FIT1013 - Week 5 Resources

Fundamentals of Programming

Week 5 Resources

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).

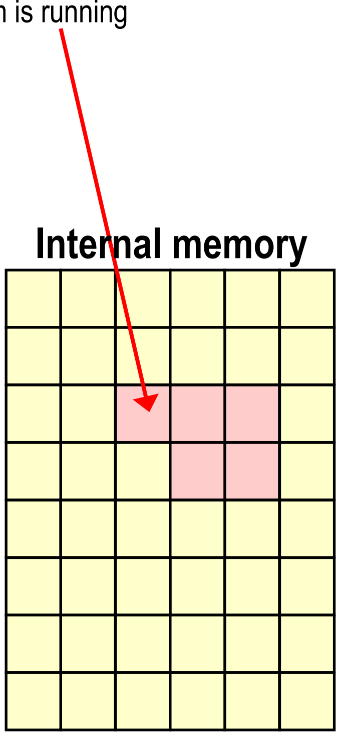
1. Objectives

- Reserve a String variable
- Use an assignment statement to assign a value to a String variable
- Create variables including object variables
- Assign data types and names for object variables
- Use the Set statement
- Use the InputBox function to get information from the user at the keyboard
- Concatenate strings
- Use the MsgBox function
- Use the Val function
- Code a workbook's Open Event procedure

2. Reserve a String variable

Variables

- A programmer can reserve memory cells for storing information
- A variable is a memory location whose value can **change** as the program is running
- It is used to hold temporary information
- It can store only one piece of data at any time
- It can be used to store different types of data: numbers, text, dates



- The variables that you create must have both a name and a data type
- Numeric variables, for example, can store only numbers, while String variables can store numbers, letters, and special characters, such as the dollar sign ($)

Data Types

- Byte
- Boolean
- Currency
- Date
- Double
- Integer
- Long
- Object
- Single
- String
- Variant

Selecting the Appropriate Data Type and Name for a Variable

You must assign a **data type** to each of the variables (memory cells) that you reserve:
- E.g. if a variable is to contain the name of a person then the variable's data type will be **string**.
- E.g. if an object variable is to contain the address of a **Worksheet** object, then the object variable's data type will be **Worksheet**
- In addition to assigning a data type to a variable, you also must assign a name to the variable
- Choose meaningful names so that they help you remember both the data type and purpose of the variable
- Usually, the first three characters should represent the data type
  - Examples: intCost, strName, strAddress

Use the Appropriate Data Type

- **Integer or Long** - Used to store whole numbers
- **Single, Double, Currency** - Used to store numbers with a decimal fraction
- **String** - Used to store strings
- **Object** - Used to store a reference to an object
- **Byte** - used to store small numbers
- **Variant** - Stores any data type, flexible, but not efficient

Types of Variables

Two types of variables:
- **Value variables** - can store values such as strings, numbers and dates
- **Reference variables** - store memory addresses
- http://www.excel-spreadsheet.com/vba/objectvariables.htm
- https://msdn.microsoft.com/en-us/library/99053c13.aspx

Some Value Variables

- A **numeric** variable is a memory cell that can store a number—for example, it can store an employee's gross pay amount
- A **Date** variable is a memory cell that can store date and time information date, such as a birth date
- A **Boolean** variable is a memory cell that can store the Boolean values True and False
- A **String** variable is a memory cell that can store a string, which is zero or more characters enclosed in quotation marks ("")

Reference Variables

- Object variables are reference variables. Object variables store the **address** of the object in memory rather than the object itself. i.e. an object variable "points to" an object.

E.g.

- A Worksheet object variable contains the address of a particular worksheet in memory

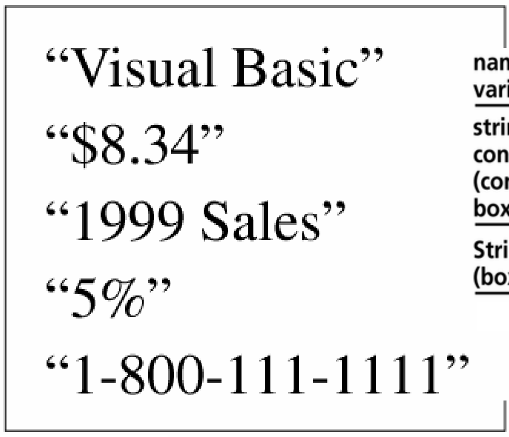
Reserving a Procedure-level String Variable

The **Dim** statement is used to reserve a procedure-level variable, and the variable can only be used by that procedure:

Dim VariableName as String

E.g.Dim strName as String

- When the procedure ends, VBA removes the procedure-level variable from memory
- When creating a String variable the **datatype** used is always the keyword **String**
- When you use the Dim statement to reserve a String variable in memory, VBA automatically initializes the variable to a zero-length string
- A zero-length string, often referred to as an empty string, is simply two quotation marks with nothing between them, like this: ""
- The more technical term for a string is **string literal constant**



"Visual Basic"

"$8.34"

"1999 Sales"

"5%"

"1-800-111-1111"

Examples of strings

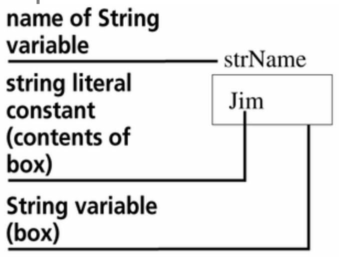
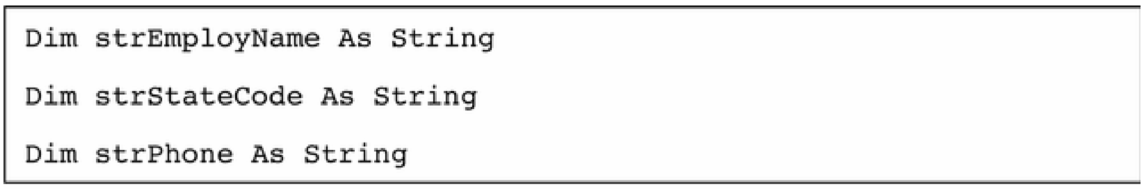


Illustration of a String variable and a string literal constant

Note: the string variable is the address. The string literal constant, or ‘string’ for short, is what is stored there.

- Literal refers to the fact that the characters enclosed within the quotation marks should be taken literally
- Constant refers to the fact that the string’s value does not change while a procedure is running
- Be careful not to confuse a String variable with a string literal constant
- Remember, when you use the Dim statement to reserve a String variable in memory, VBA automatically initializes the variable to a zero-length string
- You should assign a descriptive name to each variable that you reserve
- The name should reflect both the variable’s data type and purpose
- One popular naming convention is to have the first three characters in the name represent the data type, and the remainder of the name represent the variable’s purpose
- Variable names cannot be longer than 255 characters and they cannot be reserved words, such as Print or MsgBox

Examples:



```
Dim strEmployName As String

Dim strStateCode As String

Dim strPhone As String
```

Examples of Dim statements that reserve String variables

3. Using an Assignment Statement to Assign a Value to a String Variable

- Assignment statements are so named because they assign values to the memory cells inside the computer
- When you assign a new value to a memory cell, the new value replaces the old value, because a memory cell can store only one value at a time

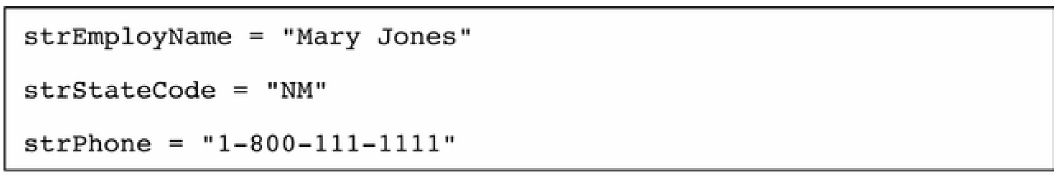
Assignment statement

variablename = *value*

Example:

strName = “Andrew”

Assigning a string literal constant to a string variable



```
strEmployName = "Mary Jones"

strStateCode = "NM"

strPhone = "1-800-111-1111"
```

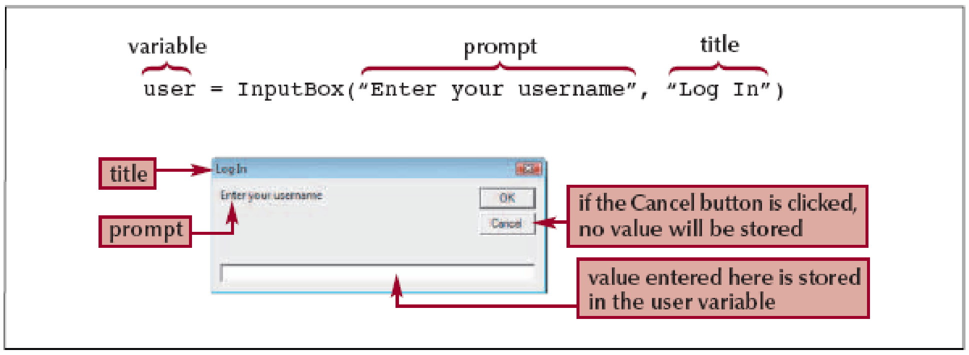
Examples of assignment statements that assign string literal constants to String variables

4. InputBox function

Reminder: a function is a set of instructions that performs a task and returns a value after the task is done

- The **InputBox function** displays one of VBA's predefined dialog boxes
- The dialog box contains a title, a message, an OK button, a Cancel button, and an input area in which the user can enter information
- The InputBox function **returns a string** which can be captured in a variable of type string
- The **syntax** of the InputBox function is:
  InputBox(Prompt:=*prompt* [, Title:=*title*] [, Default:=*defaultValue*])
- The message that you display in the dialog box should prompt the user to enter the appropriate information in the input area of the dialog box

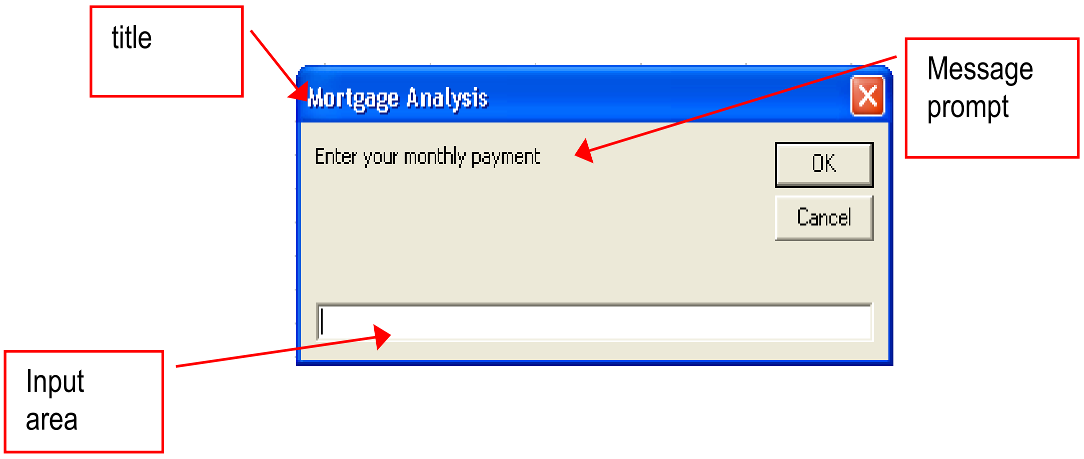
InputBox function **Figure 12-29**


$$\underbrace{\texttt{user}}_{\text{variable}} \texttt{ = InputBox(}\underbrace{\texttt{"Enter your username"}}_{\text{prompt}}\texttt{, }\underbrace{\texttt{"Log In"}}_{\text{title}}\texttt{)}$$

title
prompt
Log In
Enter your username
OK
Cancel
if the Cancel button is clicked, no value will be stored
value entered here is stored in the user variable

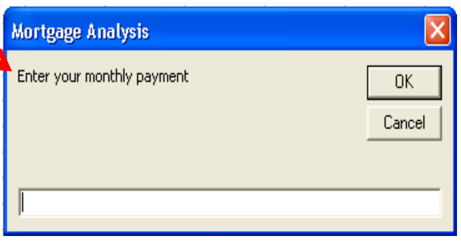
Using InputBox function

E.g. InputBox(Prompt:="Enter your monthly payment", Title:="Mortgage Analysis")



Using the InputBox Function

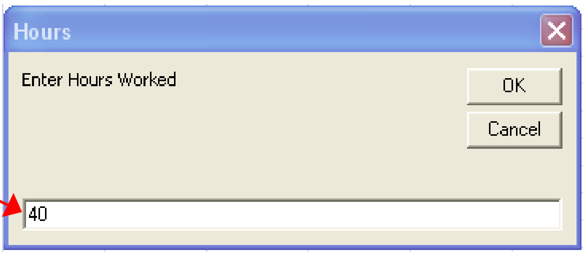
- The standard in Windows is to use sentence capitalisation for the prompt, and book title capitalisation for the title
- Sentence capitalisation means that you capitalise only the first word and any words that are customarily capitalized
- Book title capitalisation means that you capitalise the first letter in each word, except for articles, conjunctions, and prepositions that do not occur at either the beginning or end of the title



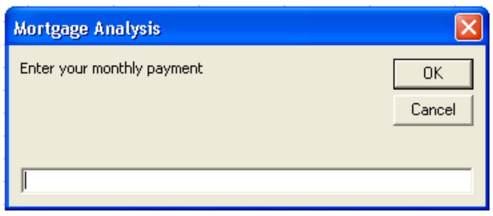
Using the InputBox Function examples

- strName = InputBox(Prompt:="What is your name? ", Title:="Name Information")
- strName = InputBox(Prompt:="Enter Hours Worked", Title:="Hours", Default:="40")

The Default parameter can be used to place a default value in the input area



Using the InputBox Function



Enter your monthly payment

The InputBox function returns a string
E.g. the following code stores the monthly payment in a string variable called strPayment:

```
        Dim strPayment as string

        strPayment = InputBox(Prompt:="Enter your monthly payment", Title:="Mortgage
Analysis")
```

Alternatively,

```
        strPayment = InputBox("Enter your monthly payment", "Mortgage Analysis")
```

5. Concatenating Strings

- Connecting (or linking) strings together is called concatenating
- In VBA, you concatenate strings with the concatenation operator—the ampersand **(&)**
- When concatenating strings, you must be sure to include a space before and after the concatenation operator
- If you do not enter a space before and after the ampersand, VBA will not recognize the ampersand as the concatenation operator
- In addition to concatenating strings, you also can use strings in calculations

String Concatenation

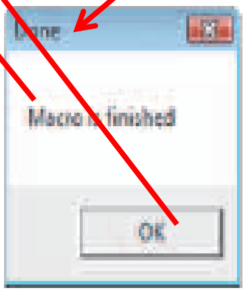
- Ampersand - &
- Example: (Assume strFirstName contains "Mary")
  - "Hello " & strFirstName
  - Result:
  - Hello Mary

Note spaces

- Ampersand - &
- Example: (Assume strFirstName contains "Mary" and sngSales contains 1000)
  - strFirstName & " sold $" & sngSales & "."
- Result:
  - Mary sold $1000.

6. Message Boxes

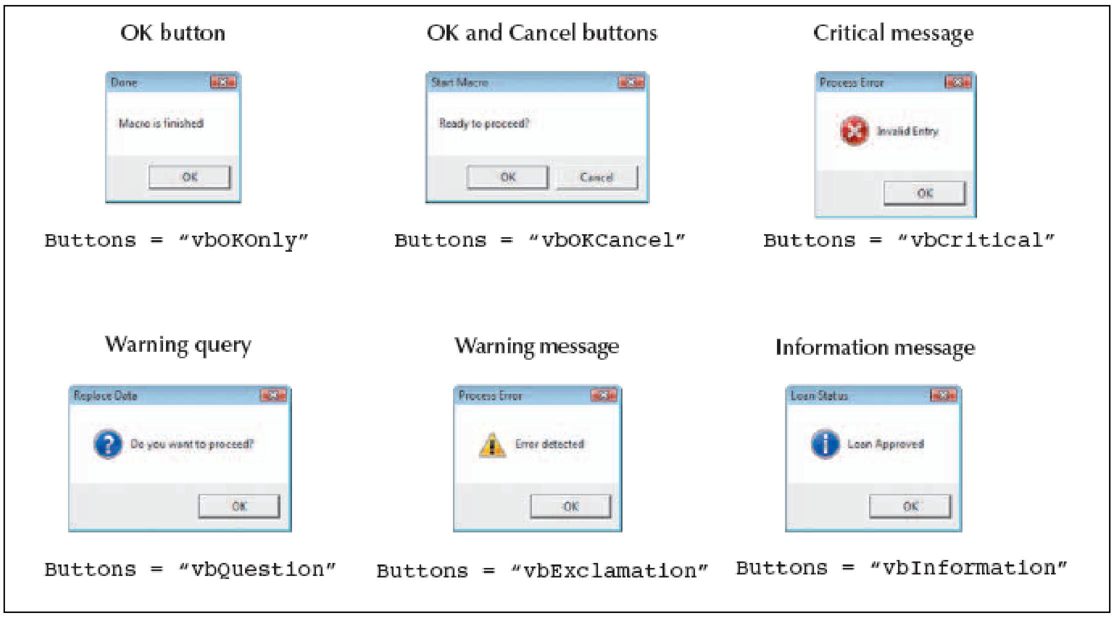
- The Message Box function can be used to display a message to the user.
- The syntax of the MsgBox function:
  - MsgBox Prompt, Buttons, Title
  - Where prompt is the message in the dialog box, title is the text in the title bar and Buttons is the type of button that appears on the message box.



Creating a Message Box

- MsgBox Prompt, Buttons, Title

Button parameters **Figure 12-39**

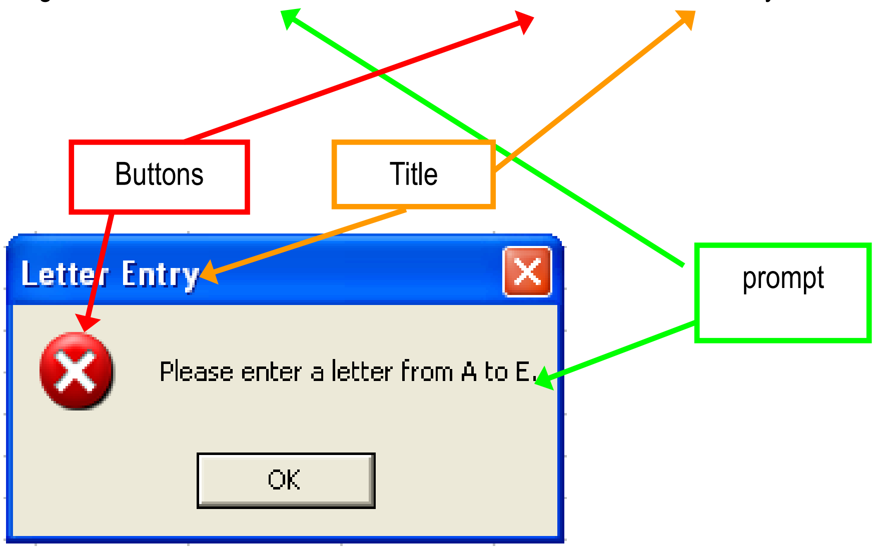


```
Buttons = "vbOKOnly"      Buttons = "vbOKCancel"      Buttons = "vbCritical"

Buttons = "vbQuestion"    Buttons = "vbExclamation"   Buttons = "vbInformation"
```

Letter Entry message InputAndMsgBoxEgs.xls

MsgBox "Please enter a letter from A to E", vbCritical, “Letter Entry“



Please enter a letter from A to E.

Can also be explicitly written as:
MsgBox Prompt:="Please enter a letter from A to E.", Buttons:=vbCritical, Title:="Letter Entry"

MsgBox Example

```
Public Sub StringMsgBoxEg()
Dim strFirst As String
Dim strFamily As String
strFirst = "Joe"
strFamily = "Blogs"
MsgBox "Hello " & strFirst & " " & strFamily, vbOKOnly, "Welcome Message"
End Sub
```

Declare 2 string variables

Assign values to the string variables

prompt

Buttons

Title

Concatenate 4 strings and present the result to the user – using the MsgBox function

MsgBox Function

NB – the MsgBox function **returns an integer**, the value of which depends on which button the user clicks.
The MsgBox function can be used to capture information about which button the user clicked.

This will be covered in week 6, with integer variables.

The Val Function – for converting strings to numbers

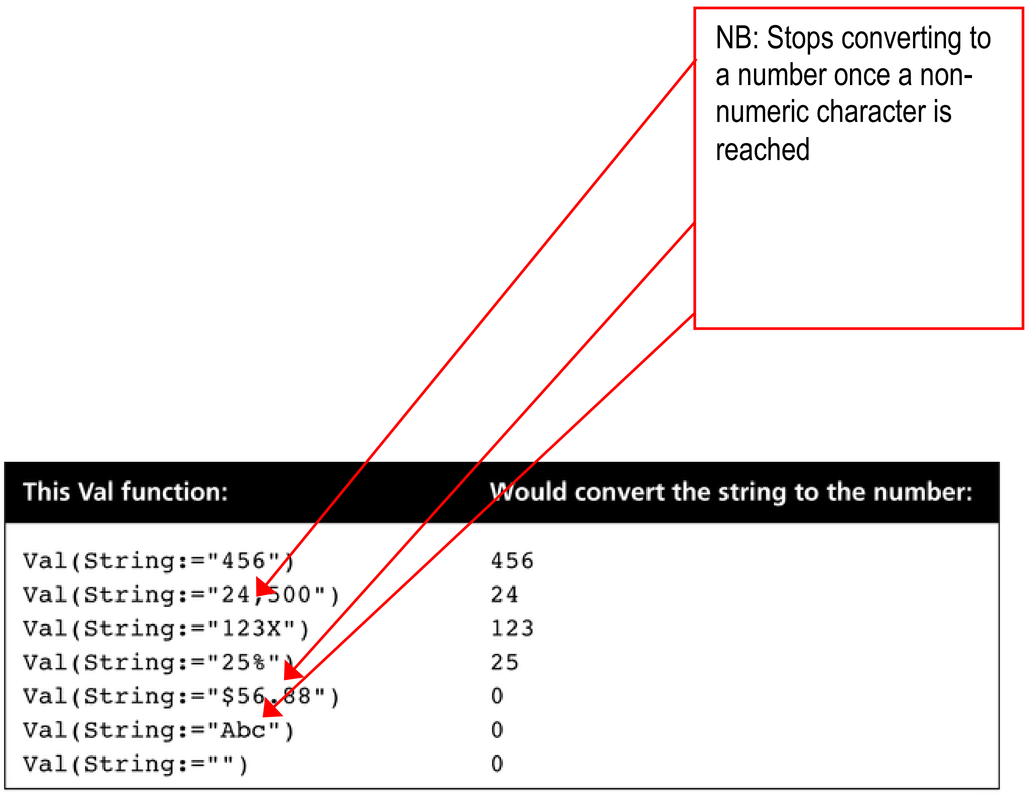
Remember – the InputBox function returns a string (not a number)

- If the user enters a number, this is stored as a string which can be converted to a number using:
    - The **Val function**
    - For using strings in calculations

Using the Val Function

- In addition to concatenating strings, you also can use strings in calculations
- Remember that the InputBox function returns a string
- The Val function converts a string into a number, and it then returns the number
- The syntax of the Val function is **Val(String:=string)**
    - E.g. **Val(String:="45")**, also **Val("45")**
- The string can contain only numbers and the full stop symbol "."
- Val("45.67") converts to the number 45.67
- In addition to string literal constants, the string in a Val function also can be a String variable
    - E.g. **Val(String:=strHours)** where strHours is a string variable.
- When you use the Val function to convert the contents of a String variable to a number, VBA first creates a temporary copy of the String variable in memory; it then converts the contents of the copy to a number

Examples of Using the Val Function to Convert String Literal Constants to Numbers



NB: Stops converting to a number once a non-numeric character is reached

This Val function:	Would convert the string to the number:
`Val(String:="456")`	456
`Val(String:="24,500")`	24
`Val(String:="123X")`	123
`Val(String:="25%")`	25
`Val(String:="$56.88")`	0
`Val(String:="Abc")`	0
`Val(String:="")`	0

Examples of using the Val function to convert string literal constants to numbers

Using strings in calculations

Using the Val function, we can now convert strings to numeric values and use them in calculations:

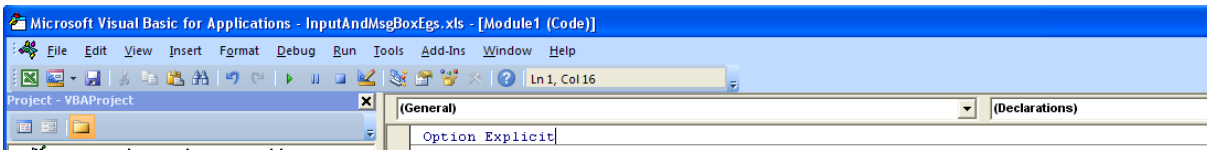
E.g.

If strAge = “23”

- **wksEmployee.Range(“B1”).Value = Val(strAge) + 1**
- Places the value 24 into cell B1 of the worksheet that wksEmployee points to

7. The Option Explicit Statement

- You can use the **Option Explicit** statement to prevent VBA from reserving variables that you did not explicitly declare
  - It is considered poor programming practice to allow VBA to reserve variables "on the fly"—in other words, to reserve variables that you did not declare in a Dim statement—because it makes finding errors in your program more difficult
- The **Option Explicit** statement tells the Visual Basic Editor to display an error message if your code contains the name of an undeclared variable
- However you must be sure to enter the statement in each of the project's modules, i.e. enter in every form's and every code module's General declarations section.


Microsoft Visual Basic for Applications - InputAndMsgBoxEgs.xls - [Module1 (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 1, Col 16
Project - VBAProject
(General)
(Declarations)

```
Option Explicit
```

e.g. code containing a misspelled variable name

Dim strName as string

strName = "Yolanda"

MsgBox Prompt:=strName

strNames = "Patty"

MsgBox Prompt:=strName

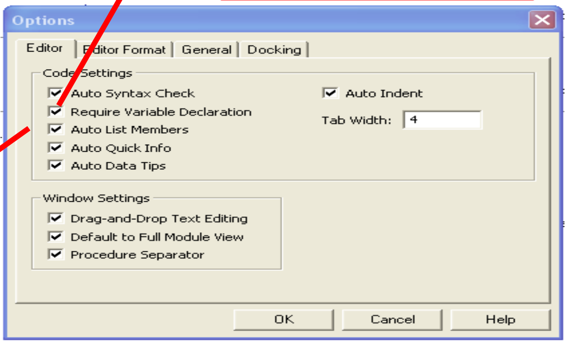
Displays the value of strName on the screen

If the Option Explicit statement is not used, there will be no error messages when an undeclared variable is mistakenly used

Misspelled variable name

- In the Visual Basic editor, use Tools, Options, Editor tab, Require Variable Declaration to have Visual Basic include Option Explicit in every new form and module.

Automatically enters the Option Explicit



Code Settings
- [✓] Auto Syntax Check
- [✓] Require Variable Declaration
- [✓] Auto List Members
- [✓] Auto Quick Info
- [✓] Auto Data Tips
- [✓] Auto Indent

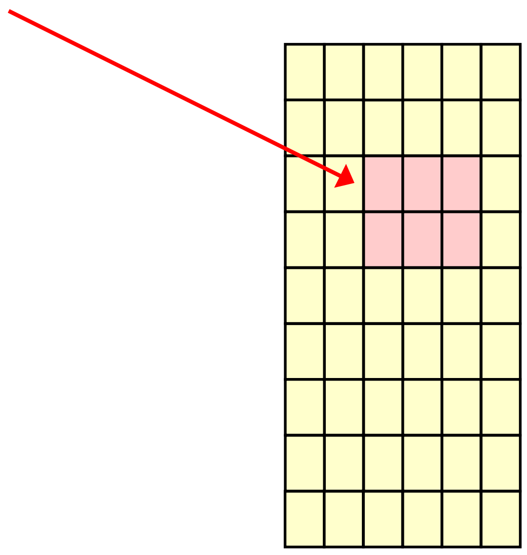
Tab Width: 4

Window Settings
- [✓] Drag-and-Drop Text Editing
- [✓] Default to Full Module View
- [✓] Procedure Separator

Options dialog box – showing Require Variable Declarations check box

8. Memory Cells – Objects

- Every **object** in a VBA-enabled application has a **set of properties** whose values control the object's appearance and behavior
- VBA stores each **property**, along with its corresponding **value**, inside the computer in an area called **internal memory**
- When VBA creates an object, it reserves a group of boxes (memory cells) in which it stores information about the object



E.g. Illustration of Memory Cells that Store Property Values for the Worksheet object

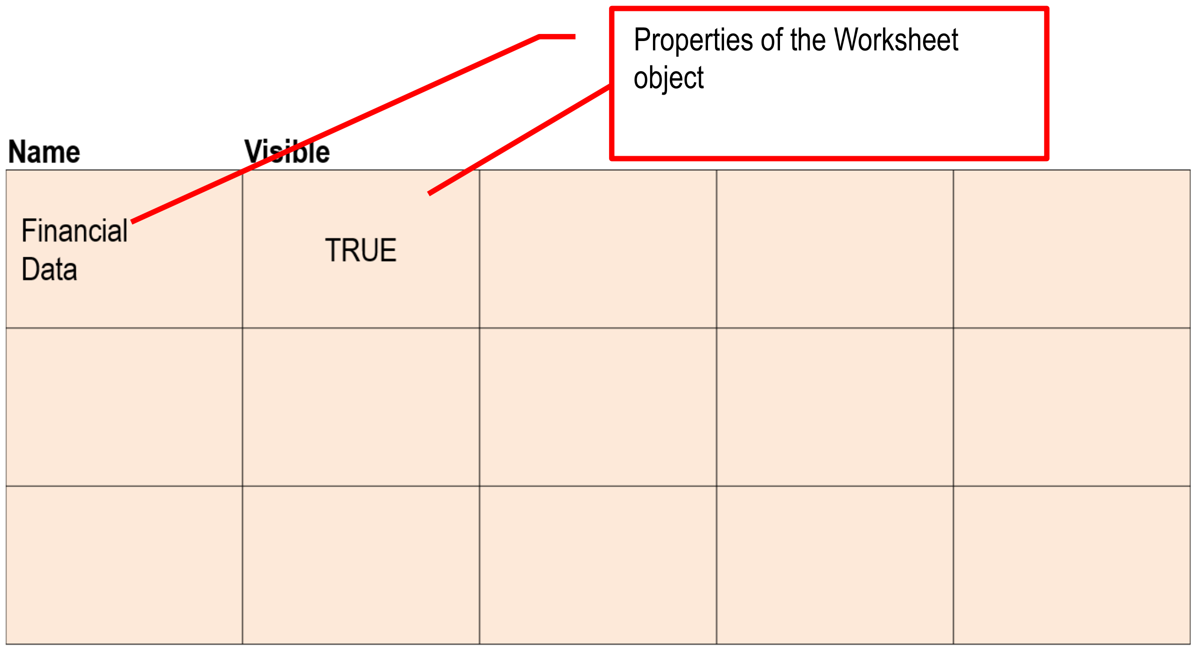
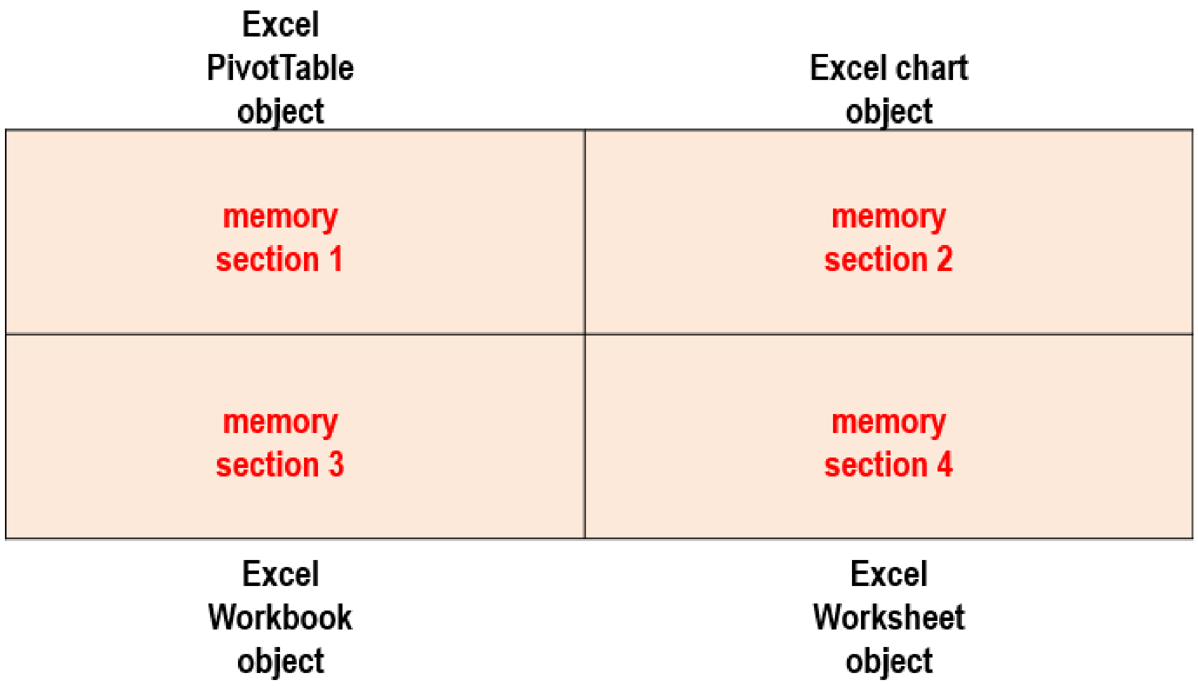


Illustration of memory cells that store property values

How Each Object Occupies a Separate Section in Memory



Memory Cells

In addition to assigning both a name and a value to each property's memory cell, VBA also assigns a **data type**
The **data type** refers to the type of data the memory cell can store

E.g. numeric, text……

e.g. for Worksheet object:
The **Name** property contains text values, so the data type is string
The **Visible** property contains a binary value (TRUE or FALSE), so the data type is binary

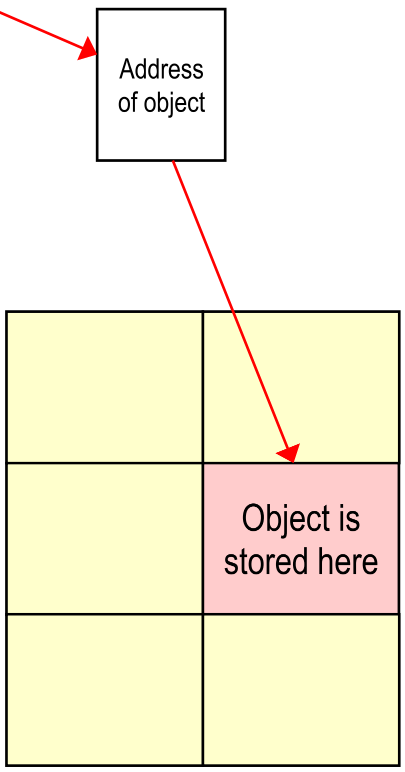
You can determine the type of data that a property's memory cell can store by viewing the property's Help screen

Name	Visible			
Financial Data	TRUE			

Illustration of memory cells that store property values

Object Variables

- An object variable is a memory cell (box) that contains the **address** of an object in memory
- The **address** tells VBA where the object is located in memory
- Object variables help to improve the performance of a procedure by allowing the procedure to run more quickly
- Programmers use object variables to make procedures easier to write and understand
- Using object variables to hold portions of VBA code shortens the instructions and this makes the code easier to execute



Example:

When VBA processes the following instructions:

MsgBox Prompt:=Application.Workbooks(1).Worksheets(1).Name

First it must locate the appropriate Application object in memory, then it must locate the first Workbook object within the Workbook **collection**, followed by the first Worksheet object within the Worksheet **collection**

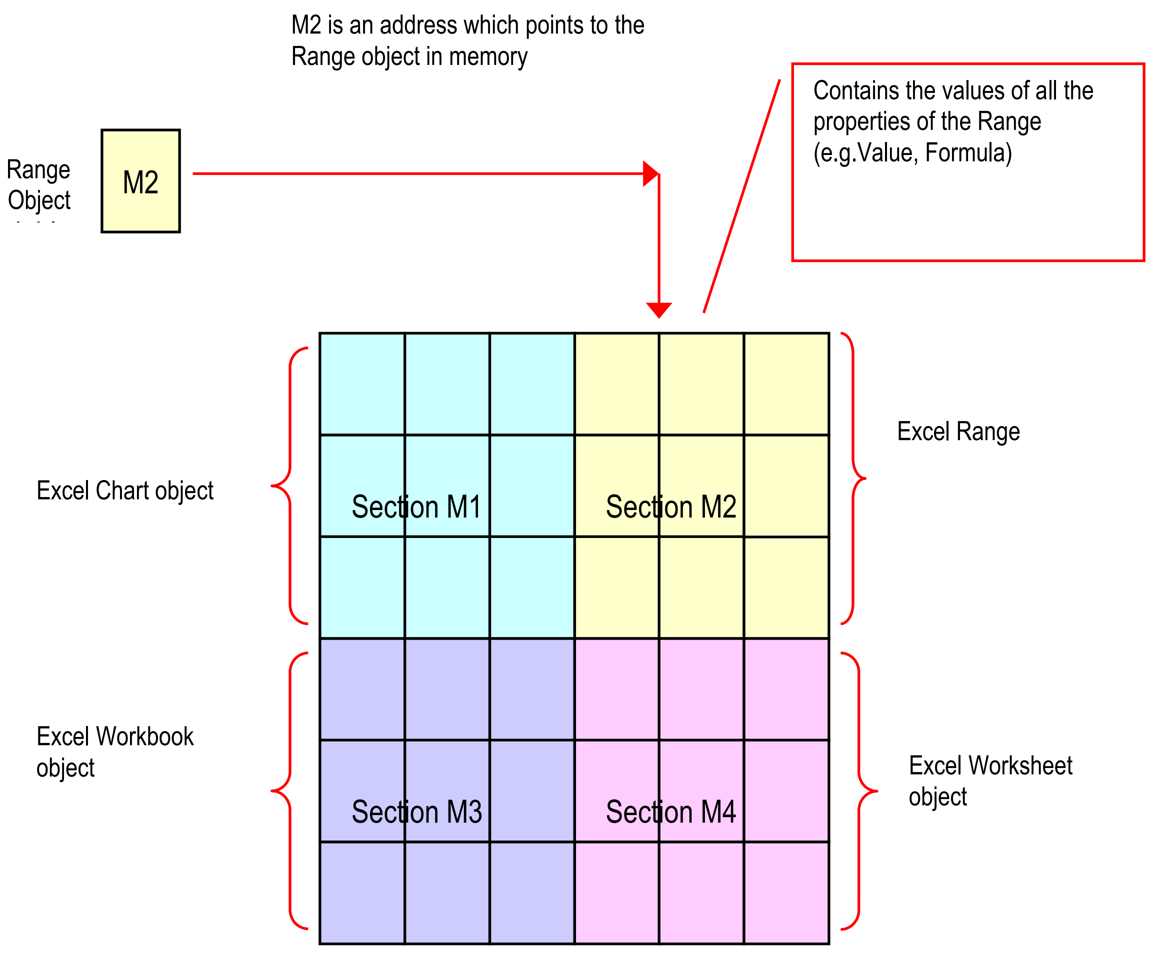
Alternative: if we assign the address of the first Worksheet object to a variable called **wksFirst**, then subsequent referencing of the first sheet's properties is done more efficiently by referencing the object **wksFirst**

Points directly to the location of the sheet in memory

```
MsgBox Prompt: = wksFirst.Name

MsgBox Prompt: = wksFirst. Range("b1").Value
```

An Object Variable and the Object to Which It Refers e.g. Range object


M2 is an address which points to the Range object in memory
Range Object
M2
Contains the values of all the properties of the Range (e.g.Value, Formula)
Excel Chart object
Section M1
Section M2
Excel Range
Excel Workbook object
Section M3
Section M4
Excel Worksheet object

9. Selecting the Appropriate Data Type and Name for an Object Variable

- You must assign a **data type** to each of the variables (memory cells) that you reserve:
- E.g. if an object variable is to contain the address of a **Worksheet** object, then the object variable's **data type** will be **Worksheet**
- E.g. if an object variable is to point to a **Range** object, then the object variable's **data type** will be **Range**
- In addition to assigning a data type to a variable, you also must assign a name to the variable
- Choose meaningful names so that they help you remember both the data type and purpose of the variable
- Usually, the first three characters should represent the data type
  - Examples: **rngSales, wksFinancial, wkbPay**
- It is a common practice to type the three-character ID in lowercase and capitalize the first letter in the part of the name that identifies the purpose
  - E.g. **wkbFinancial, wksInventory, rngCustomers**
- In addition to being descriptive, the name that a programmer assigns to a variable must follow several specific **rules**:
  - The name must begin with a letter
  - The name must contain only numbers, letters or the underscore (i.e. no punctuation or spaces are allowed in the name)
  - The name cannot be more than 255 characters long
  - The name cannot be a reserved word such as Print

Three-character IDs according to various data types

Byte	byt
Boolean	bln
Currency	cur
Date/Time	dtm
Double	dbl
Integer	int
Long	lng
Single	sng
String	str
Variant	vnt

Three-character IDs for object data types

Examples:

rng	range object
wkb	workbook object
wks	worksheet object

Valid/invalid object variable names

Valid object variable names	Invalid object variable names
rng97Sales	**97Salesrng** (the name can't start with a number)
rngRegionWest	**MsgBox** (the name can't be a reserved word)
rngEast	**rng.East** (the name can't contain punctuation marks)
wks25N	**rngRegion West** (the name can't contain spaces)

Creating (declaring) a Variable

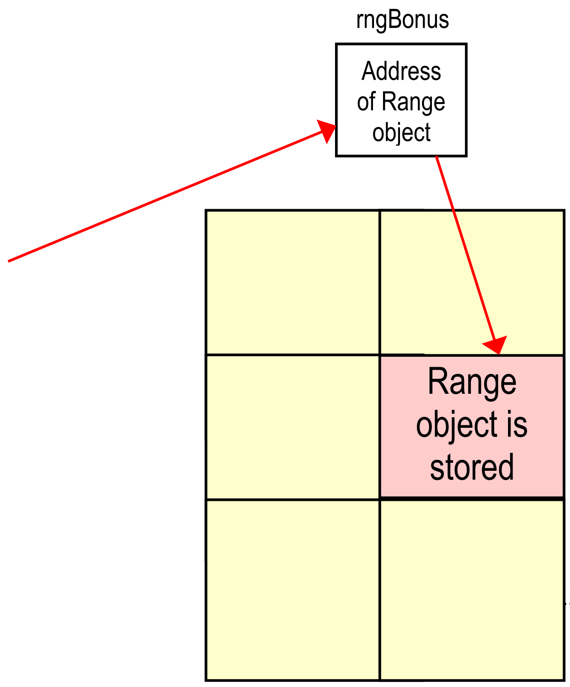
General syntax:
Dim *variablename* [As *datatype*]
E.g:

Dim strSales as string

Declaring an object variable

Use the **Dim** statement to declare an object variable, VBA reserves a memory cell to which it attaches variablename as the name and datatype as the data type
E.g.

- **Dim rngBonus As Range**
  - (creates an object variable named rngBonus that can store the **address** of a Range object)
- **Dim wkbPay As Workbook**
  - (creates an object variable named wkbPay that can store the address of a Workbook object)



- **Dim wksPay as Worksheet**
  - (creates an object variable named wksPay that can store the address of a Worksheet object)

- VBA also automatically stores the keyword *Nothing* in the object variable, which is referred to as **initialising** the variable.

What happens if you do not assign a data type to a variable?

- VBA assigns a default data type, the **variant** data type which may reduce the efficient use of memory.

10. Using the Set Statement

- You use the **Set** statement to assign the address of an object to an object variable (removes the keyword **nothing** and replaces it with the address of an object of the type specified in the **Dim** statement) The syntax of the Set statement is:

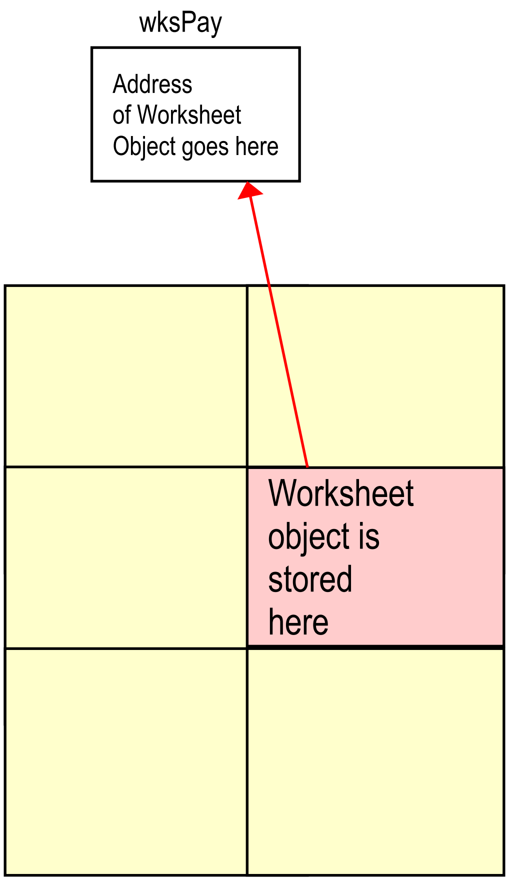
  Set objectVariableName=object

  where **objectVariableName** is the name of an object variable, and **object** is the object whose address you want to store in the variable

- The **Set** statement locates the object in memory and then stores the object's address in the memory cell whose name is **objectVariableName**

  E.g.

- Set wksPay = Application.Workbooks(1).WorkSheets(1)
  - In this example, the first sheet of the first workbook of the application is located and its address is then stored in the wksPay object variable
- If you request a property for the variable, it will retrieve the property by going to the memory location specified by the object variable


wksPay
Address of Worksheet Object goes here
Worksheet object is stored here

Examples of the Set Statement

- Set wksPay = Application.Workbooks(1).Worksheets(2)
- Set rngWest = Application.Workbooks(1).Range("database")
- Set rngCustomers = Application.Workbooks(1).Worksheets(1).Range("B3:B22")

Summary

To create a procedure-level object variable, and then assign an address to it:

1. Use the **Dim** statement to create the variable
2. Use the **Set** statement to assign the address of an object to an object variable

E.g.

```
Dim wksPay as Worksheet
Set wksPay = Application.Workbooks(1).WorkSheets(1)
```

1. Creates a worksheet object variable called wksPay whose address is initialised to Nothing.
2. Takes an actual object and stores the address of that object in the variable wksPay

11. Coding the Workbook's Open Event Procedure

Excel VBA example:
Uses Excel to calculate the sales commission based on a commission rate that is entered by the user

The procedure to calculate the commission is invoked when the workbook is opened

Uses the commission.xls workbook.

commission.xls

Event procedures

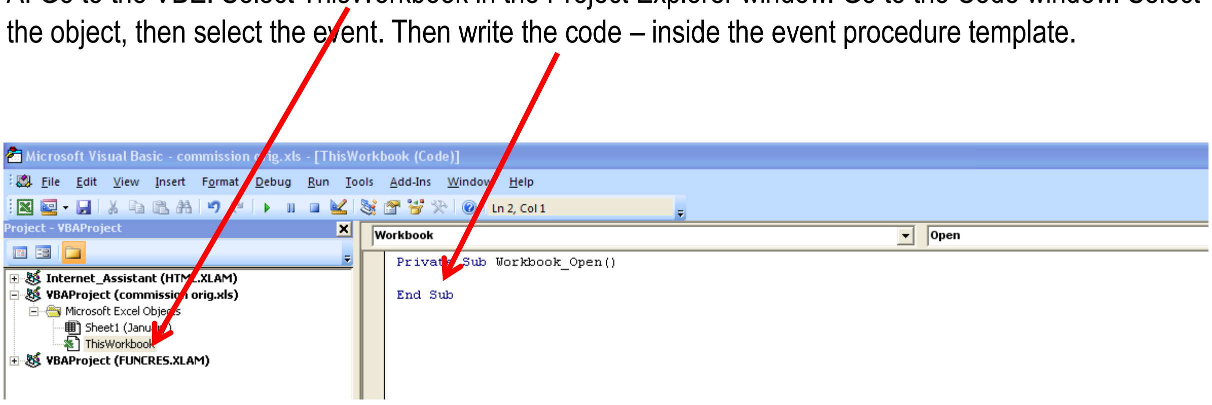
Q: What is an event procedure?
A: A procedure (program) that runs in response to an event

Q: What kind of event?
A: Events that happen within Excel – e.g.

- Opening an Excel file
- Moving to a different worksheet
- Clicking a command button on a worksheet
- Closing an Excel file
- Etc.

The Workbook Open event procedure

Q: Where do we put the code for the Commission Workbook's open event procedure?
A: Go to the VBE. Select ThisWorkbook in the Project Explorer window. Go to the Code window. Select the object, then select the event. Then write the code – inside the event procedure template.


Microsoft Visual Basic - commission orig.xls - [ThisWorkbook (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 2, Col 1
Project - VBAProject
Internet_Assistant (HTML.XLAM)
VBAProject (commission orig.xls)
Microsoft Excel Objects
Sheet1 (January)
ThisWorkbook
VBAProject (FUNCRES.XLAM)
Workbook
Open

```
Private Sub Workbook_Open()

End Sub
```

Coding the Workbook's Open Event Procedure

Declare and set the string and object variables as shown below:

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

Dim strName As String

Dim strRate As String

Dim wksJan As Worksheet

Set wksJan = _
Application.Workbooks("commission.xls").Worksheets(
"January")
```

Declare 2 string variables

Declare 1 worksheet object

Store the address of the "January" worksheet in the worksheet object variable

Coding the Workbook's Open Event Procedure

Prompt for name and rate using the InputBox() function

```
'enter name

strName = InputBox(prompt:="Enter your name",
Title:="Name")

'enter rate

strRate = _

InputBox(prompt:="Enter rate as a whole number",
Title:="Rate", Default:=wksJan.Range("b1").Value * 100)

'convert rate to decimal

wksJan.Range("b1").Value = Val(String:=strRate) / 100

'display message and name

wksJan.Range("a19").Value = "Prepared By " & strName
```

Points to January worksheet

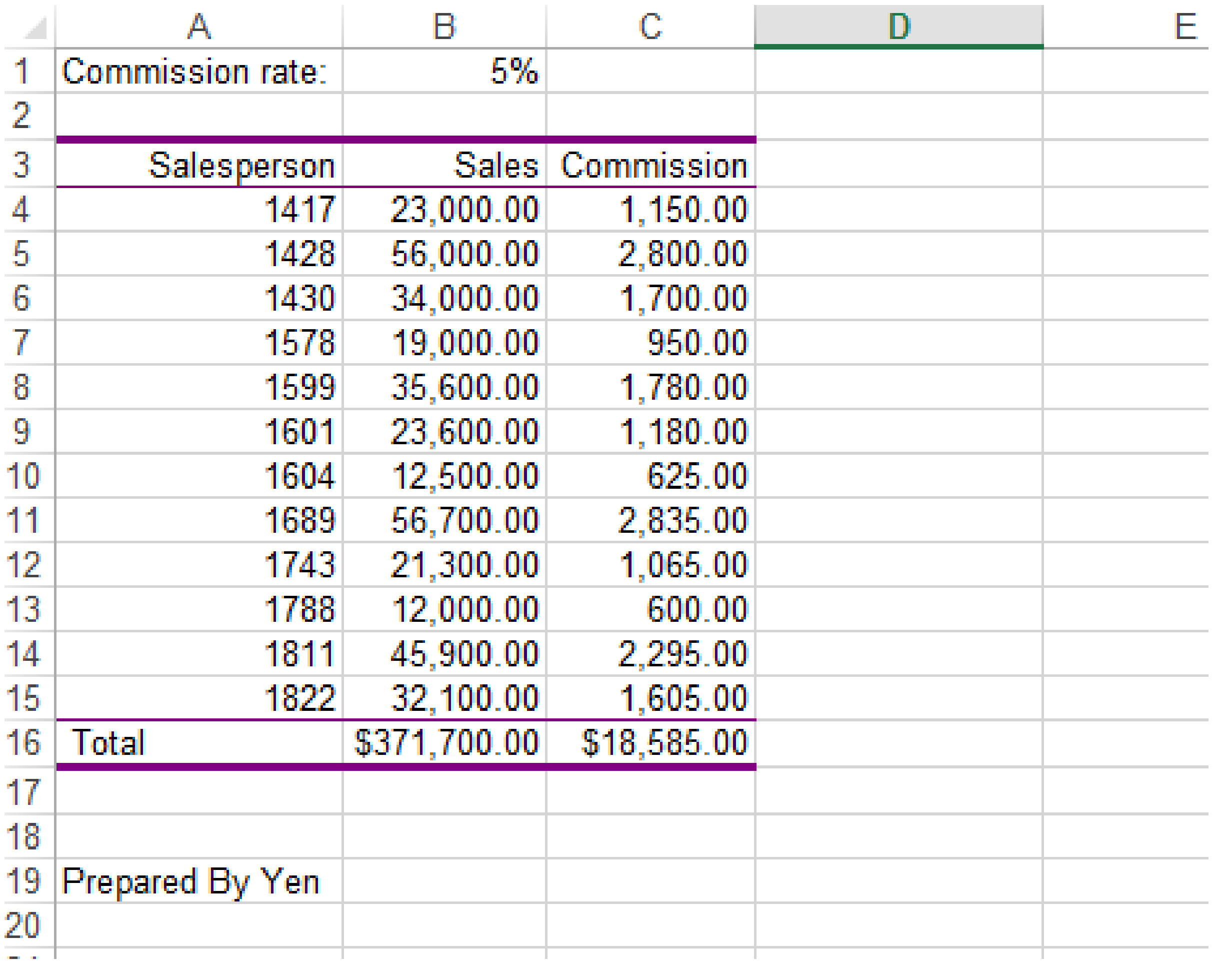
Points to cell B1 of January worksheet

Note Value property

Note Val function

Note concatenation

January Worksheet After Running the Workbook's Open Event Procedure



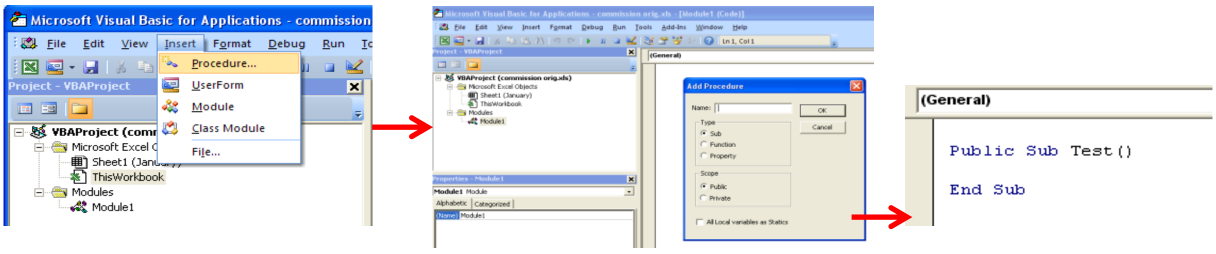
	A	B	C	D	E
1	Commission rate:	5%			
2					
3	Salesperson	Sales	Commission		
4	1417	23,000.00	1,150.00		
5	1428	56,000.00	2,800.00		
6	1430	34,000.00	1,700.00		
7	1578	19,000.00	950.00		
8	1599	35,600.00	1,780.00		
9	1601	23,600.00	1,180.00		
10	1604	12,500.00	625.00		
11	1689	56,700.00	2,835.00		
12	1743	21,300.00	1,065.00		
13	1788	12,000.00	600.00		
14	1811	45,900.00	2,295.00		
15	1822	32,100.00	1,605.00		
16	Total	$371,700.00	$18,585.00		
17					
18					
19	Prepared By Yen				
20					

Summary of procedure

- Declare string and object variables
- Store the return values for the Inputbox function in string variables
- Display the contents of one string variable on the worksheet
- Convert contents of the other string variable to a number and display on the worksheet

What should I do to create a procedure?

- Go to the Visual Basic Editor
- Double-click the object you want to put the procedure in (e.g. ThisWorkbook)
- The code window for that object will appear
- Click Insert, Procedure – you will be prompted for details – name, type, scope
- This provides a template for you to complete



```
Public Sub Test()

End Sub
```

Scope of variables

- A variable's **scope** refers to which procedures in the project can use the variable
- The Dim or Public statement is used to declare a variable
- The scope is determined by where the Dim or Public statement is entered.
- Variables in VBA can have one of three scopes:
  - Procedure-level
  - Module-level
  - Public
- Remember: a module is a collection of procedures (and or functions)

Reserving a Procedure-Level Variable

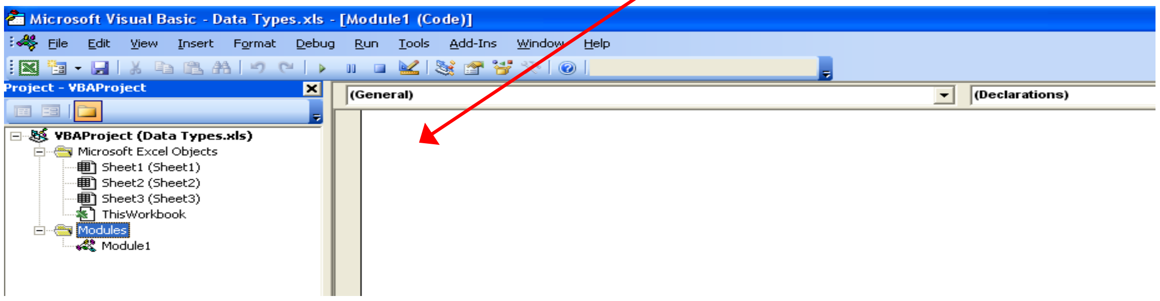
- A **procedure-level variable** is reserved, or declared, within a procedure, and it can be used only by the procedure in which it is declared
- You use the VBA **Dim statement** to reserve a procedure-level variable
- The syntax of the Dim statement is:

  Dim *variablename* As *datatype*

  where *variablename* represents the name of the variable (memory cell) and *datatype* represents its data type
- When VBA processes the Dim statement in a procedure, it reserves a memory cell to which it assigns *variablename* as the name and *datatype* as the data type
- The contents of the variable is removed from memory when the procedure ends

Module-level Variables

- Created with the Dim statement.
- The Dim statement is entered in a module's General declarations section.
- Can be used by any of the procedures in the module.
- Removed from memory when the application ends.


Microsoft Visual Basic - Data Types.xls - [Module1 (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Project - VBAProject
VBAProject (Data Types.xls)
Microsoft Excel Objects
Sheet1 (Sheet1)
Sheet2 (Sheet2)
Sheet3 (Sheet3)
ThisWorkbook
Modules
Module1
(General)
(Declarations)

Global Variables

- Created with the Public statement
- The Public statement is entered in a code module's General declarations section.
- Global variables can be used by all procedures within all modules of a project
- Removed from memory when the application ends.

Constants

- Literal constant
  - an item of data whose value cannot change while the program is running
- Examples:
  - 7
  - "Janet"

- Symbolic constant
  - a memory location whose contents cannot be changed while the program is running
- Examples:
  - conPi
  - conRate

Creating a Symbolic Constant

- A memory location whose value cannot change during run time.
- Syntax: [Public] Const *constname* [As *datatype*] = *expression*
- Examples:
  - Const conPi As Single = 3.141593
  - Public Const conMaxAge as Integer = 65

Uses Const keyword

Scope of a Symbolic Constant

- Indicates which procedures can use the symbolic constant.
- Global: **Public Const** statement in a code module's General declarations section.
- Module-level: **Const** statement in the module's General declarations section.
- Local: **Const** statement in an event procedure.

12. Practice and Apply

- Understanding how to reserve a String variable
- Understanding how to use an assignment statement to assign a value to a String variable
- Understanding how to create variables
- Understanding how to assign data types and names for object variables
- Understanding how to use the Set statement
- Understanding how to use the InputBox, MsgBox and Val functions
- Understanding how to concatenate strings
- Understanding how to code a workbook's Open Event procedure
- Complete Tutorial 5 Exercises