# FIT1013 Digital Futures: IT for Business
# Week 6: Variables and Selection Structures

**On completion of your study this week, you should aim to:**

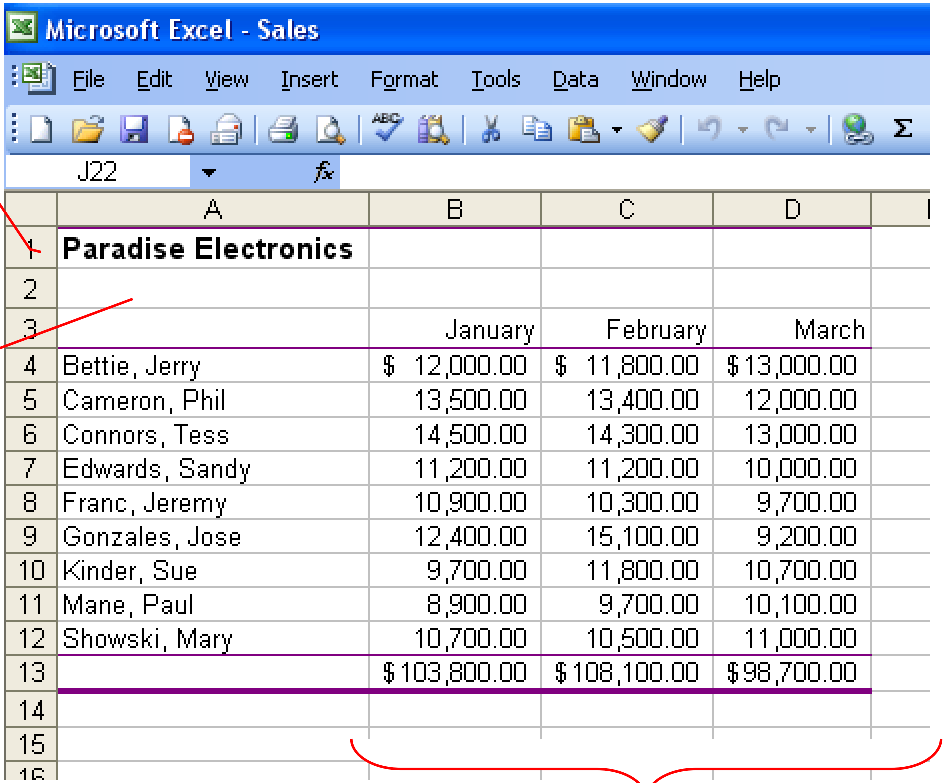
- Use object variables in Excel
- Use an assignment statement to assign a value to a numeric variable
- Add a list box to an Excel worksheet
- Use the Excel VLookup function in a procedure
- Perform selection using the **If…Then...Else** statement
- Write instructions that use comparison operators and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

# Worksheet Format Desired by the District Sales Manager

Cell A1 contains the company name

2 new rows

Accounting2 format

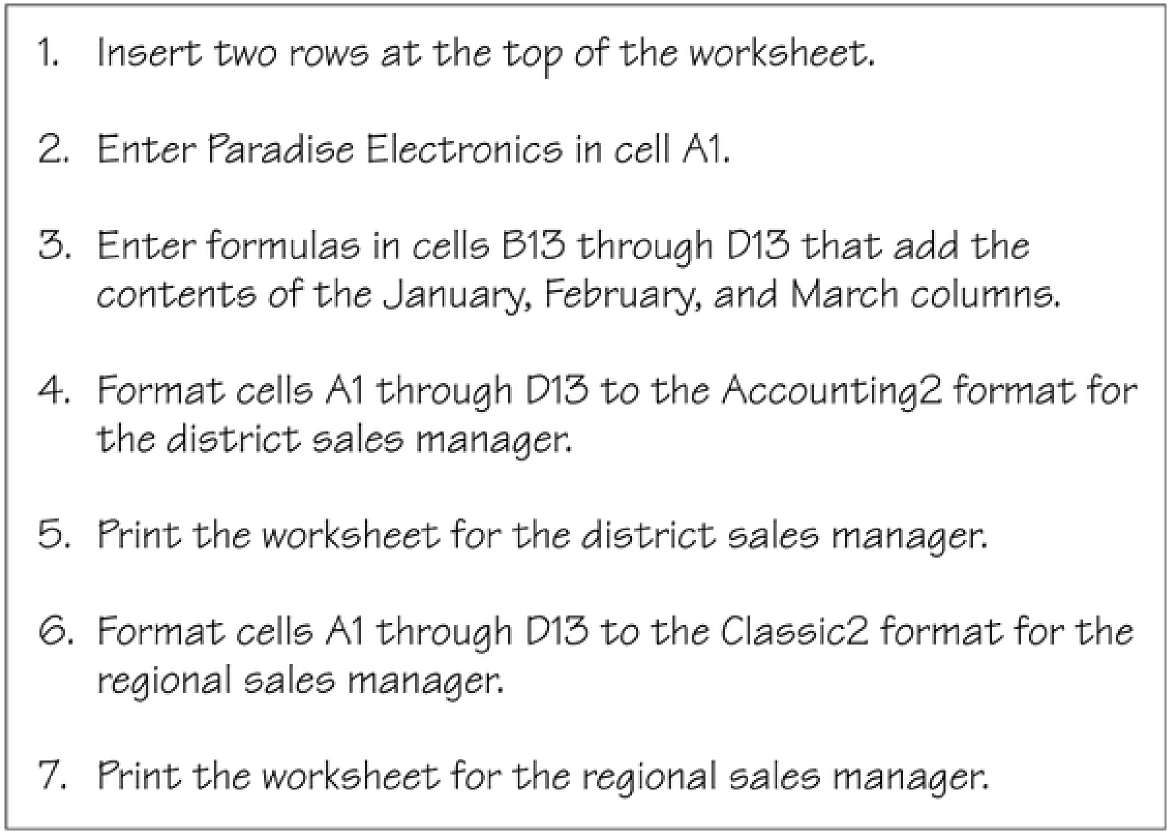


| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Paradise Electronics** | | | |
| 2 | | | | |
| 3 | | January | February | March |
| 4 | Bettie, Jerry | $ 12,000.00 | $ 11,800.00 | $ 13,000.00 |
| 5 | Cameron, Phil | 13,500.00 | 13,400.00 | 12,000.00 |
| 6 | Connors, Tess | 14,500.00 | 14,300.00 | 13,000.00 |
| 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 | | | | |

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

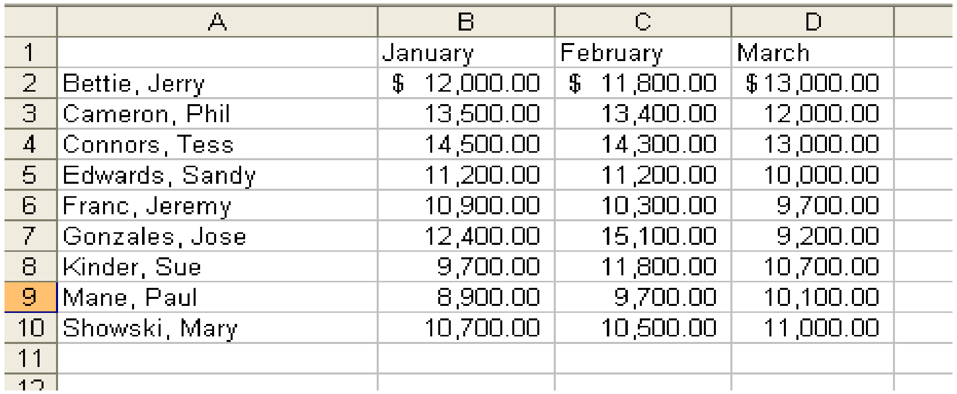


```
1. Insert two rows at the top of the worksheet.
2. Enter Paradise Electronics in cell A1.
3. Enter formulas in cells B13 through D13 that add the
   contents of the January, February, and March columns.
4. Format cells A1 through D13 to the Accounting2 format for
   the district sales manager.
5.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

# 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

Rows property of Worksheet object

Insert method

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**

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**

## Inserting Rows Into a Worksheet

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 Formula Into a Range Object

- 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

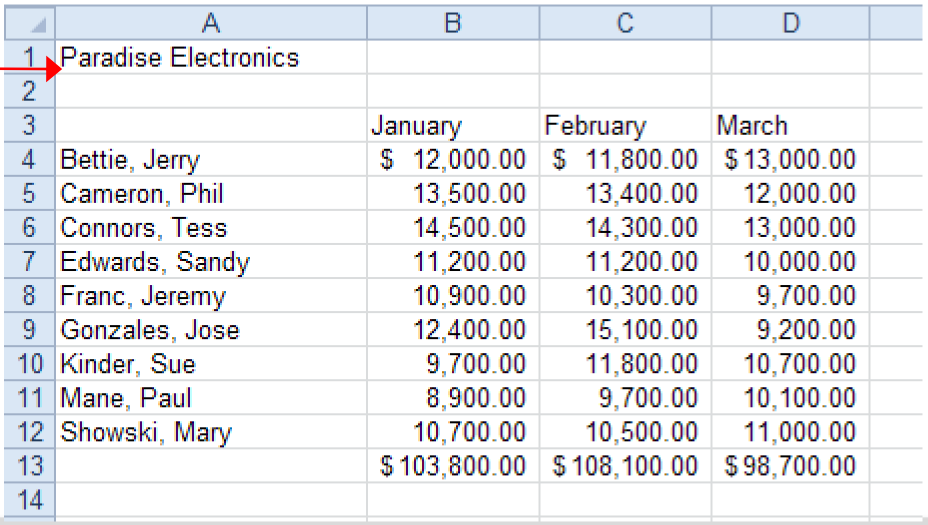
# Entering a value in a range object

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

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

**Range object**

**Value property of 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 | | | | |

MONASH University

## Code so far:

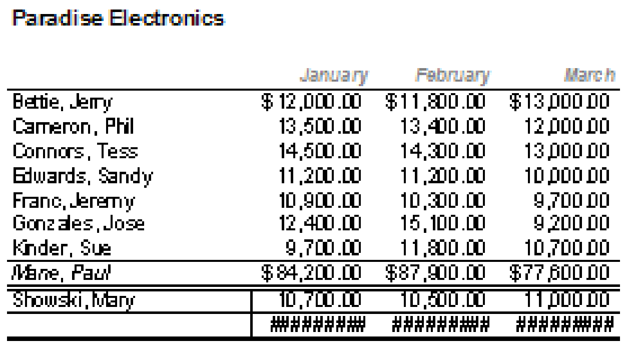
```
Public Sub FormatWorksheet()
    'declare object variable and assign address
    Dim wksFirstQ As Worksheet
    Set wksFirstQ = Application.Workbooks("sales.xls").Worksheets(1)
    'insert 2 rows above row 1
    wksFirstQ.Rows(1).Insert
    wksFirstQ.Rows(1).Insert
  'enter company name
    wksFirstQ.Range("a1").Value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

**Value property of range object**

**Formula property of range object**

# Some examples of AutoFormat formats

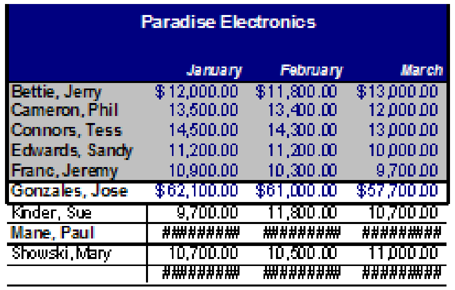
- xlRangeAutoFormatAccounting1



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 |
| Franc, 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 |
| | ######### | ######### | ######### |

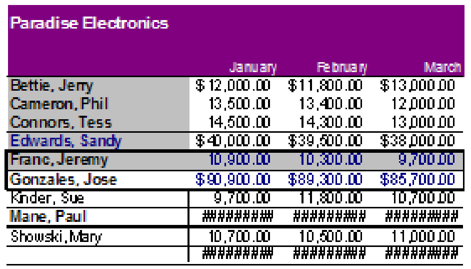
- xlRangeAutoFormatClassic3



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 |
| Franc, 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 |
| Franc, 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

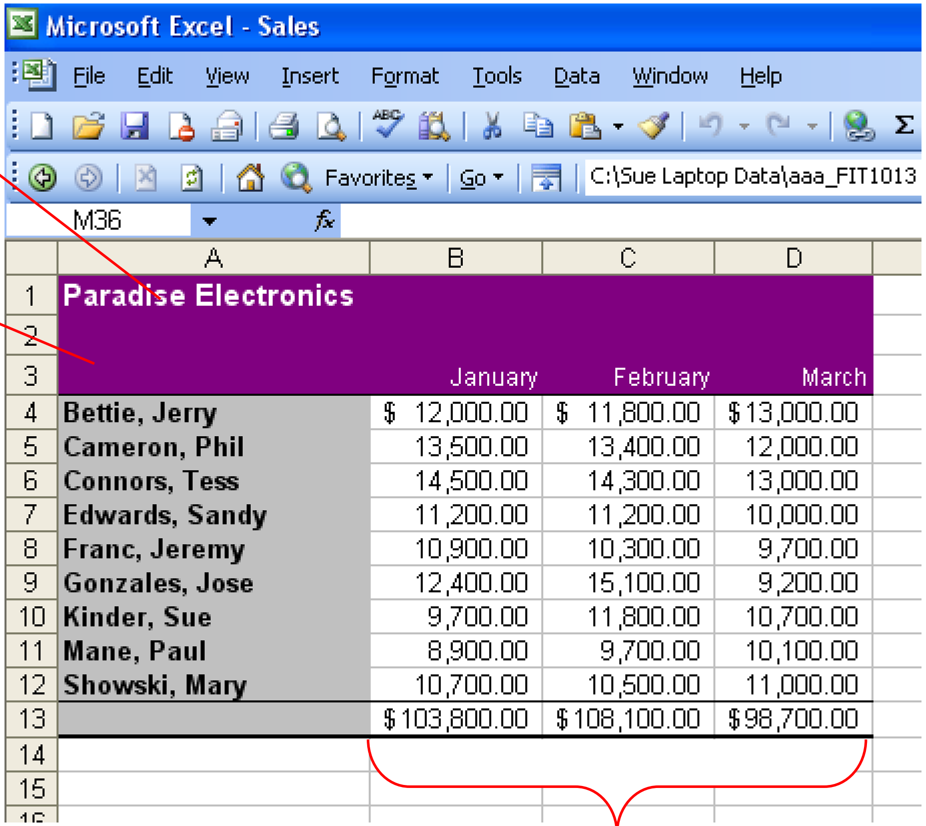
Value of the Format argument

Format argument of the AutoFormat method

## 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


Microsoft Excel - Sales
File Edit View Insert Format Tools Data Window Help
Favorites Go 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 | | | | |

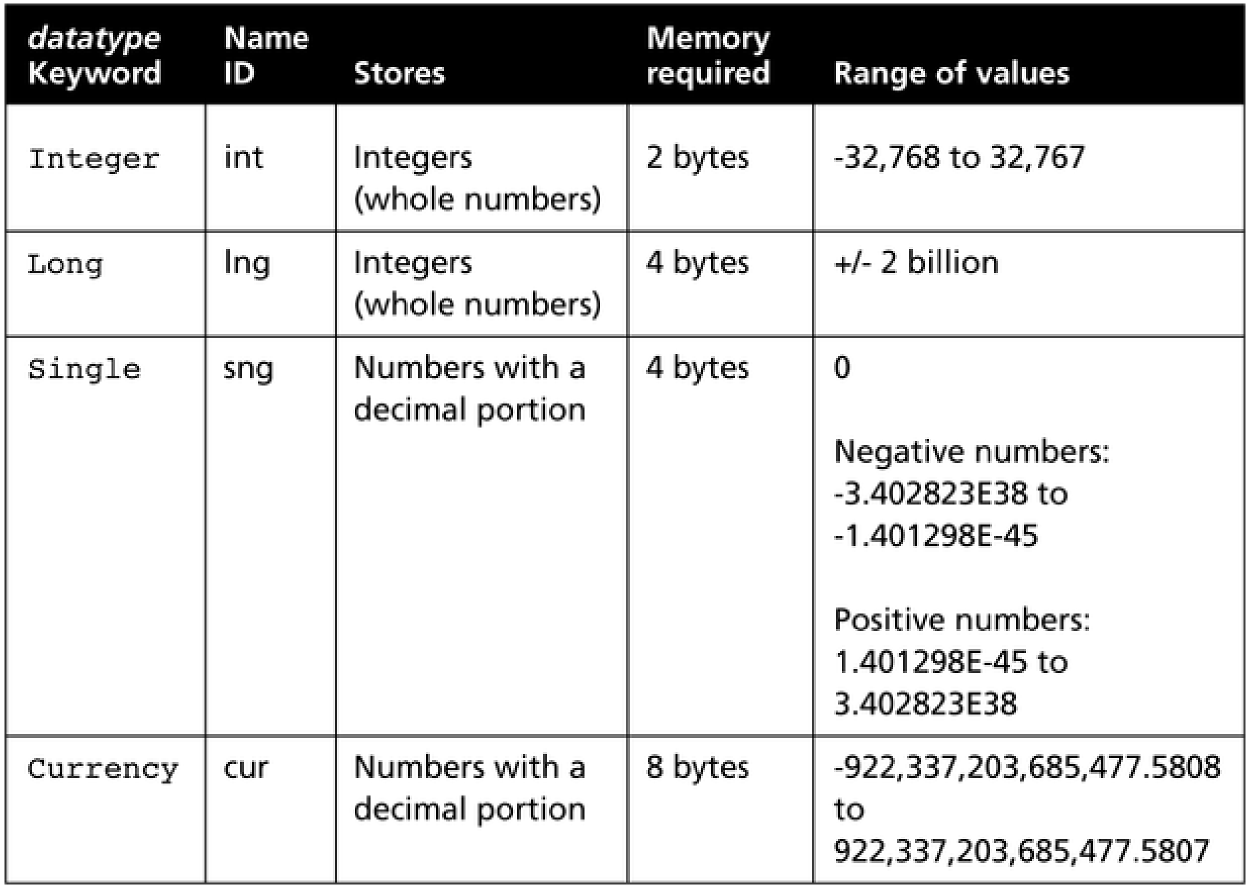
Company name

2 new rows

Classic2 format

These cells contain formulas

# Data Types Used to Reserve Numeric Variables



| *datatype* 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<br><br>Negative numbers:<br>-3.402823E38 to<br>-1.401298E-45<br><br>Positive numbers:<br>1.401298E-45 to<br>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

MONASH University

## 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.

## 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 Expression to a Numeric Variable

- 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:

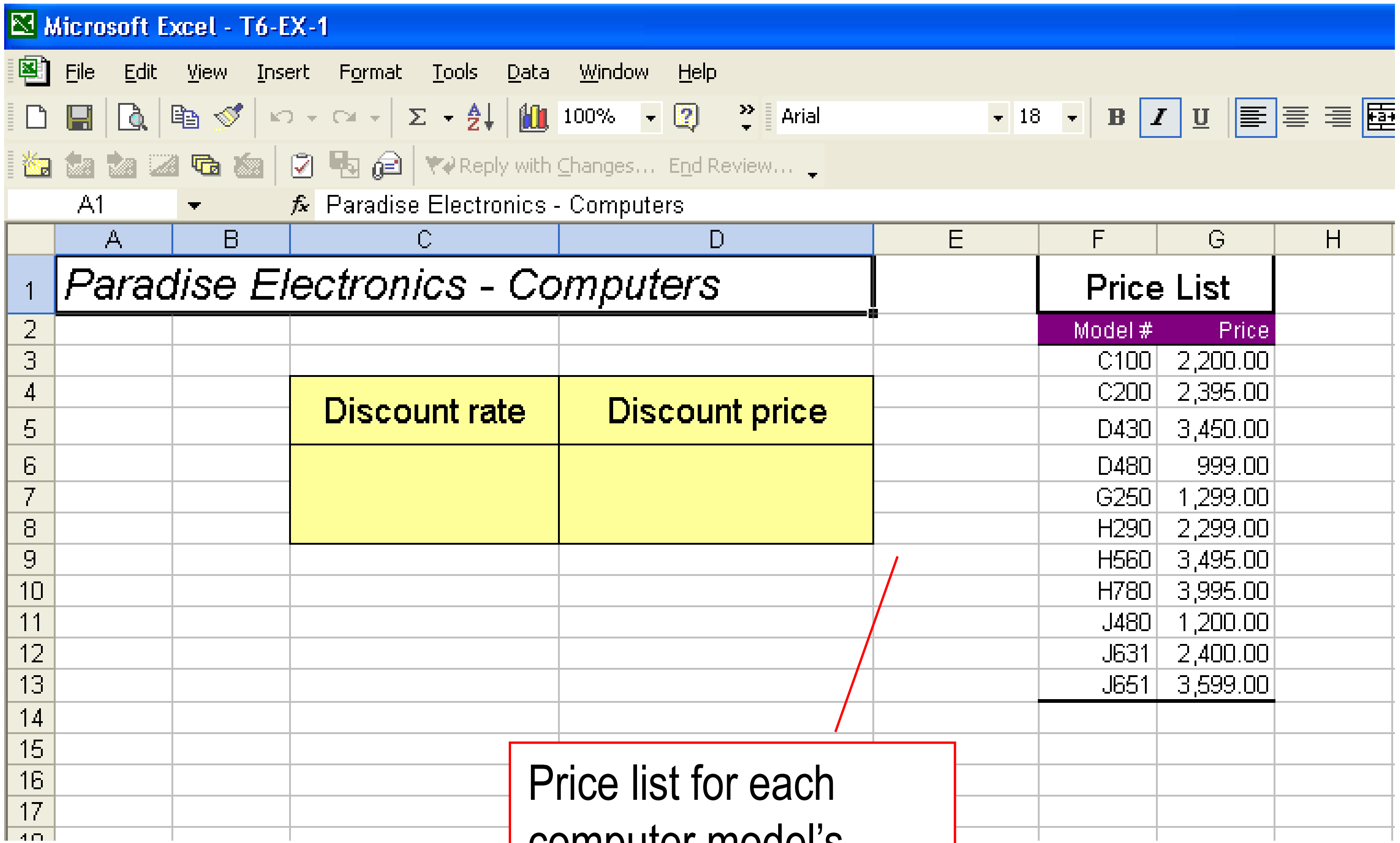
  Use an assignment statement with the following syntax:

  ***variablename*=*value***

  e.g. intAge = 21

## Example: Viewing the Paradise Electronics Price List

### The Computers worksheet


Microsoft Excel - T6-EX-1

A1 Paradise Electronics - Computers

*Paradise Electronics - Computers*

| Discount rate | Discount price |
|---|---|
| | |

Price List

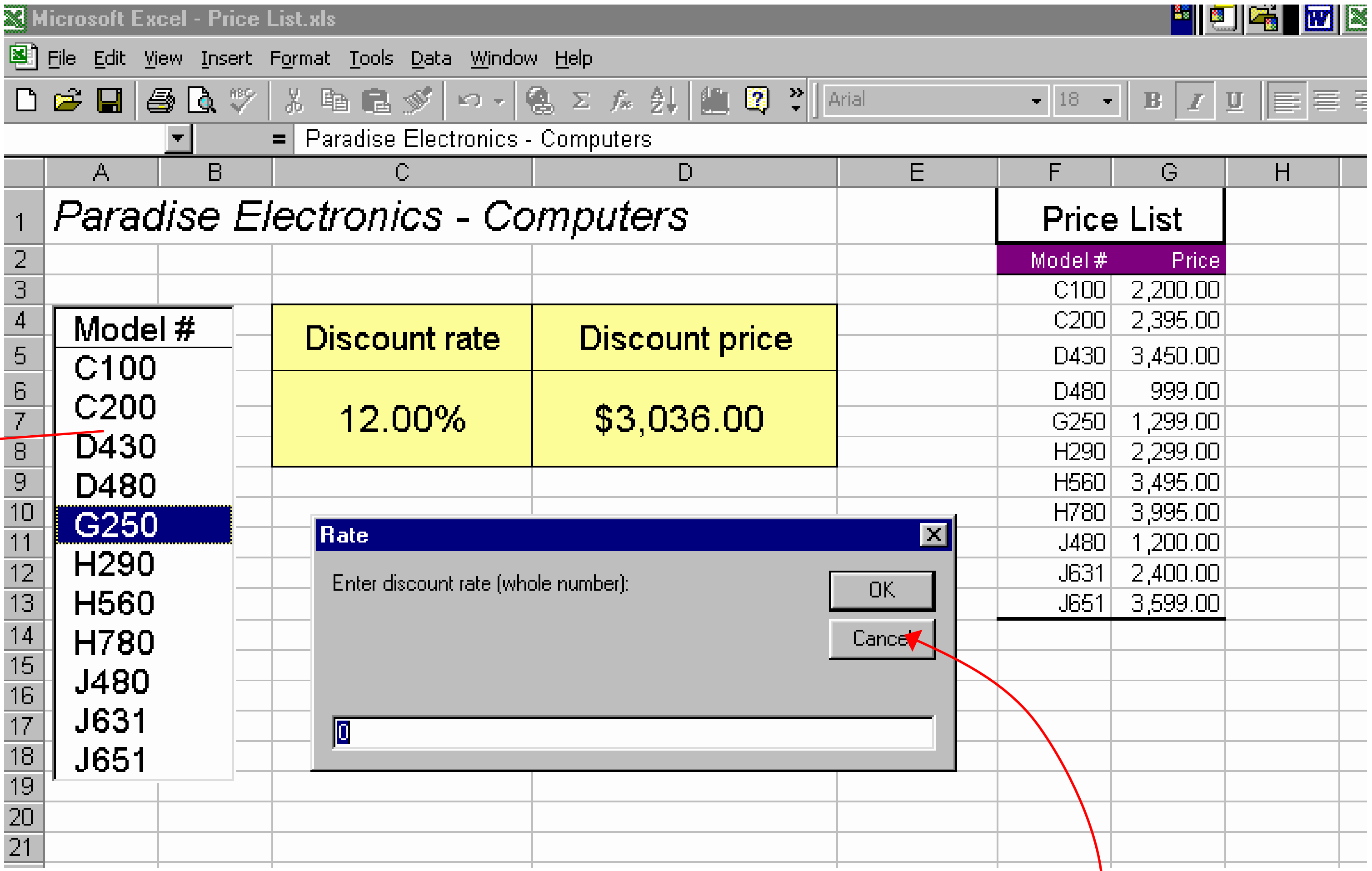
| Model # | Price |
|---|---|
| C100 | 2,200.00 |
| C200 | 2,395.00 |
| D430 | 3,450.00 |
| D480 | 999.00 |
| G250 | 1,299.00 |
| H290 | 2,299.00 |
| H560 | 3,495.00 |
| H780 | 3,995.00 |
| J480 | 1,200.00 |
| J631 | 2,400.00 |
| J651 | 3,599.00 |

Price list for each computer model's price

MONASH University

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

Listbox with contents from the price list


Microsoft Excel - Price List.xls
File Edit View Insert Format Tools Data Window Help
Arial
18
Paradise Electronics - Computers

Paradise Electronics - Computers

| Model # |
|---|
| C100 |
| C200 |
| D430 |
| D480 |
| G250 |
| H290 |
| H560 |
| H780 |
| J480 |
| J631 |
| J651 |

| Discount rate | Discount price |
|---|---|
| 12.00% | $3,036.00 |

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 |

Rate

Enter discount rate (whole number):

OK

Cancel

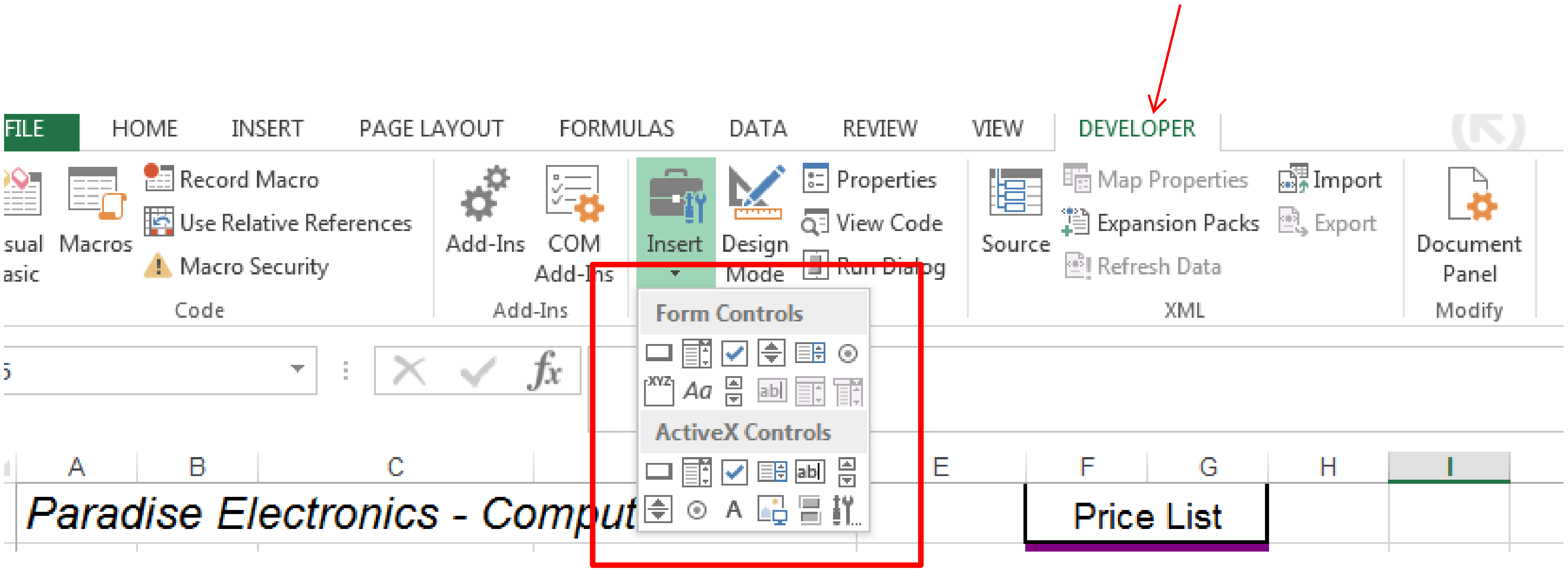
0

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.

MONASH University

# Controls



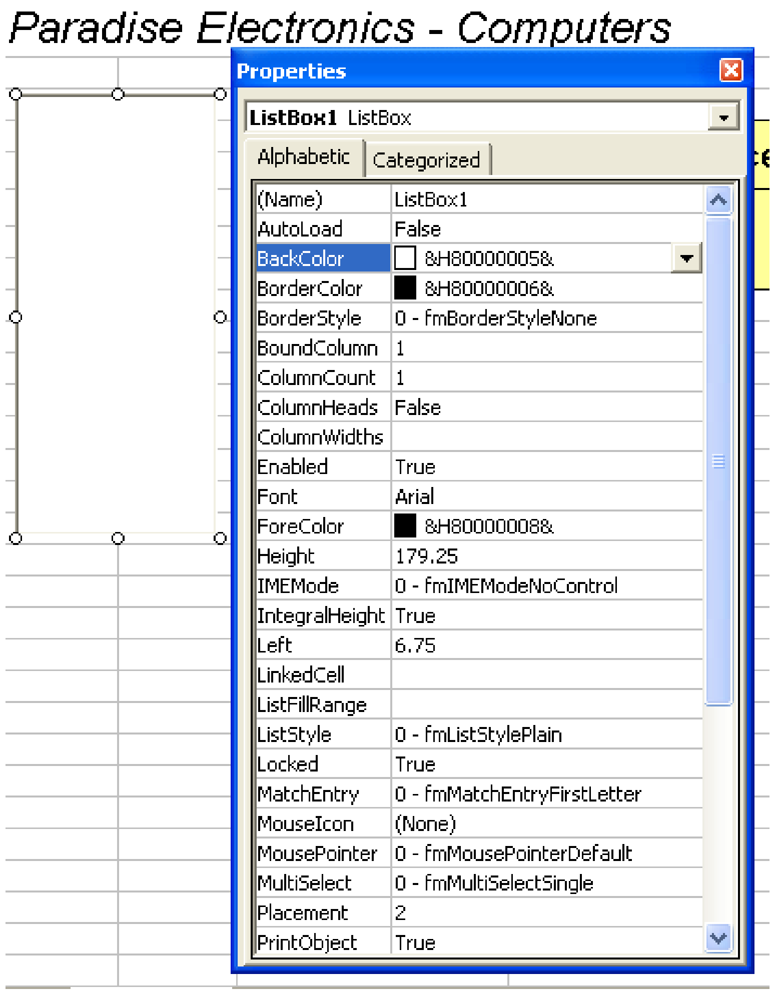
- Form controls
  - Original controls, compatible with earlier versions of Excel, starting from Excel 5.0
- ActiveX controls
  - Use on VBA UserForms and for more flexible design requirements

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

MONASH University

# Common properties for Controls Toolbox controls

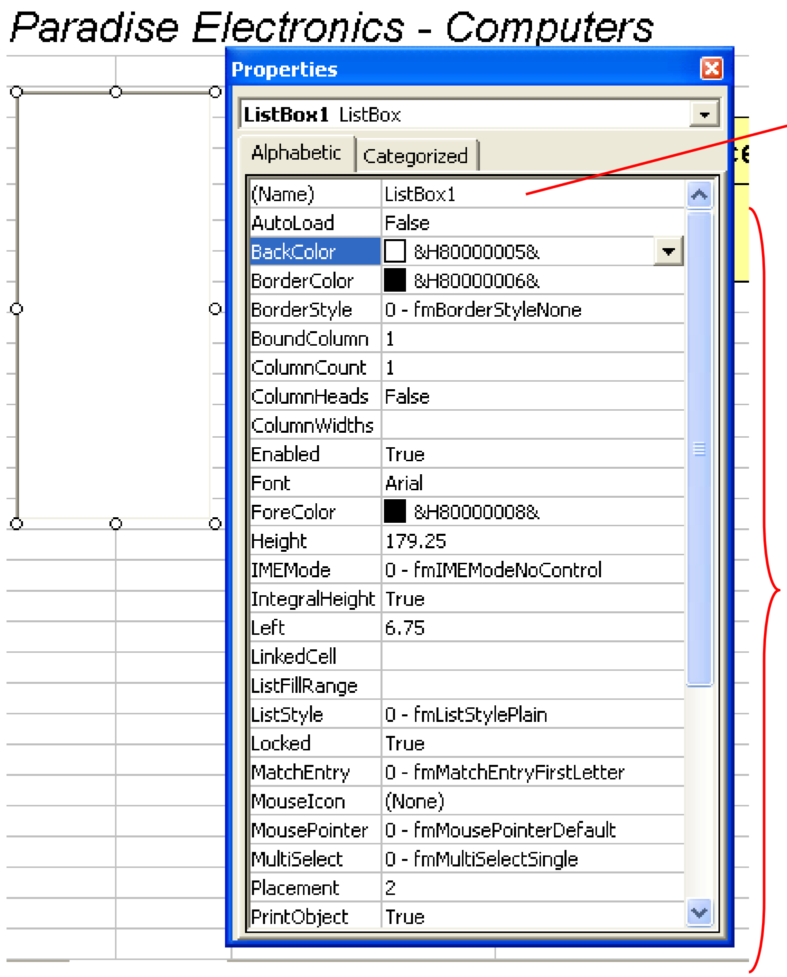
- Name
- Autosize
- Enabled
- Font
- Left, Top, Width, Height
- Linked Cell
- ListFillRange
- PrintObject
- Etc…



ListBox1 ListBox

| | |
|---|---|
| (Name) | ListBox1 |
| AutoLoad | False |
| BackColor | □ &H80000005& |
| BorderColor | ■ &H80000006& |
| BorderStyle | 0 - fmBorderStyleNone |
| BoundColumn | 1 |
| ColumnCount | 1 |
| ColumnHeads | False |
| ColumnWidths | |
| Enabled | True |
| Font | Arial |
| ForeColor | ■ &H80000008& |
| Height | 179.25 |
| IMEMode | 0 - fmIMEModeNoControl |
| IntegralHeight | True |
| Left | 6.75 |
| LinkedCell | |
| ListFillRange | |
| ListStyle | 0 - fmListStylePlain |
| Locked | True |
| MatchEntry | 0 - fmMatchEntryFirstLetter |
| MouseIcon | (None) |
| MousePointer | 0 - fmMousePointerDefault |
| MultiSelect | 0 - fmMultiSelectSingle |
| Placement | 2 |
| PrintObject | True |

# List box Control properties



ListBox1 ListBox

| | |
|---|---|
| (Name) | ListBox1 |
| AutoLoad | False |
| BackColor | □ &H80000005& |
| BorderColor | ■ &H80000006& |
| BorderStyle | 0 - fmBorderStyleNone |
| BoundColumn | 1 |
| ColumnCount | 1 |
| ColumnHeads | False |
| ColumnWidths | |
| Enabled | True |
| Font | Arial |
| ForeColor | ■ &H80000008& |
| Height | 179.25 |
| IMEMode | 0 - fmIMEModeNoControl |
| IntegralHeight | True |
| Left | 6.75 |
| LinkedCell | |
| ListFillRange | |
| ListStyle | 0 - fmListStylePlain |
| Locked | True |
| MatchEntry | 0 - fmMatchEntryFirstLetter |
| MouseIcon | (None) |
| MousePointer | 0 - fmMousePointerDefault |
| MultiSelect | 0 - fmMultiSelectSingle |
| Placement | 2 |
| PrintObject | True |

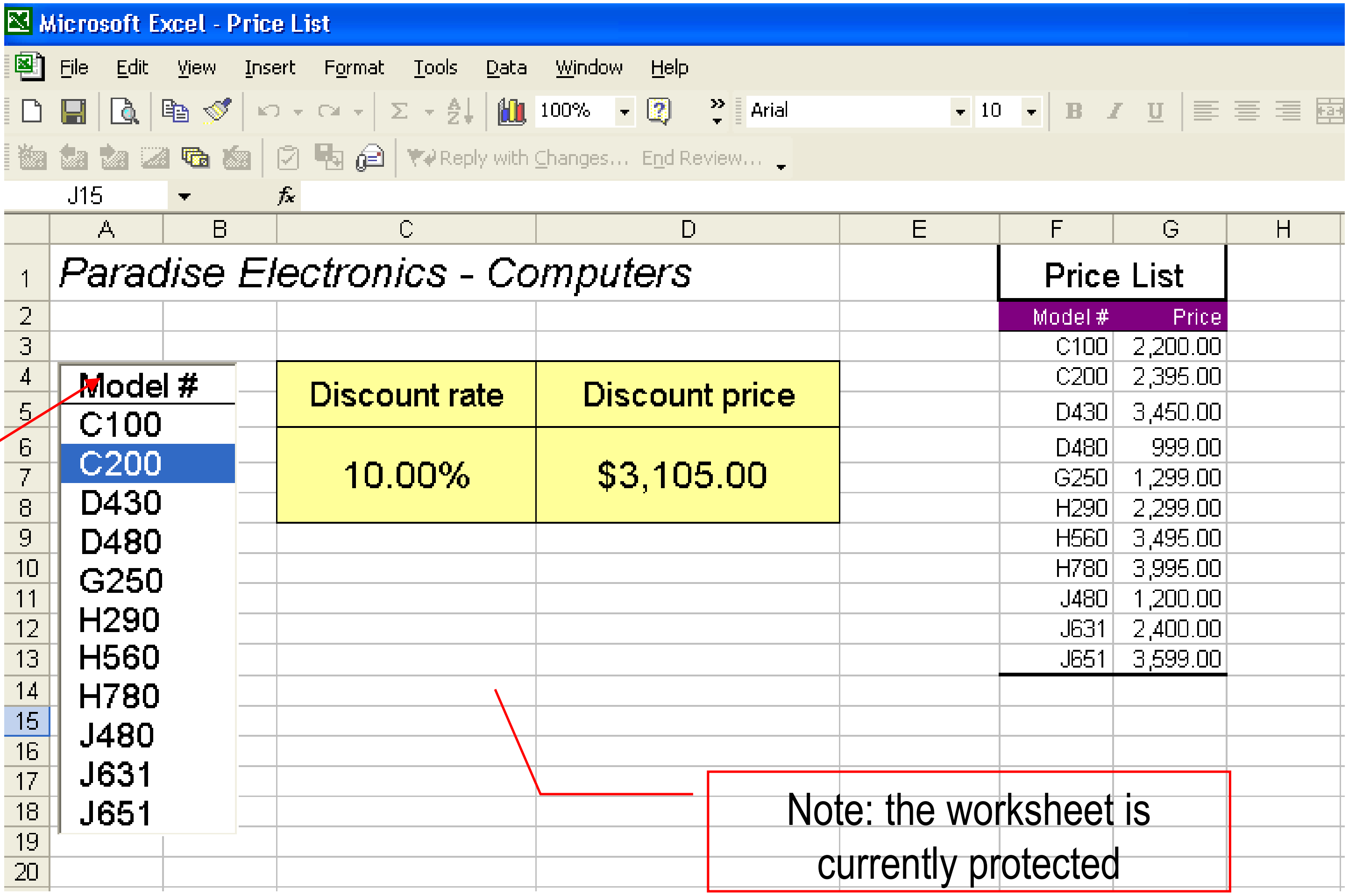
An Object box, located immediately below the Properties window's title bar, displays the **name** and type of the selected object.

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

MONASH University

# List Box after adding heading


Microsoft Excel - Price List
File Edit View Insert Format Tools Data Window Help
100%
Arial
10
Reply with Changes... End Review...
J15

Paradise Electronics - Computers

| Model # |
|---|
| C100 |
| C200 |
| D430 |
| D480 |
| G250 |
| H290 |
| H560 |
| H780 |
| J480 |
| J631 |
| J651 |

| Discount rate | Discount price |
|---|---|
| 10.00% | $3,105.00 |

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 |

Listbox with heading

Note: the worksheet is currently protected

MONASH University

## Coding the List Box's DblClick Event Procedure

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

```
Private Sub lstModel_DblClick(ByVal Cancel As MSForms.ReturnBoolean)
    'declare variables and assign address to object variable
    Dim strRate As String
    Dim sngRate As Single
    Dim curPrice As Currency
    Dim curDiscPrice As Currency
    Dim wksComputers As Worksheet
    Set wksComputers = Application.Workbooks("pricelist.xls").Worksheets("computers")
End Sub
```

For capturing user input

For converting the rate to a number

For referencing cells on the computers worksheet

For storing the price of the selected item

For storing the discount price of the selected item

## Coding the List Box's DblClick Event Procedure

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

```
Private Sub lstModel_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 (whole number):", _
        Title:="Rate", Default:=0)
    'convert rate to decimal
    sngRate = Val(strRate) / 100
End Sub
```

The worksheet Unprotect method. Unprotects the "Computers" worksheet

Obtains discount rate as a string. Assigns it to strRate

Converts string to decimal

## 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

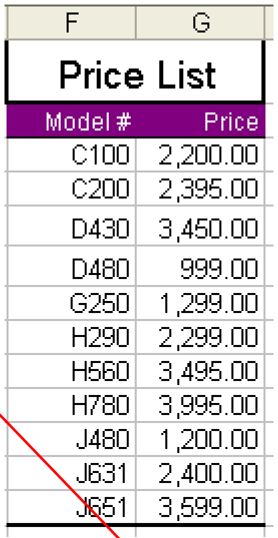
–***table_array*** **:** specifies the location of the lookup table

–***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

e.g. Lookup_value: one of the Model codes G250



| 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. F3:G13

e.g. col_index_num = 2

# Using the Excel Vlookup Function in a Procedure

```
Private Sub lstModel_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 Currenc
    Set wksComputers = Application.Workbooks("price l
'unprotect worksheet
    wksComputers.Unprotect
    'enter discount rate
    strRate = InputBox(prompt:="Enter discount rate (whole number)",
        Title:="Rate", Default:=0)
    'convert rate to decimal
    sngRate = Val(strRate) / 100
      'search for model number and return price
    curPrice = Application.WorksheetFunction.VLookup(lstModel.Text,
        Range("pricelist"), 2, False)
    'calculate the discounted price
    curDiscPrice = (1 - sngRate) * curPrice
    'display discount rate and discounted price
    wksComputers.Range("c6").Value = sngRate
    wksComputers.Range("d6").Value = curDiscPrice
    'protect worksheet
    wksComputers.Protect
End Sub
```

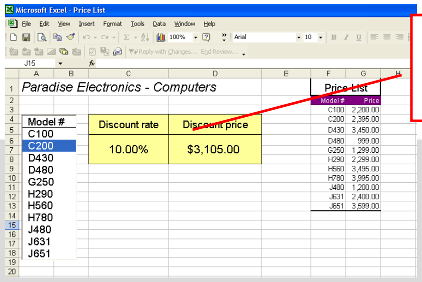
Invoking the Vlookup Worksheet function to find the current price. WorkSheetFunction object enables us to evaluate worksheet functions in VBA code

Table array – which has been named "pricelist"

col_index_num = 2

Range_lookup = false, ensures an exact match in "pricelist"

Calculating the discount price


Microsoft Excel - Price List

J15

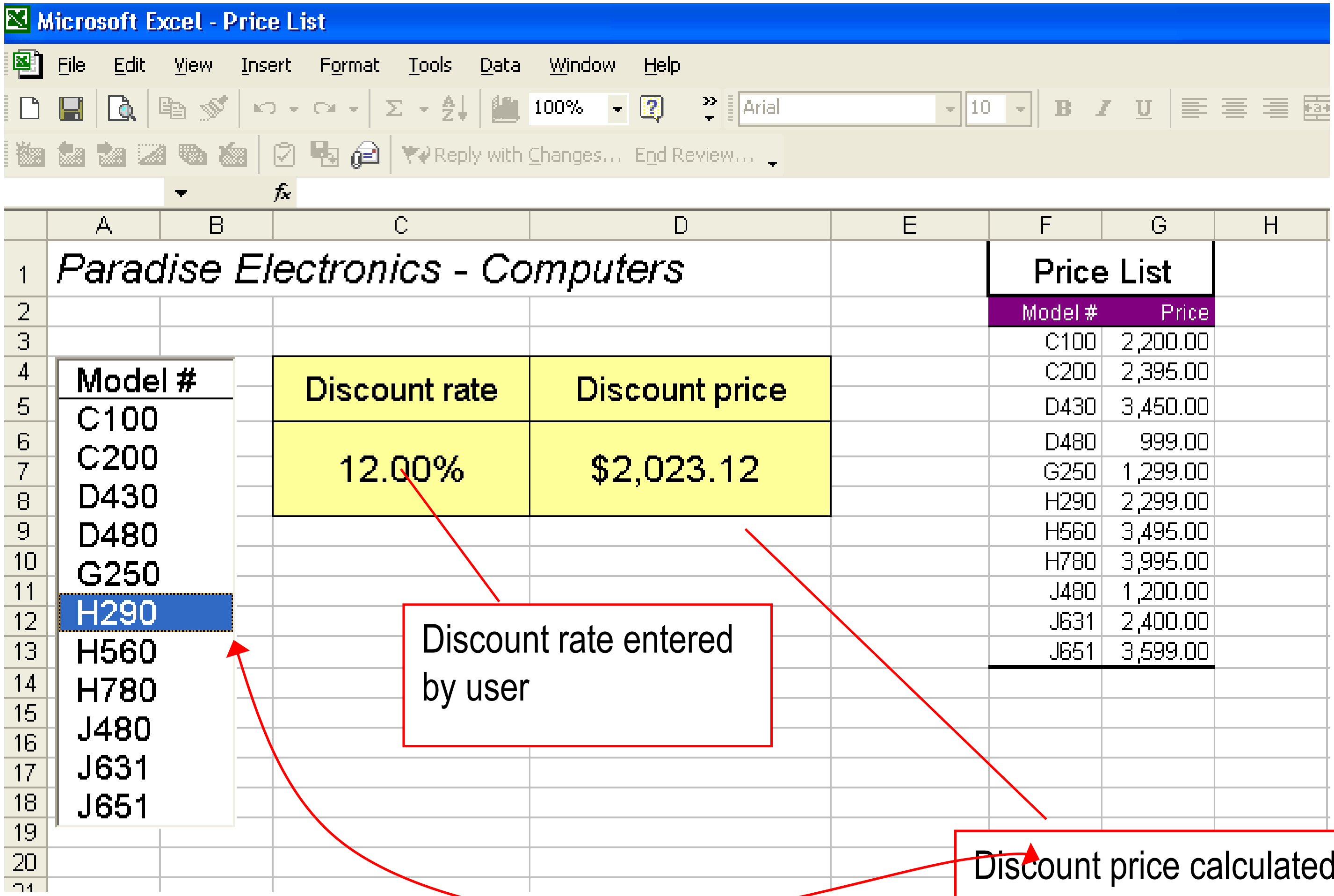
| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | *Paradise Electronics - Computers* | | | | | Price List | |
| 2 | | | | | | Model # | Price |
| 3 | | | | | | C100 | 2,200.00 |
| 4 | Model # | | Discount rate | Discount price | | C200 | 2,395.00 |
| 5 | C100 | | | | | D430 | 3,450.00 |
| 6 | C200 | | 10.00% | $3,105.00 | | 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 | | | | | | |

File Edit View Insert Format Tools Data Window Help
100%
Arial 10
Reply with Changes... End Review...

Displays the results in the "yellow" region

MONASH University

# Worksheet after running the list box's DblClick event


Microsoft Excel - Price List
File Edit View Insert Format Tools Data Window Help
100%
Arial
10
Reply with Changes... End Review...

Paradise Electronics - Computers

| Model # |
|---|
| C100 |
| C200 |
| D430 |
| D480 |
| G250 |
| H290 |
| H560 |
| H780 |
| J480 |
| J631 |
| J651 |

| Discount rate | Discount price |
|---|---|
| 12.00% | $2,023.12 |

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 |

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

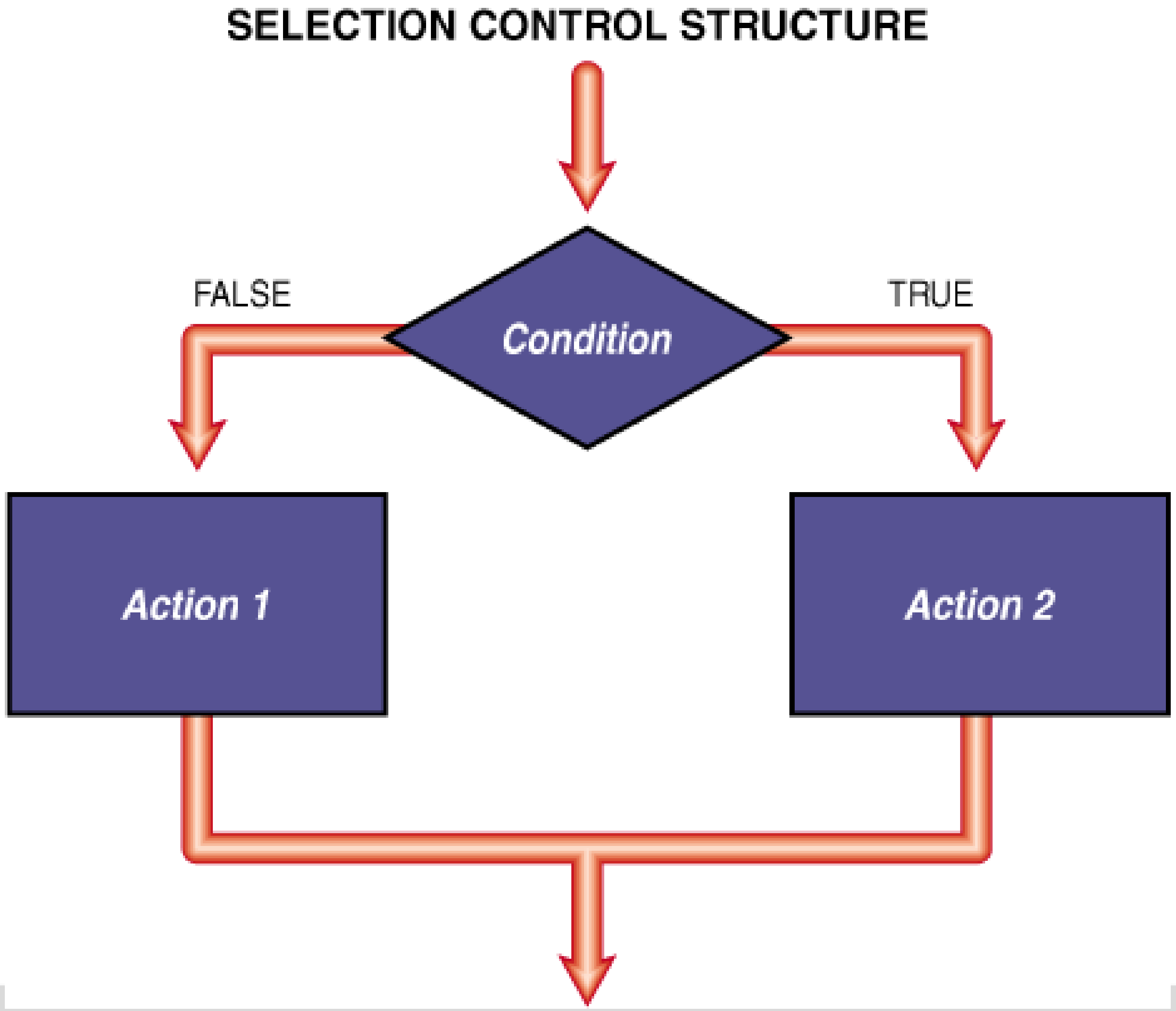
## Program design – VBA control structures

- 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


SELECTION CONTROL STRUCTURE
FALSE
Condition
TRUE
Action 1
Action 2

## 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

General Case:

```
If condition is true then
        perform these tasks
Else
        perform these tasks
End If
```

True path

False path

Case with no Else alternative:

```
If condition is true Then
        perform these tasks
End If
```

## 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

## 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.

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

| **If Then …Else statement** | **Result** |
|---|---|
| If intQuantity < 25 Then<br>    MsgBox Prompt:= “Reorder”<br>End If | Displays “Reorder” if the intQuantity variable contains a value less than 25 |
| If sngHours <= 40 Then<br>    MsgBox Prompt:= “Regular Pay”<br>Else<br>    MsgBox Prompt:= “Overtime Pay”<br>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<br>    curBonus =  curSales * .1<br>Else<br> curBonus =  curSales * .05<br>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 “JOHN ZEBEDEE”

   strName = “JOHN ZEBEDEE”

## 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. “JOHN” is not the same as “John”
- 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:= “John Zebedee”) returns “JOHN ZEBEDEE”

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".

MONASH University

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

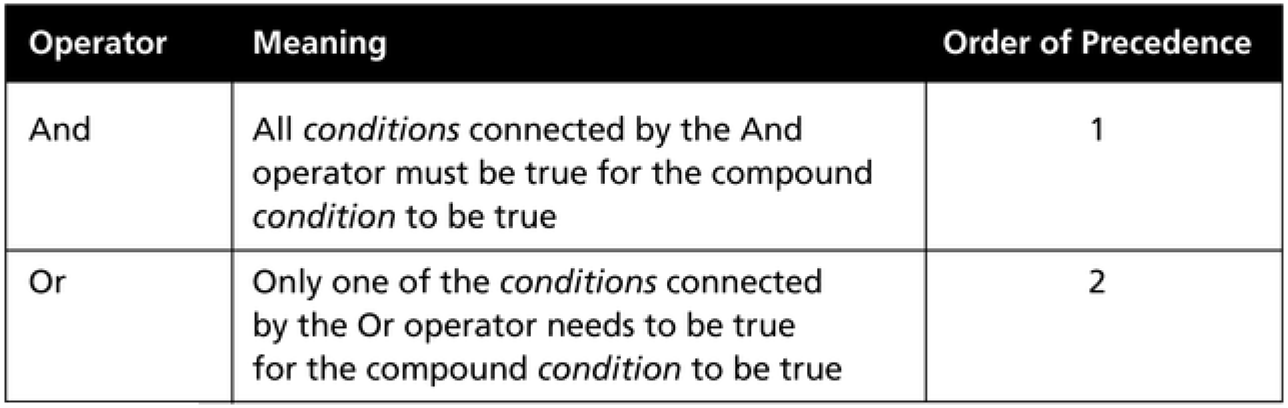
```
If UCase(strAns) = “Y” Then
      MsgBox “answered yes”
End if

If UCase(strAns) = “Y” Then
      intYes = intYes +1
Else
     intNo = intNo + 1
End if
```

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

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

## Logical Operators



| Operator | Meaning | Order of Precedence |
|---|---|---|
| And | All *conditions* connected by the And operator must be true for the compound *condition* to be true | 1 |
| Or | Only one of the *conditions* connected by the Or operator needs to be true for the compound *condition* 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

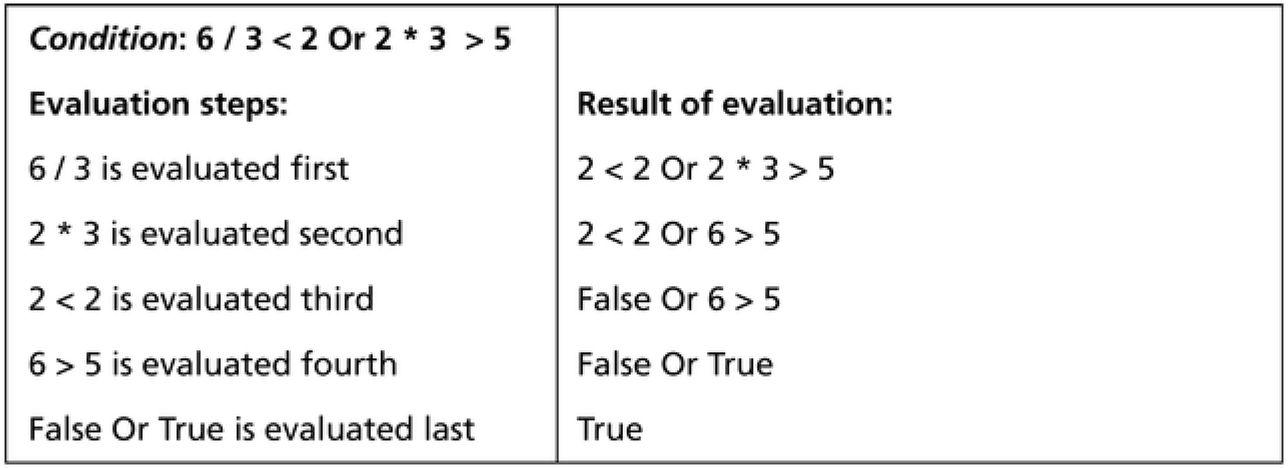
**Not** :Reverses the truth value of *condition*; false becomes true and true becomes false

**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.**

## Logical Operators – order of precedence example



| | |
|---|---|
| *Condition*: 6 / 3 < 2 Or 2 * 3 > 5 | |
| **Evaluation steps:** | **Result of evaluation:** |
| 6 / 3 is evaluated first | 2 < 2 Or 2 * 3 > 5 |
| 2 * 3 is evaluated second | 2 < 2 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

MONASH University

## 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**.

```
sngTest >= 75 And sngProj >= 35
```

## 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.

## 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
```

True path

Nested If

False path

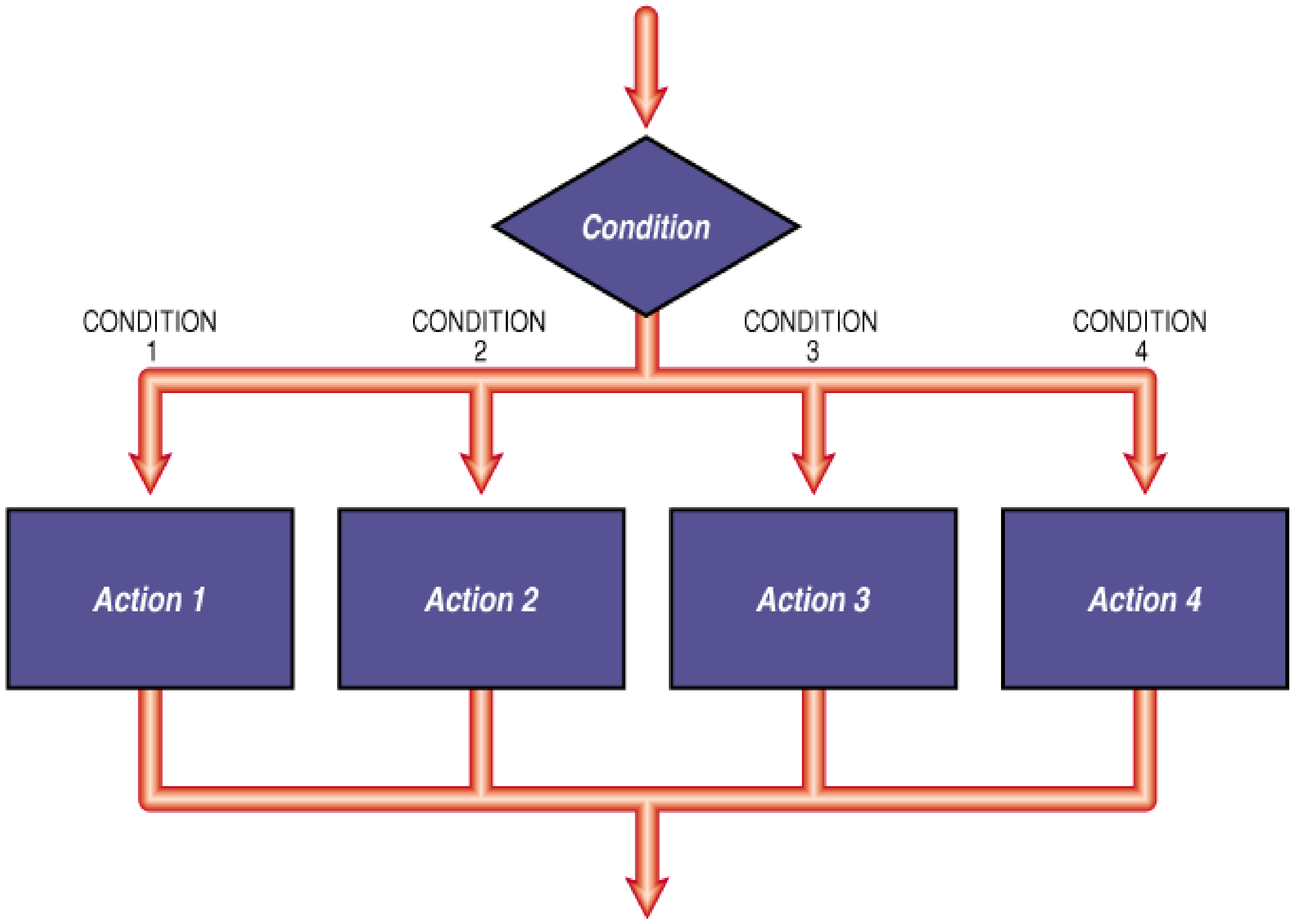
Nested If

## 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

## CASE CONTROL STRUCTURE



# The Select Case Statement

The Select Case clause must include a *testexpression*
The *testexpression* can be any numeric, string, or Boolean expression

The Select Case statement begins with the Select Case clause and ends with the End Select clause

Between the Select Case and End Select clauses are the individual Case clauses

Each Case clause represents a different path that the selection structure can follow

Each case (except Case Else) contains an expression list containing one or more expressions (numeric, string or Boolean) which are possible values of the testexpression

**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**

## 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 in for you

## Example of Select Case

Pseudocode:

```
1. Prompt the user for their test result out of 100
2. If the result is >= 80 then grade is HD
3. If the result is >= 70 then grade is D
4. If the result is >= 60 then grade is C
5. If the result is >= 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