Paradise Electronics" to cell A1:

wksFirstQ.Range("a1").Value = "Paradise Electronics"

Value property

Range object

	A	B	C	D
1	Paradise Electronics			
2				
3		January	February	March
4	Bettie, Jerry	\$ 12,000.00	\$ 11,800.00	\$ 13,000.00
5	Cameron, Phil	13,500.00	13,400.00	12,000.00
6	Connors, Tess	14,500.00	14,300.00	13,000.00
7	Edwards, Sandy	11,200.00	11,200.00	10,000.00
8	Franc, Jeremy	10,900.00	10,300.00	9,700.00
9	Gonzales, Jose	12,400.00	15,100.00	9,200.00
10	Kinder, Sue	9,700.00	11,800.00	10,700.00
11	Mane, Paul	8,900.00	9,700.00	10,100.00
12	Showski, Mary	10,700.00	10,500.00	11,000.00
13		\$ 103,800.00	\$ 108,100.00	\$ 98,700.00
14				

## Code so far:

```
Public Sub FormatWorksheet()  
    'declare object variable and assign address  
    Dim wksFirstQ As Worksheet  
    Set wksFirstQ = Application.Workbooks("sales.xls").Worksheets(1)  
    'insert 2 rows above row 1  
    wksFirstQ.Rows(1).Insert  
    wksFirstQ.Rows(1).Insert  
    'enter company name  
    wksFirstQ.Range("a1").Value = "Paradise Electronics"  
    'enter totals formulas  
    wksFirstQ.Range("b13:d13").Formula = "=sum(b4:b12)"
```

Value property of  
range object

Formula property of range  
object

## Formatting a Range Object using the AutoFormat method

- A collection of predesigned worksheet formats is available in Excel
- <https://msdn.microsoft.com/en-us/library/microsoft.office.interop.excel.xlrangeautoformat.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1>

```
xlRangeAutoFormatAccounting1  
xlRangeAutoFormatAccounting2  
xlRangeAutoFormatAccounting3  
xlRangeAutoFormatAccounting4  
xlRangeAutoFormatClassic1  
xlRangeAutoFormatClassic2  
xlRangeAutoFormatClassic3
```

Names of some of the Excel predesigned  
formats

## Some examples of AutoFormat formats

xlRangeAutoFormatAccounting1

Paradise Electronics

	January	February	March
Bettie, Jerry	\$ 12,000.00	\$ 11,800.00	\$ 13,000.00
Cameron, Phil	13,500.00	13,400.00	12,000.00
Connors, Tess	14,500.00	14,300.00	13,000.00
Edwards, Sandy	11,200.00	11,200.00	10,000.00
Franco, Jeremy	10,900.00	10,300.00	9,700.00
Gonzales, Jose	12,400.00	15,100.00	9,200.00
Kinder, Sue	9,700.00	11,800.00	10,700.00
Mane, Paul	\$ 84,200.00	\$ 87,900.00	\$ 77,600.00
Showski, Mary	10,700.00	10,500.00	11,000.00
	#####	#####	#####

xlRangeAutoFormatClassic3

Paradise Electronics			
	January	February	March
Bettie, Jerry	\$ 12,000.00	\$ 11,800.00	\$ 13,000.00
Cameron, Phil	13,500.00	13,400.00	12,000.00
Connors, Tess	14,500.00	14,300.00	13,000.00
Edwards, Sandy	11,200.00	11,200.00	10,000.00
Franco, Jeremy	10,900.00	10,300.00	9,700.00
Gonzales, Jose	\$ 62,100.00	\$ 61,000.00	\$ 57,700.00
Kinder, Sue	9,700.00	11,800.00	10,700.00
Mane, Paul	#####	#####	#####
Showski, Mary	10,700.00	10,500.00	11,000.00
	#####	#####	#####

xlRangeAutoFormatClassic2

Paradise Electronics			
	January	February	March
Bettie, Jerry	\$ 12,000.00	\$ 11,800.00	\$ 13,000.00
Cameron, Phil	13,500.00	13,400.00	12,000.00
Connors, Tess	14,500.00	14,300.00	13,000.00
Edwards, Sandy	\$ 40,000.00	\$ 39,500.00	\$ 38,000.00
Franco, Jeremy	10,900.00	10,300.00	9,700.00
Gonzales, Jose	\$ 90,900.00	\$ 89,300.00	\$ 85,700.00
Kinder, Sue	9,700.00	11,800.00	10,700.00
Mane, Paul	#####	#####	#####
Showski, Mary	10,700.00	10,500.00	11,000.00
	#####	#####	#####

## Formatting a Range Object

'format worksheet for district sales manager

wksFirstQ.Range("a1:d13").AutoFormat \_

Format:=xlRangeAutoFormatAccounting2

Using the AutoFormat method of the Range object to format a range using an Excel predesigned format

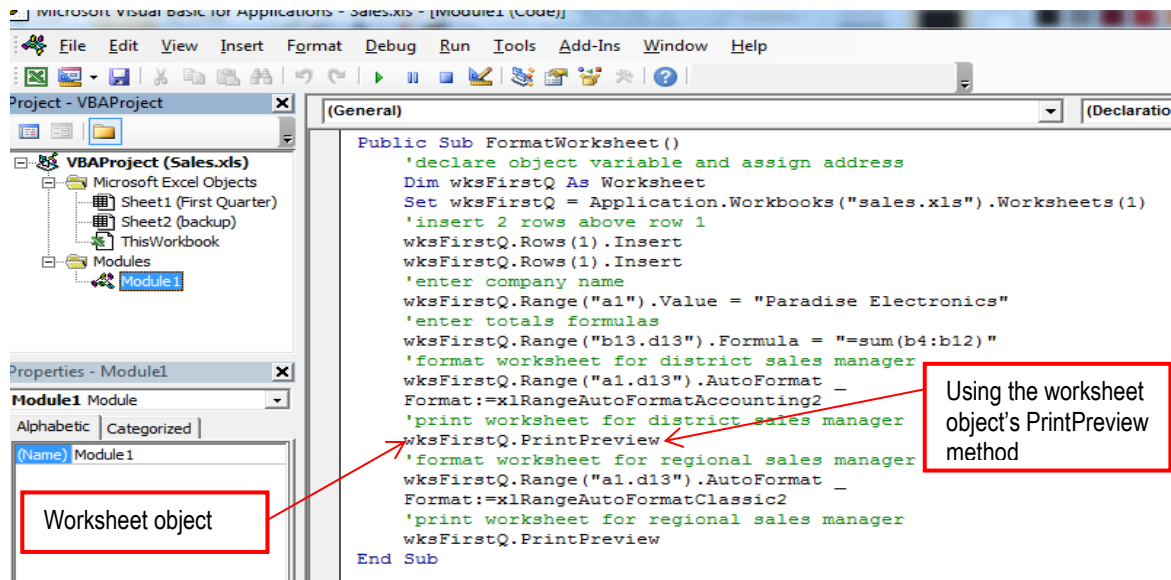
Value of the Format argument

Format argument of the AutoFormat method

## Previewing and Printing a Worksheet Object

'print worksheet for district sales manager

wksFirstQ.PrintPreview



## Completed Code (module1)

```
Public Sub FormatWorksheet()  
    'declare object variable and assign address  
    Dim wksFirstQ As Worksheet  
    Set wksFirstQ = Application.Workbooks("sales.xls").Worksheets(1)  
    'insert 2 rows above row 1  
    wksFirstQ.Rows(1).Insert  
    wksFirstQ.Rows(1).Insert  
    'enter company name  
    wksFirstQ.Range("a1").Value = "Paradise Electronics"  
    'enter totals formulas  
    wksFirstQ.Range("b13.d13").Formula = "=sum(b4:b12)"  
    'format worksheet for district sales manager  
    wksFirstQ.Range("a1.d13").AutoFormat _  
    Format:=xlRangeAutoFormatAccounting2  
    'print worksheet for district sales manager  
    wksFirstQ.PrintPreview  
    'format worksheet for regional sales manager  
    wksFirstQ.Range("a1.d13").AutoFormat _  
    Format:=xlRangeAutoFormatClassic2  
    'print worksheet for regional sales manager  
    wksFirstQ.PrintPreview  
End Sub
```

## First Quarter Worksheet After Running the FormatWorksheet Macro

Company name

2 new rows

Classic2 format

These cells contain formulas

Microsoft Excel - Sales				
File Edit View Insert Format Tools Data Window Help				
C:\Sue Laptop Data\aaa_FIT1013				
M36				
A B C D				
1	Paradise Electronics			
2				
3		January	February	March
4	Bettie, Jerry	\$ 12,000.00	\$ 11,800.00	\$ 13,000.00
5	Cameron, Phil	13,500.00	13,400.00	12,000.00
6	Connors, Tess	14,500.00	14,300.00	13,000.00
7	Edwards, Sandy	11,200.00	11,200.00	10,000.00
8	Franc, Jeremy	10,900.00	10,300.00	9,700.00
9	Gonzales, Jose	12,400.00	15,100.00	9,200.00
10	Kinder, Sue	9,700.00	11,800.00	10,700.00
11	Mane, Paul	8,900.00	9,700.00	10,100.00
12	Showski, Mary	10,700.00	10,500.00	11,000.00
13		\$ 103,800.00	\$ 108,100.00	\$ 98,700.00
14				
15				
16				

## Summary

- **Declaring** object variables using the **Dim** statement
- **Assigning the address** of an object to an object variable using the **Set statement**
- Inserting a new line on a worksheet
- Formatting a range using the AutoFormat method for a Range
- Viewing a range in print preview mode

### 3. Data Types Used to Reserve Numeric Variables

<i>datatype</i> Keyword	Name ID	Stores	Memory required	Range of values
Integer	int	Integers (whole numbers)	2 bytes	-32,768 to 32,767
Long	lng	Integers (whole numbers)	4 bytes	+/- 2 billion
Single	sng	Numbers with a decimal portion	4 bytes	0  Negative numbers: -3.402823E38 to -1.401298E-45  Positive numbers: 1.401298E-45 to 3.402823E38
Currency	cur	Numbers with a decimal portion	8 bytes	-922,337,203,685,477.5808 to 922,337,203,685,477.5807

Data types used to reserve numeric variables

#### Reserving a Procedure-level Numeric Variable

**Dim** statements can be used to reserve a procedure-level numeric variable, which is a memory cell that can store a number only.

E.g.

```
Dim intAge as Integer
Dim lngPopSize as Long
Dim sngGSTRate as single
Dim curNet as currency
```

- Variables assigned either the Integer or the Long data type can store integers, which are whole numbers
- The difference between the two data types is in the range of numbers each type can store and the amount of memory each type needs to store the numbers
- After declaration, numeric variables are automatically initialised to 0.

## Reserving a Procedure-level Numeric Variable

- The memory requirement of a data type is an important consideration when coding a procedure
- Long data type uses 4 bytes of memory, while the Integer data type uses only 2 bytes

Dim statement	Result
Dim intAge As Integer	Declares an Integer variable named intAge
Dim lngPop As Long	Declares a Long variable named lngPop
Dim sngRate As Single	Declares a Single variable named sngRate
Dim curGross As Currency	Declares a Currency variable named curGross

Examples of Dim statements that reserve numeric variables

## 4. Using an Assignment Statement to Assign a Value to a Numeric Variable

To assign a value to a variable:

*variablename = value*

- When *variablename* is the name of a numeric variable, a value can be a **number**, more technically referred to as a **numeric literal constant**, or it can be a **numeric expression**

### Assigning a Numeric Literal Constant to a Numeric Variable

- A **numeric literal** constant is simply a number e.g. 3.14159
- A **numeric literal** constant cannot contain any letter, except for the letter E, which is used in exponential notation (e.g. `sngGSTRate = 1.35E-2`)
- **Numeric literal** constants also cannot contain special symbols, such as the % sign, the \$ sign, or the comma
- They also cannot be enclosed in quotation marks (") or number signs (#), because numbers enclosed in quotation marks are considered string literal constants, and numbers enclosed in number signs are considered date literal constants

Valid numeric literal constants	Invalid numeric literal constants
0	2%
.3	\$4.56
4.5	123A
5	5,678
3200.67	"45"

Examples of valid and invalid numeric literal constants

```
intAge = 21
lngPop = 60450
sngRate = 0.05
curGross = 300.75
```

Examples of assignment statements that assign numeric literal constants to variables



## 5. Assigning a Numeric Expression to a Numeric Variable

- Numeric expressions can contain items such as numeric literal constants, variable names, functions, and arithmetic operators
- The precedence numbers represent the order in which the arithmetic operations are processed in an expression
- You can use parentheses to override the order of precedence because operations within parentheses always are performed before operations outside of parentheses

Operator	Operation	Order of precedence
^	exponentiation (raises a number to a power)	1
-	negation	2
*, /	multiplication and division	3
+, -	addition and subtraction	4
<b>Important Note:</b> You can use parentheses to override the order of precedence. Operations within parentheses are always performed before operations outside parentheses.		

Arithmetic operators and their order of precedence

- When you create a numeric expression that contains more than one arithmetic operator, keep in mind that VBA follows the same order of precedence as you do when evaluating the expression

E.g.

```
sngMinutes = Val(strHours) * 60
curNet = CurGross * (1-sngTaxRate)
sngAvg = intN1 + intN2 / 2
sngAvg = intN1 / 2 + intN2 / 2
sngAvg = (intN1 + intN2) / 2
```

### Summary

- To reserve a procedure-level numeric variable:
- Use the **Dim** statement. The syntax of the **Dim** statement is:  
**Dim variablename As datatype**  
where variablename represents the name of the variable (memory cell) and datatype is the type of data the variable can store  
**e.g. Dim intAge as Integer**  
(Recall: variable names must begin with a letter and they can contain only letters, numbers, and the underscore)
- To assign a value to a numeric variable:  
Use an assignment statement with the following syntax:  
**variablename=value**  
e.g. intAge = 21

## 6. Example: Viewing the Paradise Electronics Price List

The Computers worksheet

Paradise Electronics - Computers				Price List			
				Model #	Price		
				C100	2,200.00		
				C200	2,395.00		
				D430	3,450.00		
				D480	999.00		
				G250	1,299.00		
				H290	2,299.00		
				H560	3,495.00		
				H780	3,995.00		
				J480	1,200.00		
				J631	2,400.00		
				J651	3,599.00		

Excel Numeric Var e.g.: Viewing the Paradise Electronics Price List

Paradise Electronics - Computers				Price List			
				Model #	Price		
				C100	2,200.00		
				C200	2,395.00		
				D430	3,450.00		
				D480	999.00		
				G250	1,299.00		
				H290	2,299.00		
				H560	3,495.00		
				H780	3,995.00		
				J480	1,200.00		
				J631	2,400.00		
				J651	3,599.00		

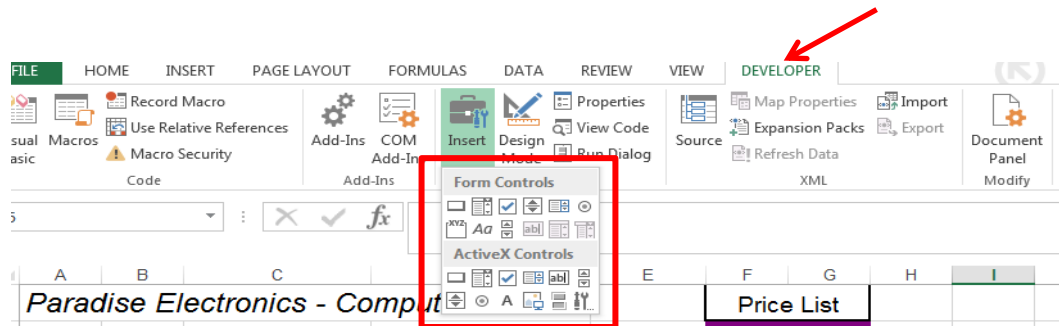
This exercise involves:

- Creating a list box that contains the model numbers of the products
- When the user double clicks the selected model number, an input dialogue box is displayed
- When the user types in the discount rate and presses the OK button, the yellow box as shown is updated.

## Creating a List Box

- A list box is one of several objects, called controls, that can be added to a worksheet
- You typically use a list box to display a set of choices from which the user can select only one
- List boxes help prevent errors from occurring in the worksheet

## Controls

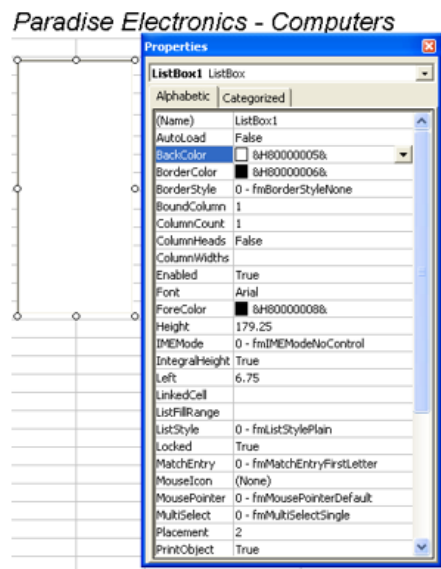


- Form controls
  - Original controls, compatible with earlier versions of Excel, starting from Excel 5.0
- ActiveX controls
  - Use on VBA UserForms and for more flexible design requirements

<https://support.office.com/en-us/article/Overview-of-forms-Form-controls-and-ActiveX-controls-on-a-worksheet-15BA7E28-8D7F-42AB-9470-FFB9AB94E7C2>

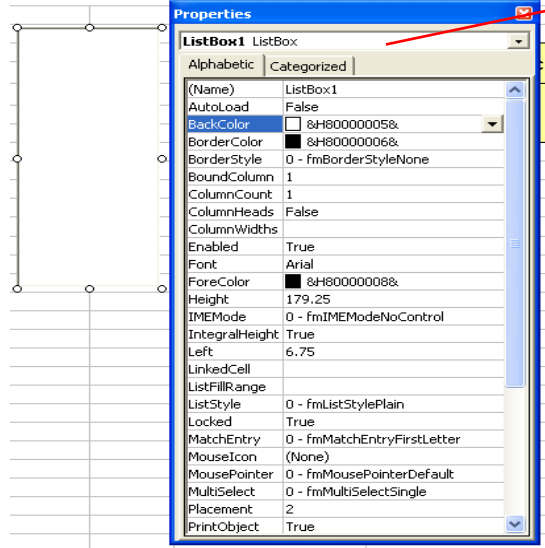
## Common properties for Controls Toolbox controls

- Name
- Autosize
- Enabled
- Font
- Left, Top, Width, Height
- Linked Cell
- ListFillRange
- PrintObject
- Etc...



## List box Control properties

*Paradise Electronics - Computers*

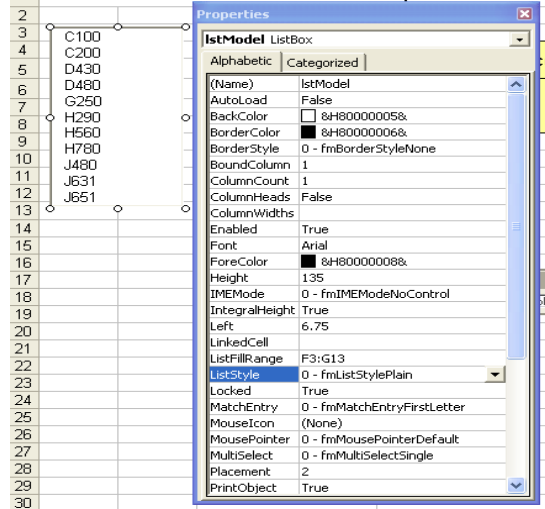


An Object box, located immediately below the Properties window's title bar, displays the **name** and type of the selected object.

A Properties list displayed (alphabetically/categorically) has 2 columns containing:

1. A list of all properties associated with the selected object
2. Settings (or current value) of each of those properties

*Paradise Electronics - Computers*

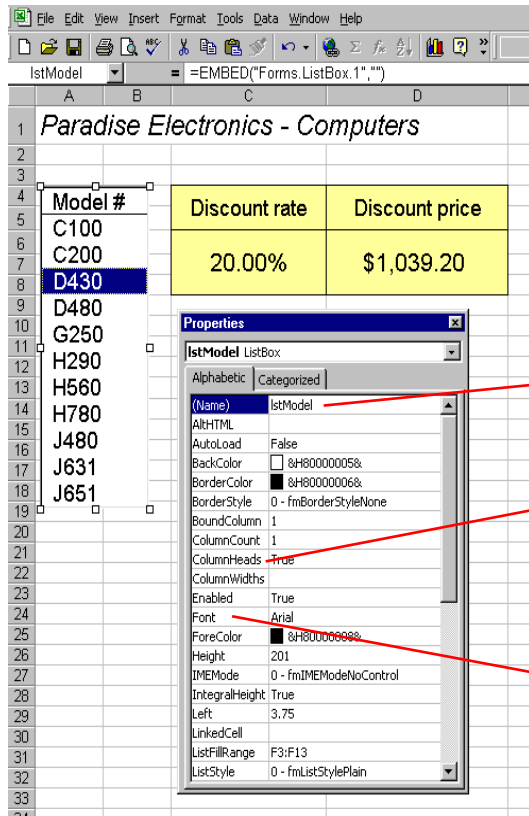


C100	2,200.00
C200	2,395.00
D430	3,450.00
D480	999.00
G250	1,299.00
H290	2,299.00
H560	3,495.00
H780	3,995.00
J480	1,200.00
J631	2,400.00
J651	3,599.00

ListFillRange refers to F3:G13, the price

## Naming of Controls

- The Name of the control has a special meaning in VBA
- This name is used by VBA to identify the object
- It is used in your VBA code
- The rules for naming controls are the same as the rules that you've seen for naming variables
- In this case, the name of the list box is **IstModel**

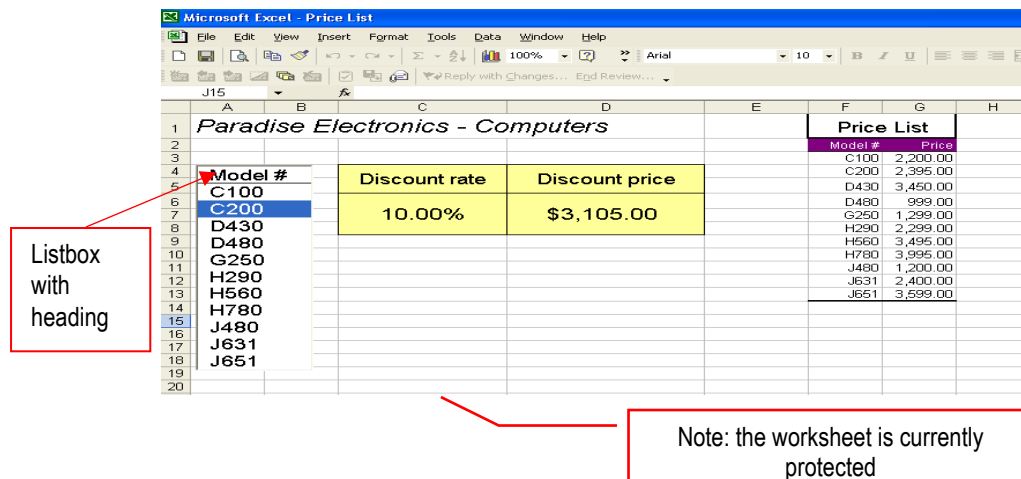


Default name changed to IstModel

Change ColumnHeads to True to see the heading

Click Font and ellipsis (...) to change font properties

## List Box after adding heading



Listbox with heading

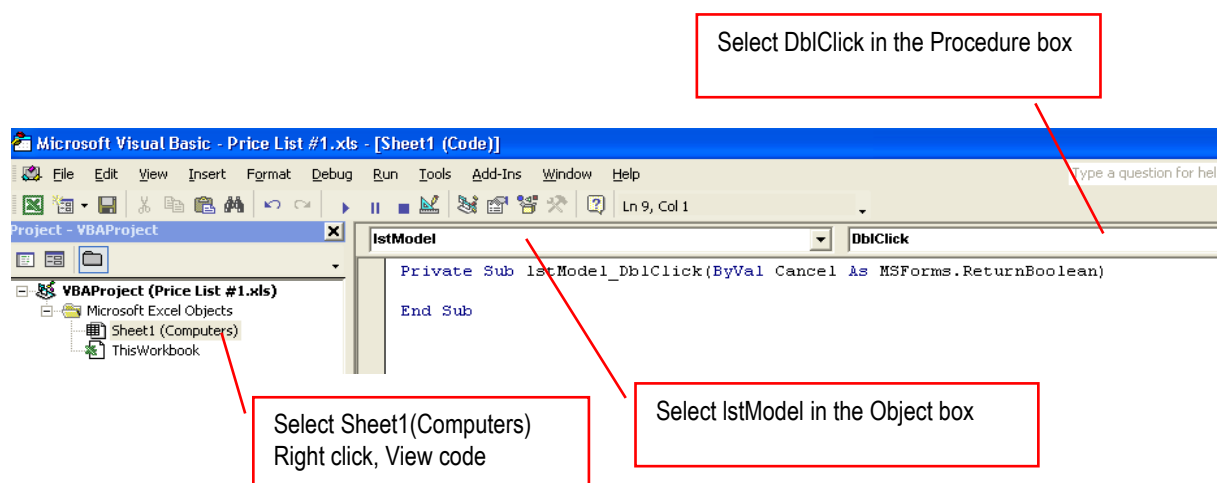
Note: the worksheet is currently protected

## Variables Used by the List Box's DblClick Event Procedure

Variable	Data type
strRate	String
sngRate	Single
curPrice	Currency
curDiscPrice	Currency
wksComputers	Worksheet

## Coding the List Box's DblClick Event Procedure

To begin coding the DblClick event procedure



To begin coding the DbClick event procedure, declare and set the variables as shown

```
Private Sub IstModel_DbClick(ByVal Cancel As MSForms.ReturnBoolean)
```

```
'declare variables and assign address to object variable
```

```
Dim strRate As String
```

For capturing user input

```
Dim sngRate As Single
```

For converting the rate to a number

```
Dim curPrice As Currency
```

```
Dim curDiscPrice As Currency
```

For referencing cells on the computers worksheet

```
Dim wksComputers As Worksheet
```

```
Set wksComputers = Application.Workbooks("pricelist.xls").Worksheets("computers")
```

```
End Sub
```

For storing the price of the

For storing the discount price of the selected item

## Coding the List Box's DbClick Event Procedure

Unprotecting the worksheet and using the InputBox to prompt for the discount rate

```
Private Sub IstModel_DbClick(ByVal Cancel As MSForms.ReturnBoolean)
```

```
'declare variables and assign address to object variable
```

```
Dim strRate As String, sngRate As Single
```

```
Dim curPrice As Currency, curDiscPrice As Currency, wksComputers As Worksheet
```

```
Set wksComputers = Application.Workbooks("price list.xls").Worksheets("computers")
```

```
'unprotect worksheet
```

```
wksComputers.Unprotect
```

The worksheet Unprotect method. Unprotects the "Computers" worksheet

```
'enter discount rate
```

```
strRate = InputBox(prompt:="Enter discount rate (whole number):", _
```

```
Title:="Rate", Default:=0)
```

```
'convert rate to decimal
```

```
sngRate = Val(strRate) / 100
```

Obtains discount rate as a string. Assigns it to strRate

```
End Sub
```

Converts string to decimal

## 7. Using the Excel VLookup function

- You can use Excel's **VLookup** function to search for, or "look up," a value located in the first column of a vertical list, and then return a value located in one or more columns to its right
- In the **VLookup** function's syntax, **lookup\_value** is the value to be found in the first column of table, which is the location of the range that contains the table of information
- When **range\_lookup** is True, or when the argument is omitted, the VLookup function performs a case-insensitive approximate search, stopping when it reaches the largest value that is less than or equal to the **lookup\_value**

### Syntax for vlookup() function

- **VLOOKUP(lookup\_value, table\_array, col\_index\_num, range\_lookup)**
  - **lookup\_value**: the value that is sent to the table; it can be a value or a reference to a cell that contains a value or text string
  - **table\_array**: specifies the location of the lookup table
  - **col\_index\_num**: the column number of the lookup table containing the information you want to retrieve
  - **range\_lookup**: a logical value (TRUE or FALSE) tells VLOOKUP how to match the compare values in the first column of the lookup table. If **range\_lookup** = FALSE then VLOOKUP looks for an exact match. If **range\_lookup** = TRUE (or omitted) then VLOOKUP looks for the largest compare value that is less or equal to the lookup value

F	G
Price List	
Model #	Price
C100	2,200.00
C200	2,395.00
D430	3,450.00
D480	999.00
G250	1,299.00
H290	2,299.00
H560	3,495.00
H780	3,995.00
J480	1,200.00
J631	2,400.00
J651	3,599.00

e.g. Lookup\_value: one of the Model codes G250

e.g. F3:G13

e.g. col\_index\_num = 2



## Coding the List Box's DblClick Event Procedure

A list box's **DblClick event procedure** occurs when the user double-clicks an item in the list.

1. **Unprotect** the Computers worksheet
2. Use the **InputBox** function to prompt the user to enter the discount rate as a whole number. Store the user's response in a string variable named **strRate**.
3. Convert the rate stored in **strRate** to its decimal equivalent by using the Val function and dividing the rate by 100. Store the result in a single variable called **sngRate**.
4. Use the **Vlookup** worksheet function to search the first column of the Price List for the model number that is selected in the **lstModel** List box. Assign the computer's price from the second column of the price list to a Currency variable named **curPrice**
5. Calculate the discounted price first by subtracting the discount rate from 1, then multiplying that difference by the computer's price, which is stored in the **curPrice** variable. Assign the discounted price to a currency variable named **curDiscPrice**.
6. Display the discount rate, stored in the **sngRate** variable, in cell C6 in the Computers worksheet
7. Display the discounted price, stored in the **curDiscPrice** variable, in cell D6 in the computers worksheet
8. **Protect** the Computers worksheet.

## Using the Excel Worksheet functions in a Procedure

- Most Excel worksheet functions are held in the **WorksheetFunction** object
- To use an Excel worksheet function in VBA:
  - `Application.WorkSheetFunction.Function_Name`
  - E.g. `Application.WorkSheetFunction.Min(Range("A1:B100"))`

## Using the Excel Vlookup Function in a Procedure

```
Private Sub IstModel_DblClick(ByVal Cancel As MSForms.ReturnBoolean)
```

```
    'declare variables and assign address to object variable
```

```
    Dim strRate As String, sngRate As Single
```

```
    Dim curPrice As Currency, curDiscPrice As Currency, wksComputers As Worksheet
```

```
    Set wksComputers = Application.Workbooks("price list.xls").Worksheets("computers")
```

```
    'unprotect worksheet
```

```
    wksComputers.Unprotect
```

```
    'enter discount rate
```

```
    strRate = InputBox(prompt:="Enter discount rate", _  
        Title:="Rate", Default:=0)
```

```
    'convert rate to decimal
```

```
    sngRate = Val(strRate) / 100
```

Table array – which has been named “pricelist”

col\_index\_num = 2

Invoking the Vlookup Worksheet function to find the current price. WorksheetFunction object enables us to evaluate worksheet functions in VBA code

```
    'search for model number and return price
```

```
    curPrice = Application.WorksheetFunction.VLookup(IstModel.Text, _  
        Range("pricelist"), 2, False)
```

Range\_lookup = false, ensures an exact match in “pricelist”

```
    'calculate the discounted price
```

```
    curDiscPrice = (1 - sngRate) * curPrice
```

Calculating the discount price

```
    'display discount rate and discounted price
```

```
    wksComputers.Range("c6").Value = sngRate
```

```
    wksComputers.Range("d6").Value = curDiscPrice
```

```
    'protect worksheet
```

```
    wksComputers.Protect
```

```
End Sub
```

Model #	Discount rate	Discount price
C100	10.00%	\$3,105.00
D430		3450.00
D480		3600.00
G260		3750.00
H260		3900.00
H560		4050.00
H780		4200.00
J480		4350.00
J831		4500.00
J851		4650.00

Displays the results in the “yellow” region

## Worksheet after running the list box's DbClick event

Microsoft Excel - Price List

FileEditViewInsertFormatToolsDataWindowHelp

100%

Arial

10

**B***I*U

fx

	A	B	C	D	E	F	G	H
1	Paradise Electronics - Computers						Price List	
2						Model #	Price	
3						C100	2,200.00	
4	Model #		Discount rate	Discount price		C200	2,395.00	
5	C100		12.00%	\$2,023.12		D430	3,450.00	
6	C200				D480	999.00		
7	D430					G250	1,299.00	
8	D480					H290	2,299.00	
9	G250					H560	3,495.00	
10	H290					H780	3,995.00	
11	H560					J480	1,200.00	
12	H780					J631	2,400.00	
13	J480					J651	3,599.00	
14	J631							
15	J651							
16								
17								
18								
19								
20								
21								

Discount rate entered by user

Discount price calculated for model H290

## Summary

- Declaring numeric variables
- Types of numeric variables
- Programming a worksheet ListBox control event procedure
- Using a worksheet function in a procedure
- <https://www.youtube.com/watch?v=BCss2QMSIM4>

## 8. Program design – VBA control structures

Structured design

- Selection control structure
  - If-then-else control structure
  - Select Case control structure

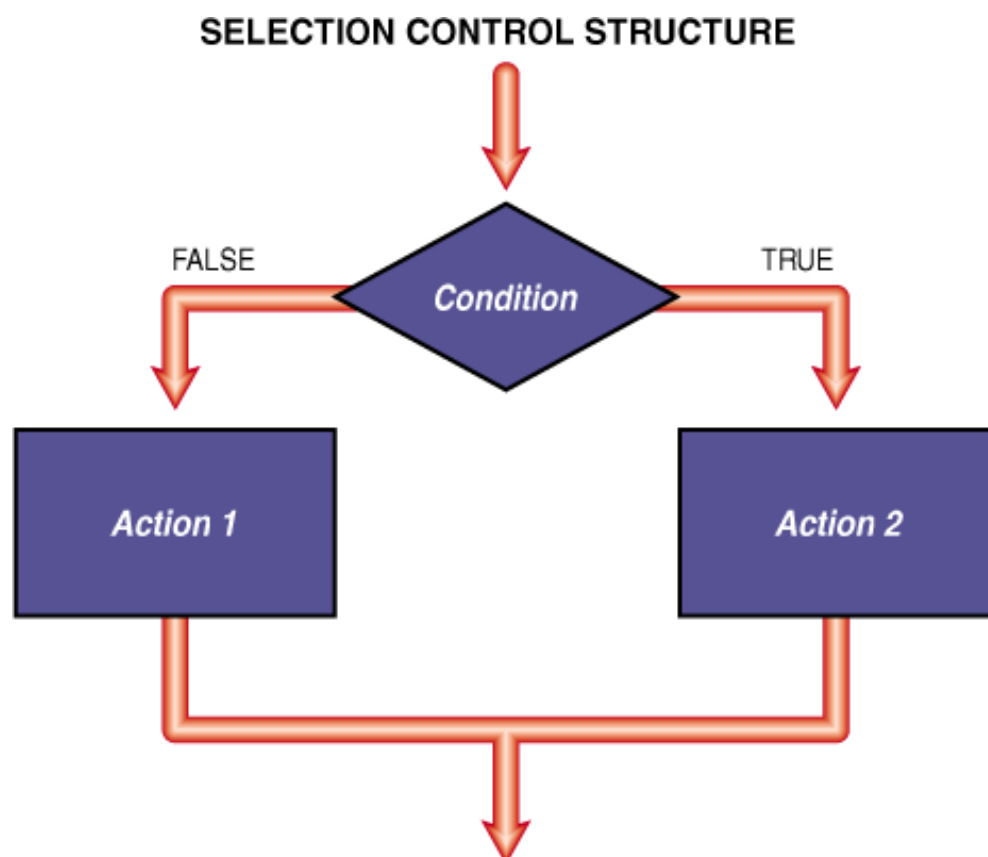
Repetition control structure

- Do-while control structure
- Do-until control structure
- For....Next
- For Each....Next

### Control Structures: If...Then...Else

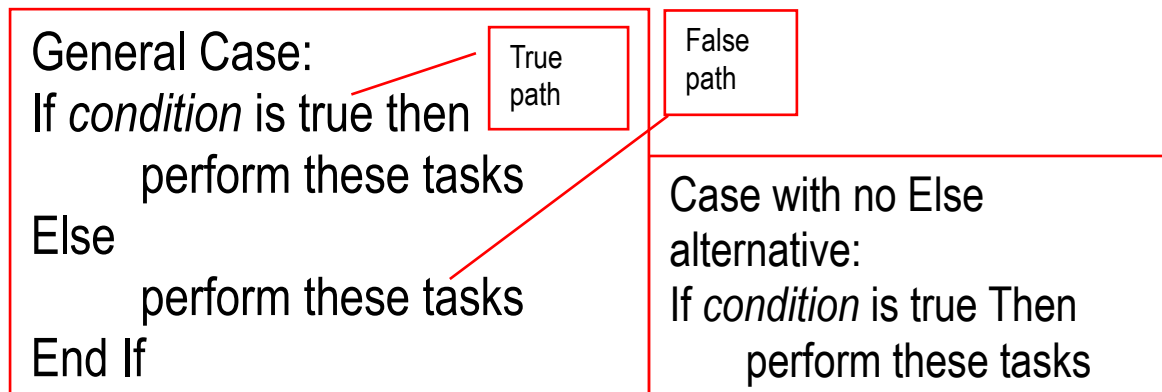
Objectives:

- Perform selection using the **If...Then...Else** statement
- Write instructions that use comparison operators and logical operators
- Use the **UCase** function
- Use the nested **If...Then...Else** statement



## The Selection Structure Pseudocode

- You use the **selection structure**, also called the **decision structure**, when you want a procedure to make a decision or comparison and then, based on the result of that decision or comparison, select one of two paths
- You can use the VBA **If...Then...Else statement** to include a selection structure in a procedure



## Using the If...Then...Else Statement

### If *condition* Then

*[Then clause instructions, which will be processed when the condition evaluates to true]*

### [Else

*[Else clause instructions, which will be processed when the condition evaluates to false]]*

### End If

- The items appearing in square brackets ([ ]) in the syntax are optional
- The remaining components are essential
  - I.e. the words, **If**, **Then**, and **End If** must be included in the statement
- Items in *italics* indicate where the programmer must supply information pertaining to the current procedure
- The **If...Then...Else** statement's *condition* can contain variables, constants, functions, arithmetic operators, comparison operators, and logical operators

## 9. Comparison Operators

- You use **comparison operators**, sometimes referred to as relational operators, to compare two values
- When a condition contains more than one comparison operator, the comparison operators are evaluated from left to right in the condition
- Comparison operators are evaluated after any arithmetic operators

### Relational Operators (Comparison Operators)

=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

These operators are evaluated from left to right, and are evaluated after any mathematical operators.

### Numeric Operator Order of Precedence

^	exponentiation
-	negation
*, /	multiplication and division
Mod	modulus arithmetic
+, -	addition and subtraction

You can use parentheses to override the order or precedence.

### Examples of Expressions Containing Relational Operators

$10 + 3 < 5 * 2$

- $5 * 2$  is evaluated first, giving 10
- $10 + 3$  is evaluated second, giving 13
- $13 < 10$  is evaluated last, giving false

$7 > 3 * 4 / 2$

- $3 * 4$  is evaluated first, giving 12
- $12 / 2$  is evaluated second, giving 6
- $7 > 6$  is evaluated last, giving true

NB: All expressions containing a relational operator will result in either a true or false answer only

### Comparison Operators – more examples using If Then ...Else

If Then ...Else statement	Result
If intQuantity < 25 Then MsgBox Prompt:= "Reorder" End If	Displays "Reorder" if the intQuantity variable contains a value less than 25
If sngHours <= 40 Then MsgBox Prompt:= "Regular Pay" Else MsgBox Prompt:= "Overtime Pay" End If	Displays "Regular Pay" if the sngHours variable contains a value less than or equal to 40. Otherwise the message "Overtime pay" is displayed.
If curSales > 1000 Then curBonus = curSales * .1 Else curBonus = curSales * .05 End If	Calculates a 10% bonus on sales that are greater than \$1000, otherwise calculates a 5% bonus.

### Examples of Relational Operators used in the *condition*

1. Write a *condition* that checks if the value stored in the intNum variable is greater than 123  
intNum > 123
2. Write a *condition* that checks if the value stored in the strName variable is "YEN CHEUNG"  
strName = "YEN CHEUNG"

## 10. UCase Function

- String comparisons in VBA are **case sensitive**, which means that the uppercase version of a letter is not the same as its lowercase counterpart
  - E.g. "YEN" is not the same as "Yen"
- The UCase function
  - UCase(String:=string)
  - Returns the uppercase equivalent of string
- The UCase function is useful if you don't wish to discriminate between upper and lower case
  - E.g. if you want "Y" and "y" to be equivalent.
- You can also use the UCase function in an assignment statement to convert to upper case

string is the name of the parameter

e.g. UCase(String:=strName)

e.g. UCase(String:= "Yen Cheung") returns "YEN CHEUNG"

e.g. strName = UCase(String:=strName)

### Also ....LCase function

LCase Function Example:

This example uses the **LCase** function to return a lowercase version of a string.

```
Dim strUpperCase As String
```

```
Dim strLowerCase As String
```

```
strUpperCase = "Hello World 1234"
```

```
strLowerCase = LCase(strUpperCase)
```

String to convert.

Returns "hello world 1234".

### Examples of If...Then...Else Statements Whose Conditions Contain the UCase Function

```
If UCase(strAns) = "Y" Then
```

```
    MsgBox "answered yes"
```

```
End if
```

Displays "answered yes" if the contents of strAns is "y" or "Y"

```
If UCase(strAns) = "Y" Then
```

```
    intYes = intYes + 1
```

```
Else
```

```
    intNo = intNo + 1
```

```
End if
```

Adds 1 to intYes if the contents of strAns is "y" or "Y",  
Otherwise Adds 1 to intNo

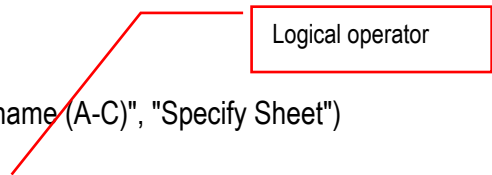


### Example: If Then Else, Ucase - pseudocode

- Ask the user to enter a sheet name - either 'A', 'B' or 'C'
  - If the user enters one of 'A', 'B' or 'C', in either upper case or lower case, then display the worksheet corresponding to that name
  - If the user enters something other than 'A', 'B' or 'C', provide a message asking them to enter a valid name
- [IfThenElse and Select case.xls](#)

### Example: If Then Else, Ucase

```
Public Sub ChooseSheet()  
Dim strChoice As String  
strChoice = InputBox("Enter a sheet name (A-C)", "Specify Sheet")  
strChoice = UCase(strChoice)  
If strChoice = "A" Or strChoice = "B" Or strChoice = "C" Then  
    ActiveWorkbook.Worksheets(strChoice).Select  
Else  
    MsgBox "Please enter a letter from A to C.", vbCritical, "Invalid Name"  
End If  
End Sub
```



### Logical Operators

Operator	Meaning	Order of Precedence
And	All <i>conditions</i> connected by the And operator must be true for the compound <i>condition</i> to be true	1
Or	Only one of the <i>conditions</i> connected by the Or operator needs to be true for the compound <i>condition</i> to be true	2

Most commonly used logical operators

- The two most commonly used logical operators are **And** and **Or**
- You use the **And** and **Or** operators to combine several conditions into one compound condition

## Logical Operators

**Not** :Reverses the truth value of *condition*; false becomes true and true becomes false

**And**: All *conditions* connected by the And operator must be true for the compound *condition* to be true

**Or**: Only one of the *conditions* connected by the Or operator needs to be true for the compound *condition* to be true.

When a ***condition*** contains arithmetic, comparison, and logical operators:

the arithmetic operators are evaluated first

then the comparison operators are evaluated

and then the logical operators are evaluated.

**The order of precedence is Not, And, Or.**

Truth Table for the And Operator

Value of <i>condition1</i>	Value of <i>condition2</i>	Value of <i>condition1</i> And <i>condition2</i>
True	True	True
True	False	False
False	True	False
False	False	False

- All compound conditions containing a logical operator will evaluate to an answer of either True or False only

**Truth Table for the Or Operator**

<b>Value of <i>condition1</i></b>	<b>Value of <i>condition2</i></b>	<b>Value of <i>condition1</i> Or <i>condition2</i></b>
True	True	True
True	False	True
False	True	True
False	False	False

- All compound conditions containing a logical operator will evaluate to an answer of either True or False only

**Truth Table for the Not Operator**

<b>Value of <i>condition</i></b>	<b>Value of Not <i>condition</i></b>
True	False
False	True

- All compound conditions containing a logical operator will evaluate to an answer of either True or False only

## Examples of If...Then...Else Statements Whose Conditions Contain Logical Operators

### e.g. 1

```
If intQuantity > 0 And intQuantity < 101 Then
  MsgBox Prompt:="Valid Quantity"
Else
  MsgBox Prompt:="Quantity error"
End If
```

### e.g. 2

```
If intCode = 1 Or intCode = 3 Then
  strRaise = "Y"
Else
  strRaise = "N"
End If
```

## Logical Operators – order of precedence example

<b>Condition: <math>6 / 3 &lt; 2 \text{ Or } 2 * 3 &gt; 5</math></b>	
<b>Evaluation steps:</b>	<b>Result of evaluation:</b>
6 / 3 is evaluated first	$2 < 2 \text{ Or } 2 * 3 > 5$
$2 * 3$ is evaluated second	$2 < 2 \text{ Or } 6 > 5$
$2 < 2$ is evaluated third	False Or $6 > 5$
$6 > 5$ is evaluated fourth	False Or True
False Or True is evaluated last	True

|                      **Evaluation steps for a *condition* containing arithmetic, comparison, and logical operators**

## Expressions Containing the And Logical Operator

$3 > 2 \text{ And } 6 > 5$

- $3 > 2$  is evaluated first, giving true
- $6 > 5$  is evaluated second, giving true
- true And true is evaluated last, giving true

$10 < 25 \text{ And } 6 > 5 + 1$

- $5 + 1$  is evaluated first, giving 6
- $10 < 25$  is evaluated second, giving true
- $6 > 6$  is evaluated third, giving false
- true And false is evaluated last, giving false

### Expression Containing the Or Logical Operator

$8 = 4 * 2 \text{ Or } 7 < 5$

- $4 * 2$  is evaluated first, giving 8
- $8 = 8$  is evaluated second, giving true
- $7 > 5$  is evaluated third, giving false
- true Or false is evaluated last, giving true

All expressions containing a relational operator will result in either a true or false answer only.

### Example of Logical Operators used in the *condition*

- To pass a course, a student must have an average test score of at least 75 and an average project score of at least 35. Write the condition using the variables sngTest and sngProj.

$\text{sngTest} \geq 75 \text{ And } \text{sngProj} \geq 35$

### Example of Logical Operators used in the *condition*

- Only people living in the state of Victoria who are over 60 years old receive a discount. Write the *condition* using the variables strState and intAge.

$\text{strState} = \text{"Victoria"} \text{ And } \text{intAge} > 60$

- Only employees with job codes of 12 or 18 will receive a raise. Write the condition using the variable intCode.

$\text{intCode} = 12 \text{ Or } \text{intCode} = 18$

## 11. Nested Selection Structure

- A nested selection structure is one in which either the true path or the false path includes yet another selection structure.
- Any of the statements within either the true or false path of one selection structure may be another selection structure.

### Nested If in the true path

```
If condition1 Then
    [instructions when condition1 is true]
    If condition2 Then
        [instructions when both condition1 and condition2 are true]
    Else
        [instructions when condition1 is true and condition2 is false]
    End If
Else
    [instructions when condition1 is false]
End If
```

The diagram illustrates a nested if structure where the second condition is nested within the true path of the first condition. A red box labeled "Nested If" points to the inner if-else block. Another red box labeled "True path" points to the true branch of the inner if statement.

### Example of Nested If in the true path

```
If intAge >= 18 Then
    If UCase(string:= strRegistered) = "Y" Then
        MsgBox prompt:="You can vote"
    Else
        MsgBox prompt:="You need to register before you can vote"
    End If
Else
    MsgBox prompt:="You are too young to vote"
End If
```

The diagram shows an example of a nested if structure. A red box labeled "Nested If" points to the inner if-else block. Another red box labeled "True path" points to the true branch of the inner if statement.

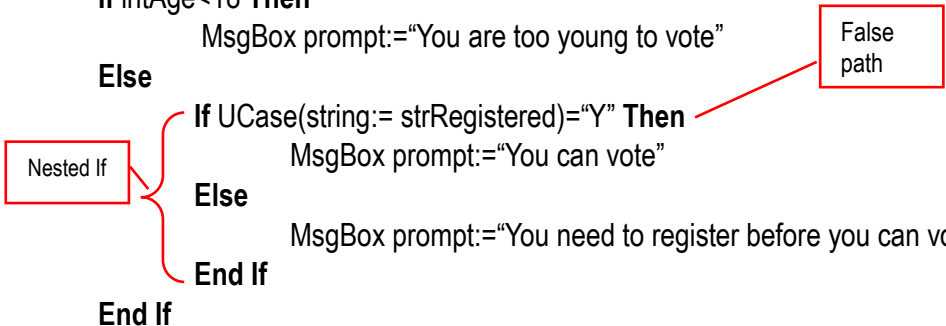
### Nested If in the false path

```
If condition1 Then
    [instructions when condition1 is true]
Else
    If condition2 Then
        [instructions when condition1 is false and condition2 is true]
    Else
        [instructions when both condition1 and condition2 are false]
    End If
End If
```

The diagram illustrates a nested if structure where the second condition is nested within the false path of the first condition. A red box labeled "Nested If" points to the inner if-else block. Another red box labeled "False path" points to the false branch of the outer if statement.

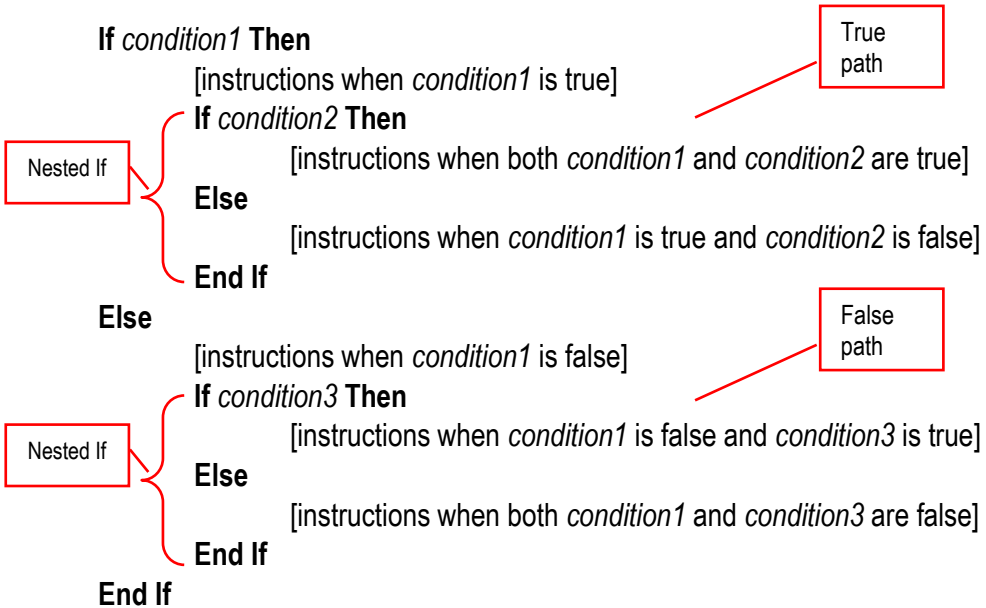
### Example of Nested If in the false path

```
If intAge < 18 Then
    MsgBox prompt:="You are too young to vote"
Else
    If UCase(string:= strRegistered)="Y" Then
        MsgBox prompt:="You can vote"
    Else
        MsgBox prompt:="You need to register before you can vote"
    End If
End If
```



### Nesting If...Then...Else Statements

```
If condition1 Then
    [instructions when condition1 is true]
    If condition2 Then
        [instructions when both condition1 and condition2 are true]
    Else
        [instructions when condition1 is true and condition2 is false]
    End If
Else
    [instructions when condition1 is false]
    If condition3 Then
        [instructions when condition1 is false and condition3 is true]
    Else
        [instructions when both condition1 and condition3 are false]
    End If
End If
```



## Alternate version of If Then Else

- Uses an **Elseif**
- Avoids nesting

Syntax:

**If** *condition1* **Then**

[instructions when *condition1* is true]

**[Elseif** *condition2* **Then**

[instructions when *condition2* is true]]

.....

**[Elseif** *conditionn* **Then**

[instructions when *conditionn* is true]]

**[Else**

[default instructions when all conditions are false]]

**End If**

Elseif can be included for any number of alternatives

## Example of Alternate version of If Then Else

**If** intAge<18 **Then**

MsgBox prompt:="You are too young to vote"

**Elseif** UCase(string:= strRegistered)="Y" **Then**

MsgBox prompt:="You can vote"

**Else**

MsgBox prompt:="You need to register before you can vote"

**End If**

## Summary – If Then Else

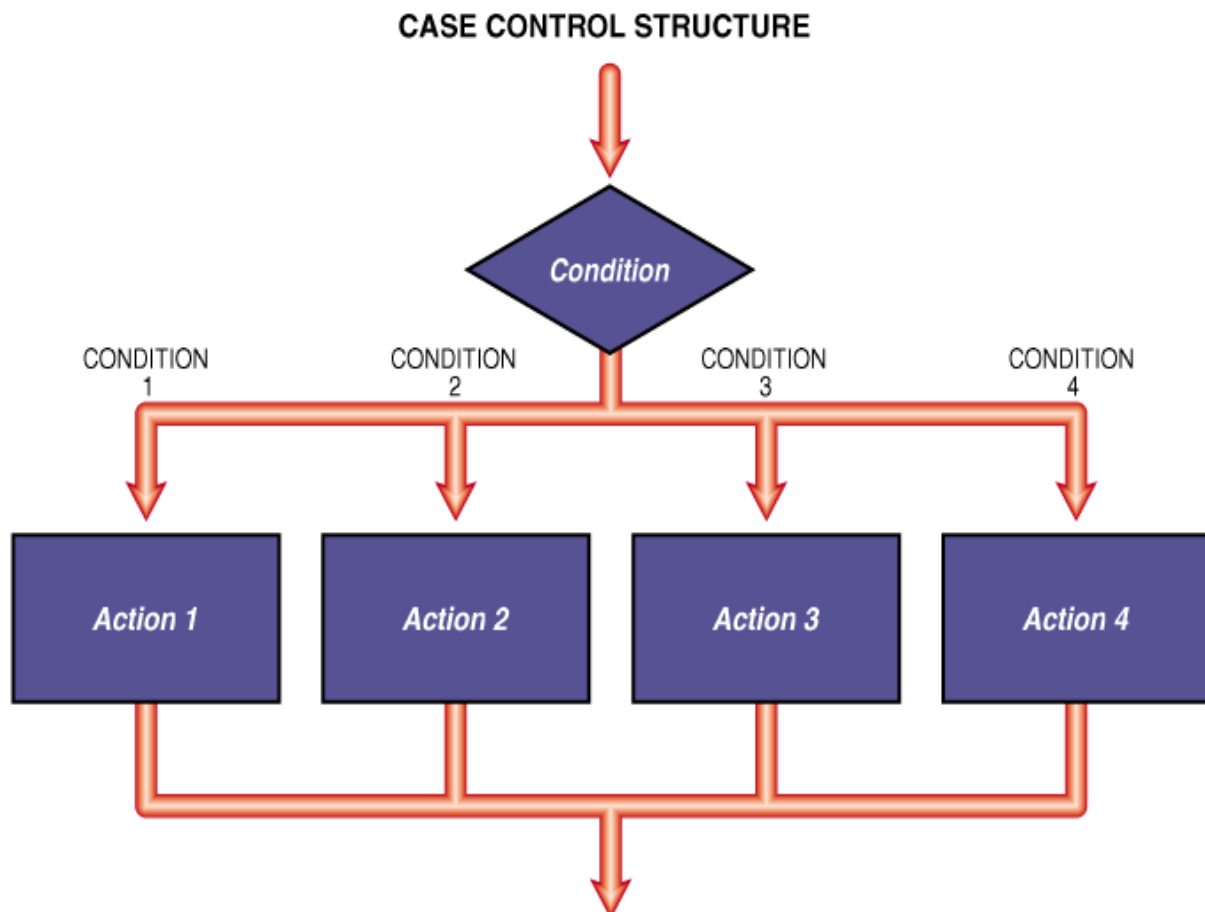
- To use the **If...Then...Else** statement to code the selection structure:
- Use the syntax shown in slide 6, where condition can contain variables, constants, functions, arithmetic operators, comparison operators, and logical operators
- To compare two values:
  - Use the comparison operators  
(=, >, <, >=, <=, <>)
- To return the uppercase equivalent of a string:
- Use the UCase function, whose syntax is UCase(String:=string)
- To return the lower case equivalent of a string:
- Use the LCase function, whose syntax is LCase(String:=string)
- To create a compound condition:
- Use the logical operators (And and Or)
- To nest If...Then...Else statements:



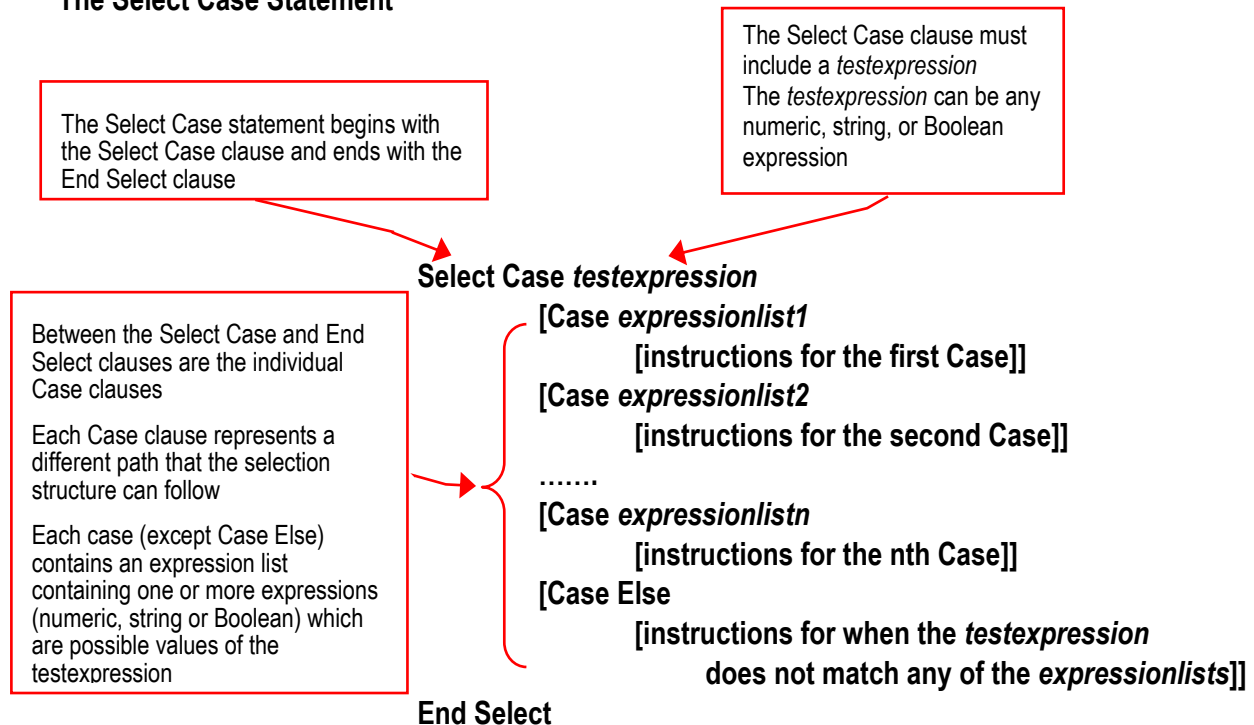
## The Case Form of the Selection Structure

The Select Case control structure:

- When you have more than two paths in your program design, an extended selection structure such as the **Case** statement can be used.
- It is usually simpler, clearer and easier to use the **Case** form of the selection structure instead of the nested **If** form



## The Select Case Statement



- Each of the individual **Case** clauses, except the **Case Else**, must contain an expression list, which can include one or more numeric, string, or Boolean expressions
- The data type of the expressions included in the expression lists must be compatible with the data type of the test expression
- When the Select Case statement is processed, the value of the test expression is first compared with the values listed in expression list1
- If a match is found, the corresponding instructions are processed and then the Select Case statement is exited etc....
- You can have a **Case Else** clause at the end of the **Select case** structure that runs if none of the other cases are valid

## Syntax and an Example of the Select Case Statement

Syntax	Example
<pre>Select Case testexpression   [Case expressionlist1     [instructions for the first Case]]   [Case expressionlist2     [instructions for the second Case]]   [Case expressionlistn     [instructions for the nth Case]]   [Case Else     [instructions for when the testexpression     does not match any of the expressionlists]] End Select</pre>	<pre>Select Case strGrade   Case "A"     MsgBox Prompt:="Excellent"   Case "B"     MsgBox Prompt:="Above Average"   Case "C"     MsgBox Prompt:="Average"   Case "D", "F"     MsgBox Prompt:="Below Average"   Case "I"     MsgBox Prompt:="Incomplete"   Case "W"     MsgBox Prompt:="Withdrawal"   Case Else     MsgBox Prompt:="Incorrect Grade" End Select</pre>

**Syntax and an example of the Select Case statement**

## Using To and Is key words in an Expressionlist

- You can use either the keyword **To** or **Is** to specify a range of values in an expressionlist; the values included in the range can be either numeric or a string
- When you use the **To** keyword in a **Case** clause, the value preceding the **To** always must be smaller than the value following the **To**
- Use the **To** keyword to specify a range of values when you know both the minimum and maximum values
- Use the **Is** keyword to specify a range of values when you know only one value, either the minimum or the maximum
- If you neglect to type the keyword **Is** in an expression, the Visual Basic Editor will type it for you

## Example of Select Case

Pseudocode:

1. Prompt the user for their test result out of 100
2. If the result is  $\geq 80$  then grade is HD
3. If the result is  $\geq 70$  then grade is D
4. If the result is  $\geq 60$  then grade is C
5. If the result is  $\geq 50$  then grade is P
6. Else  $< 50$  then N

## Example of Select Case

```
Private Sub CaseEg()  
Dim strMark As String  
Dim intMark As Integer  
strMark = InputBox("What is your mark?", "Mark-Grade conversion")  
intMark = Val(strMark)  
Select Case intMark  
Case Is >= 80  
    MsgBox "Grade is HD"  
Case Is >= 70  
    MsgBox "Grade is D"  
Case Is >= 60  
    MsgBox "Grade is C"  
Case Is >= 50  
    MsgBox "Grade is P"  
Case Else  
    MsgBox "Grade is N, you will have to repeat"  
End Select  
End Sub
```

The *testexpression* is the value of intMark

expressionlist1

### Example of Using the To and Is Keywords in a Select Case Statement

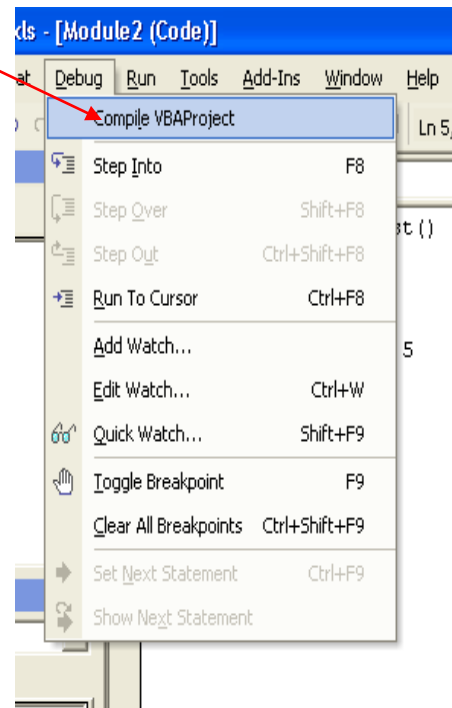
```
Select Case intNumOrdered
    Case 1 To 5
        intPrice = 25
    Case 6 To 10
        intPrice = 23
    Case Is > 10
        intPrice = 20
    Case Else
        intPrice = 0
        MsgBox Prompt:="Incorrect number ordered"
End Select
```

Example of using the To and Is keywords in a Select Case statement

## 12. Compiling your VBA code

Compiling (Debug, Compile ProjectName):

- 
- Converts your procedure to machine language that your computer can understand
- Checks code for errors – more stringent than single-line syntax checker
- E.g. Variables checked for proper reference and type
- E.g. Functions are checked for correct parameters



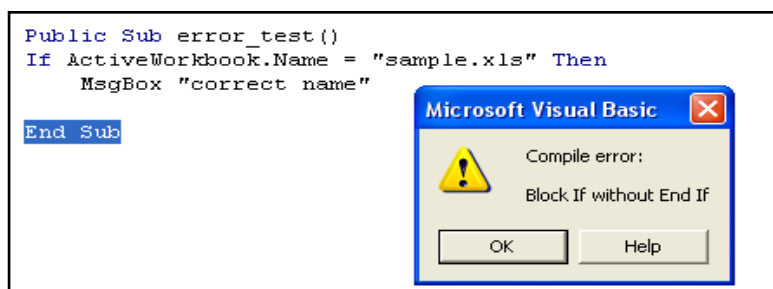
### Types of errors

- Compile errors
- Runtime errors
- Logic errors

### Compile errors

Occur when you don't construct the code correctly.

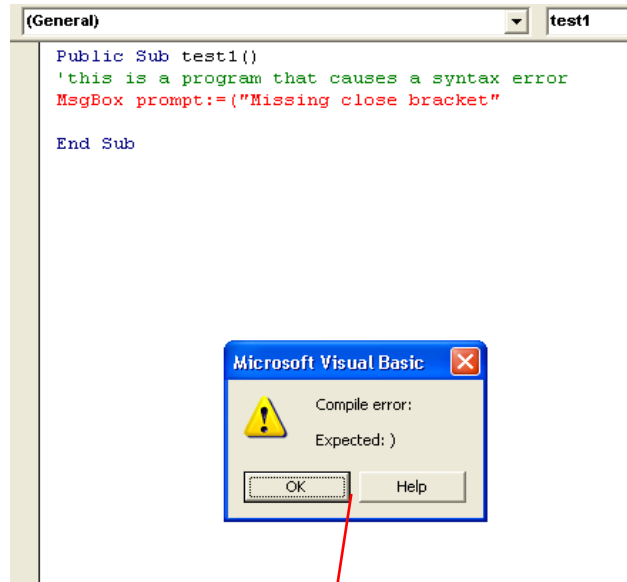
- E.g. if you leave out a bracket.
- E.g. if you use an If without an Endif



Compile error due to missing End if

## Compile errors continued

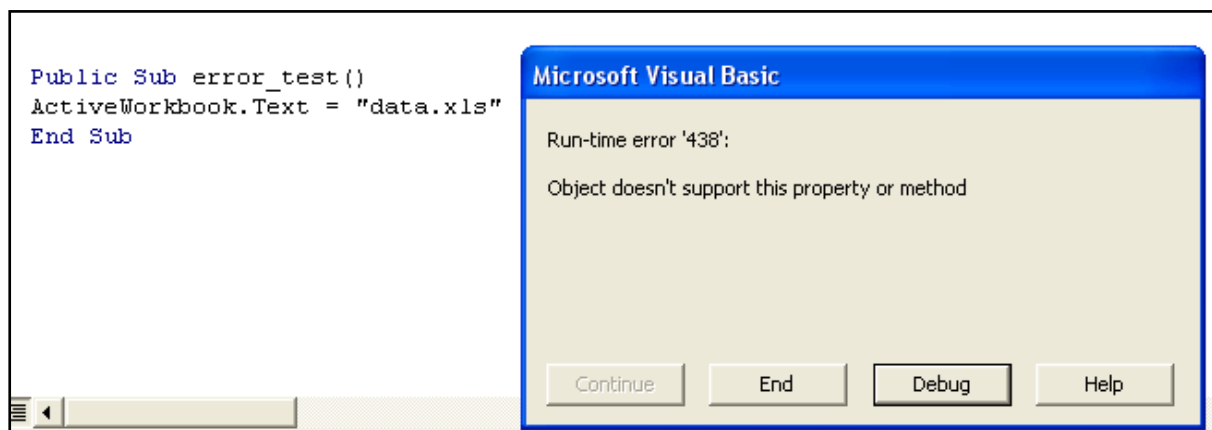
- **Syntax** errors are one type of compile error:
- Syntax = computer grammar
- VBA has a syntax checker that checks after you type each line of code
- The syntax checker finds some syntax errors



## Runtime errors

Occur when your program is running

- E.g. if you try to use a non-existent property or method
- E.g. if one of your calculations results in a divide by zero



## Logic errors

Occur when your program runs but not the way you intended it to. They can be very difficult to locate. They have the potential to survive testing and therefore exist in the final version of your code thus producing erroneous results.

- E.g. 1 not initialising the value of a variable that is used in a calculation can result in a logic error
- E.g. 2

```
Public Sub Logic_error()  
Dim curPrice1 As Currency  
Dim curPrice2 As Currency  
Dim curTotalCost As Currency  
curPrice1 = 56.5  
curPrice2 = 34.8  
curTotalCost = (curPrice1 + curPrice2) * 1.75  
End Sub
```

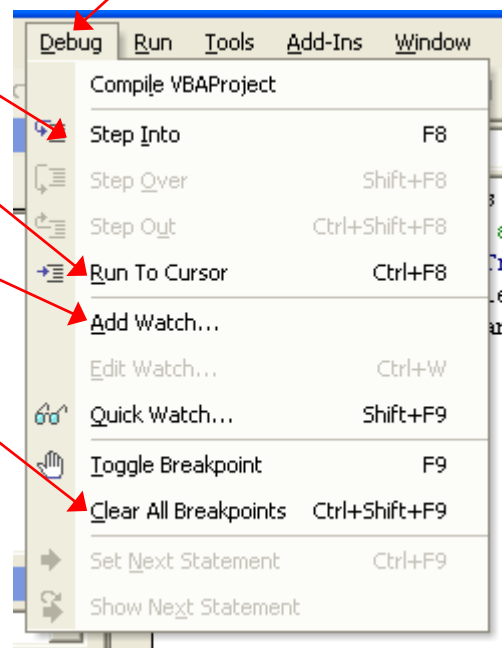
Should be  
1.075

This code is syntactically correct and will compile and run, but the sales tax rate is 7.5%, however the code is using 75% due to a missing 0

## Tools for debugging

- Single stepping through code
- Run to cursor position in code
- Adding a Watch
- Adding breakpoints
- Demo – [debugging.xls](#)

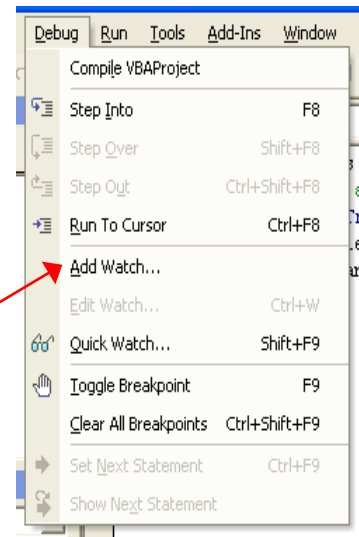
Debug menu in VBE





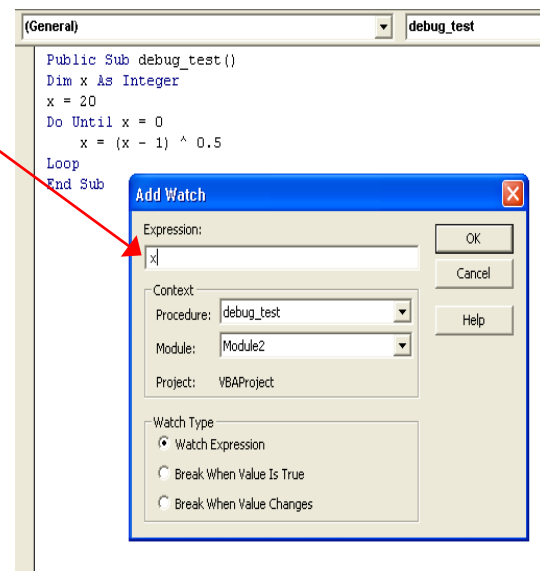
## The Debug Menu – add Watch

- A watch enables you to determine the value of a variable while the program is executing
- You can see the value of variables change as the program progresses
- Use the Add Watch option from the Debug Menu



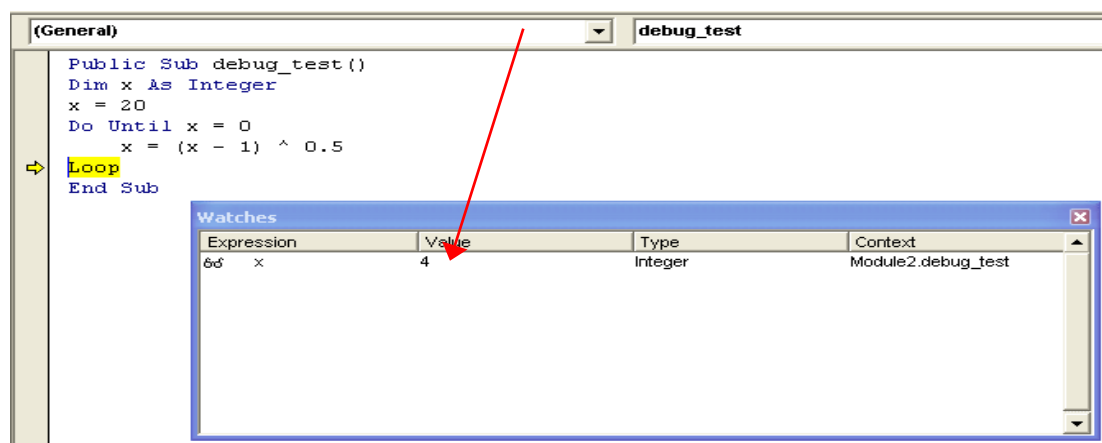
## The Watch Option

- Enter the expression you are interested in then step through the code – the value of the expression is visible in the Watch window [debugging.xls](#)
- (module2)



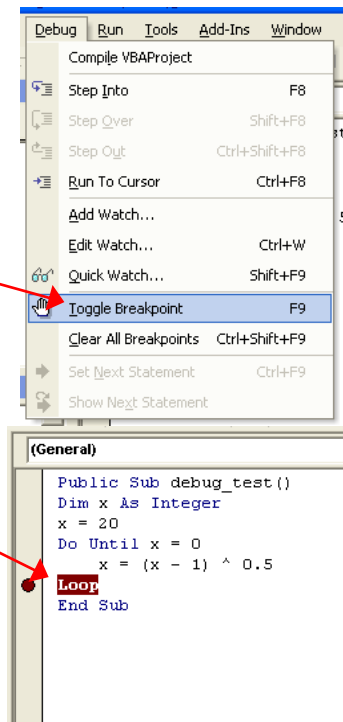
## The Watch Option

The value of the expression (in this case x) is visible in the Watch window [debugging.xls](#) (module2)

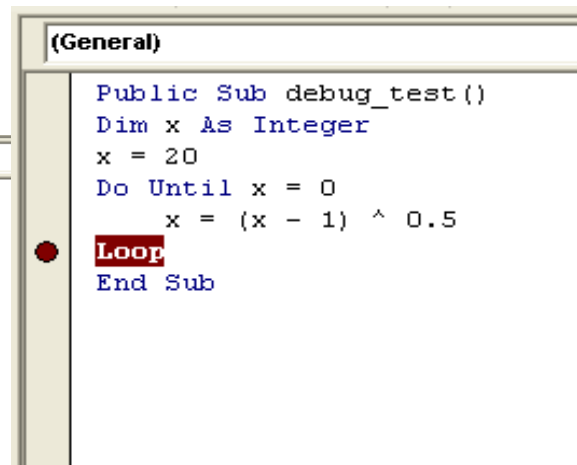
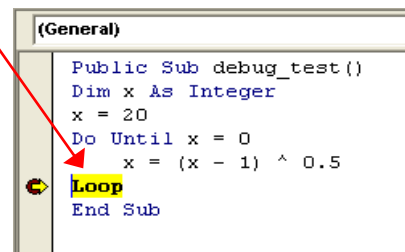


## Adding a break point

- A break is a point where your program stops
- When it stops you can test the value of variables
- You can use breaks to stop at various points where you are interested in the program flow
- You can have as many break points as required

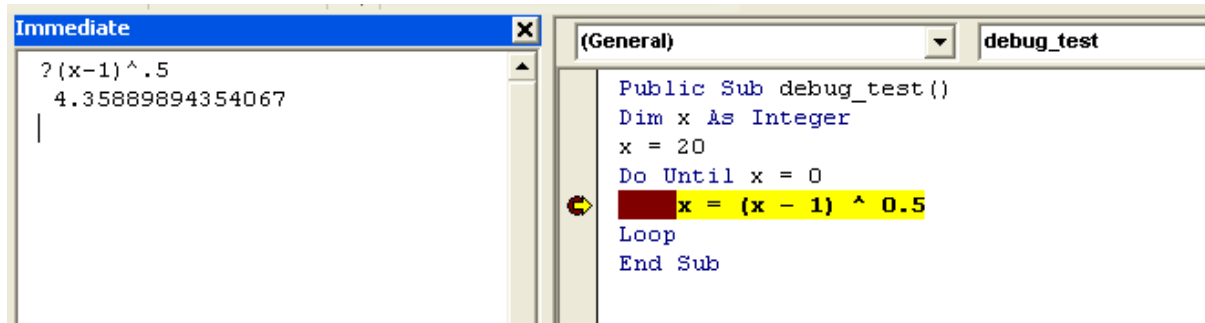


When the break point is reached the break point line is highlighted as shown



## The Debug Menu – the Immediate Window

- View, Immediate Window
- Allows you to evaluate any VBA statement involving the values of variables at the point where execution is stopped
- E.g. [debugging.xls](#)



## Avoiding errors

- Declare all variables
- Always add comments to your code to make it easier for someone to understand the program:
  - Explain what each variable represents
  - Explain what each step is meant to be doing (avoid stating the obvious)
  - Explain what each function does
  - Explain what each calculation does
- Break down your application into small components:
  - Easier to track down errors

## 13. Practice and Apply

- Understanding how to use object variables in Excel
- Understanding how to reserve a numeric variable
- Understanding how to use an assignment statement to assign a value to a numeric variable
- Understanding how to perform calculations using arithmetic operators
- Understanding how to add a list box to an Excel worksheet
- 
- Understanding how to use the Excel VLookup function in a procedure
- Understanding how to perform selection using the
- **If...Then...Else** statement
- Understanding how to write instructions that use comparison operators and logical operators
- Understanding how to use the **UCase** function
- Understanding how to use the nested **If...Then...Else** statement
- Complete Tutorial 6 Exercises