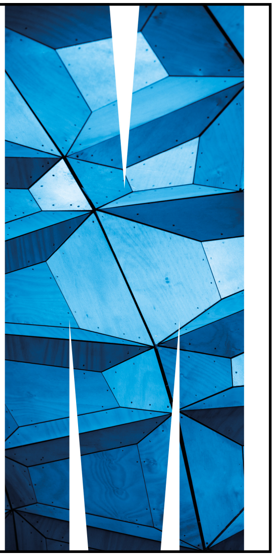
MONASH University

FIT1013 Digital Futures: IT for Business
Week 5: Fundamentals of Programming

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

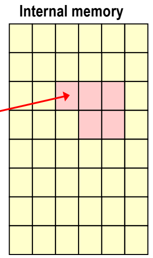
- Reserve a String variable
- Use an assignment statement to assign a value to a String variable
- Create variables including object variables
- Assign data types and names for object variables
- Use the Set statement
- Use the InputBox function to get information from the user at the keyboard
- Concatenate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

GROUP OF EIGHT AUSTRALIA



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



Variables

- 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

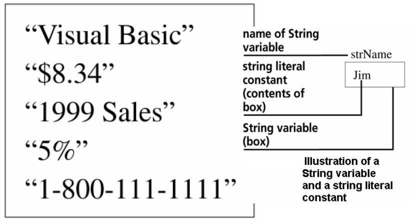
The **Dim** statement is used to reserve a procedure-level variable, and the variable can only be used by that procedure:

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**

Reserving a Procedure-level String Variable



Examples of strings

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

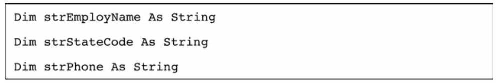
Reserving a Procedure-level String Variable

- 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

Reserving a Procedure-level String Variable

- 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

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

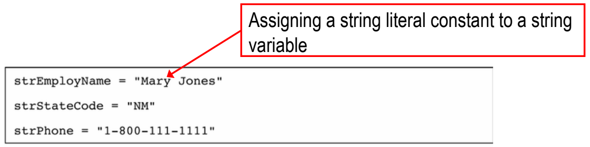
- Assignment statements are so named because they assign values to the memory cells inside the computer
- When you assign a new value to a memory cell, the new value replaces the old value, because a memory cell can store only one value at a time

Assignment statement

variablename = value

Example:

strName = "Andrew"



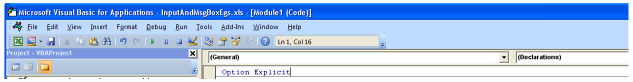
Assigning a string literal constant to a string variable

```
strEmployName = "Mary Jones"
strStateCode = "NM"
strPhone = "1-800-111-1111"
```

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

The Option Explicit Statement

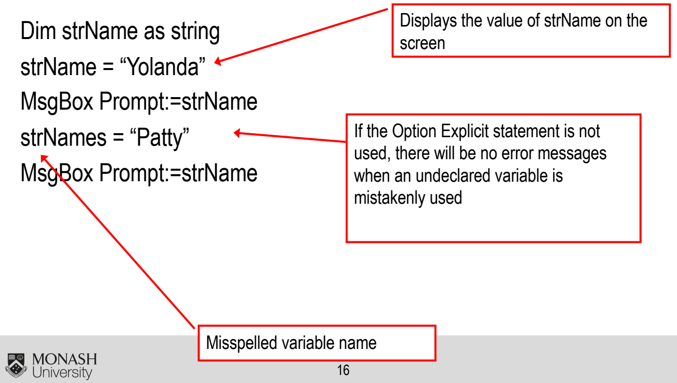
- You can use the **Option Explicit** statement to prevent VBA from reserving variables that you did not explicitly declare
  - It is considered poor programming practice to allow VBA to reserve variables "on the fly"—in other words, to reserve variables that you did not declare in a Dim statement—because 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.



```
Option Explicit
```

The Option Explicit Statement

e.g. code containing a misspelled variable name



```
Dim strName as string
strName = "Yolanda"
MsgBox Prompt:=strName
strNames = "Patty"
MsgBox Prompt:=strName
```

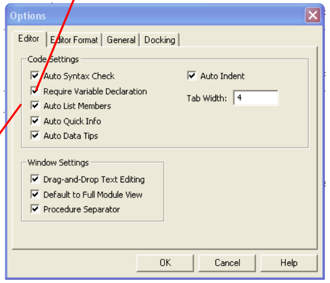
Displays the value of strName on the screen

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

Misspelled variable name

The Option Explicit Statement

- 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.



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

Automatically enters the Option Explicit statement

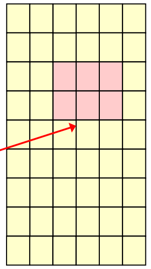
Options dialog box – showing Require Variable Declarations check box

Summary – Variables

- Reserving a String variable
- Using an assignment statement to assign a value to a String variable
- Using the Option Explicit statement – to force variable declaration

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

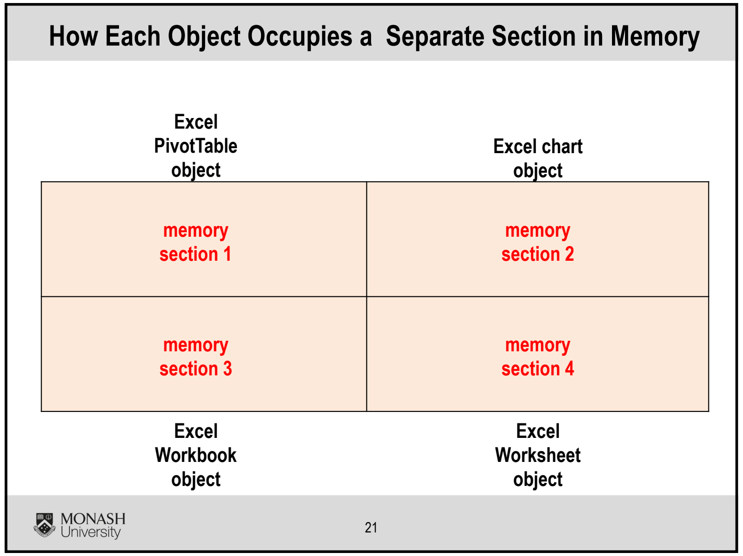


E.g. Illustration of Memory Cells that Store Property Values for the Worksheet object

Properties of the Worksheet object

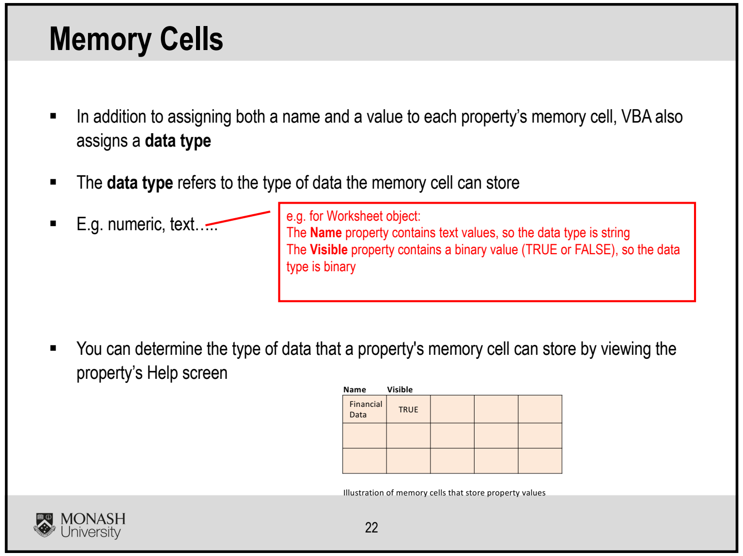
Name	Visible			
Financial Data	TRUE			

Illustration of memory cells that store property values



How Each Object Occupies a Separate Section in Memory

Excel PivotTable object	Excel chart object
memory section 1	memory section 2
memory section 3	memory section 4
Excel Workbook object	Excel Worksheet object



Memory Cells

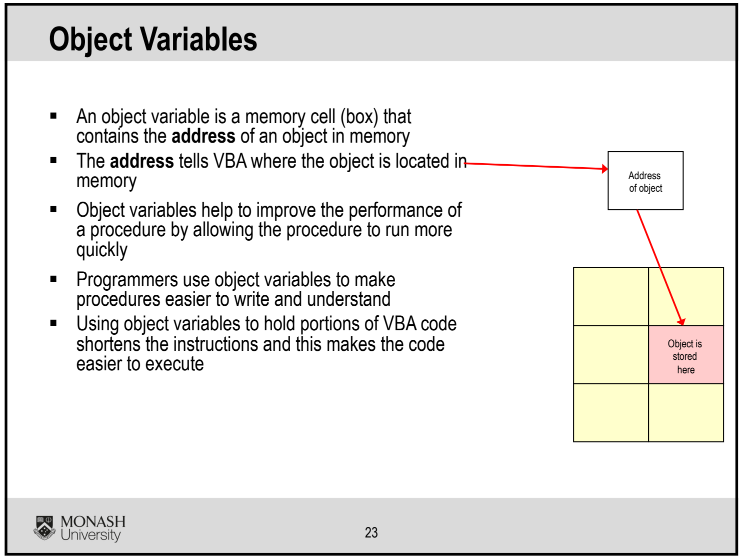
- In addition to assigning both a name and a value to each property's memory cell, VBA also assigns a **data type**
- The **data type** refers to the type of data the memory cell can store
- E.g. numeric, text…..

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

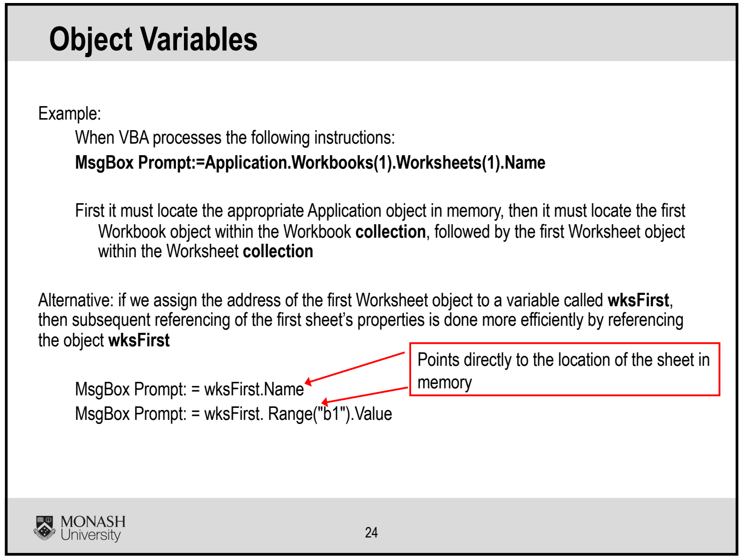


Object Variables

- 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

Address of object

Object is stored here



Object Variables

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**

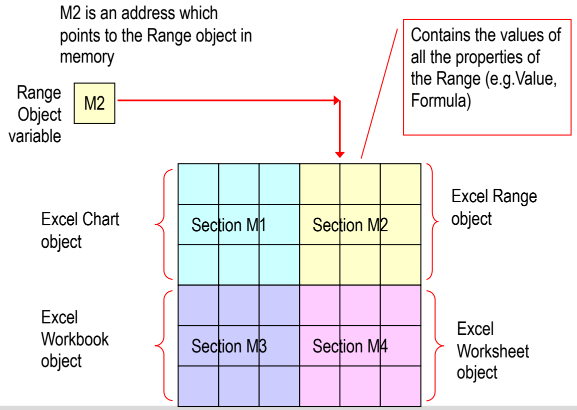
Alternative: if we assign the address of the first Worksheet object to a variable called **wksFirst**, then subsequent referencing of the first sheet's properties is done more efficiently by referencing the object **wksFirst**

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 variable

M2

Contains the values of all the properties of the Range (e.g.Value, Formula)

Excel Chart object

Excel Workbook object

Section M1

Section M2

Section M3

Section M4

Excel Range object

Excel Worksheet object

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**

Selecting the Appropriate Data Type and Name for an Object Variable

- 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	Integer	int
Boolean	bln	Long	lng
Currency	cur	Single	sng
Date/Time	dtm	String	str
Double	dbl	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

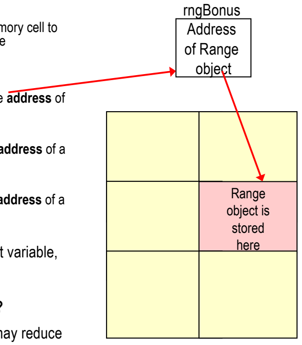
E.g.

- **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.



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

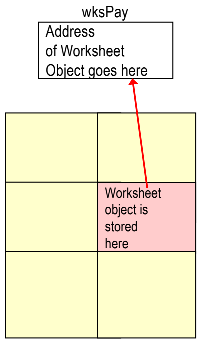
  where **objectVariableName** is the name of an object variable, and **object** is the object whose address you want to store in the variable

Using the Set Statement

- 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 – Object Variables

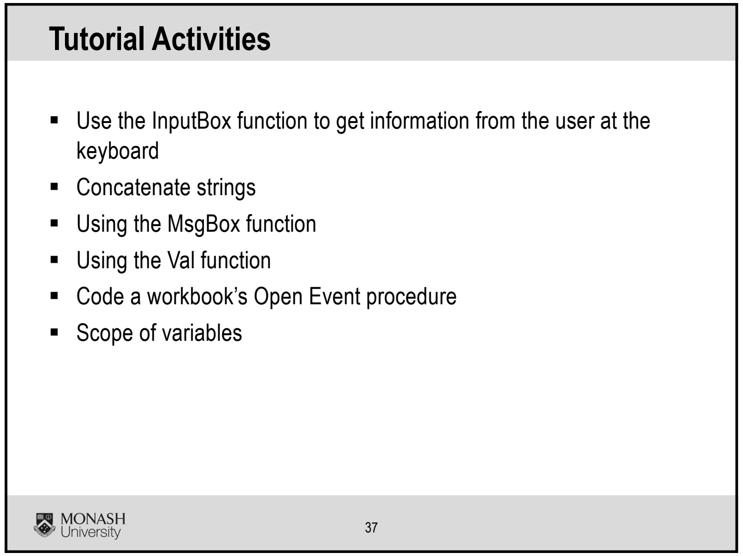
To create a procedure-level object variable, and then assign an address to it:

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.

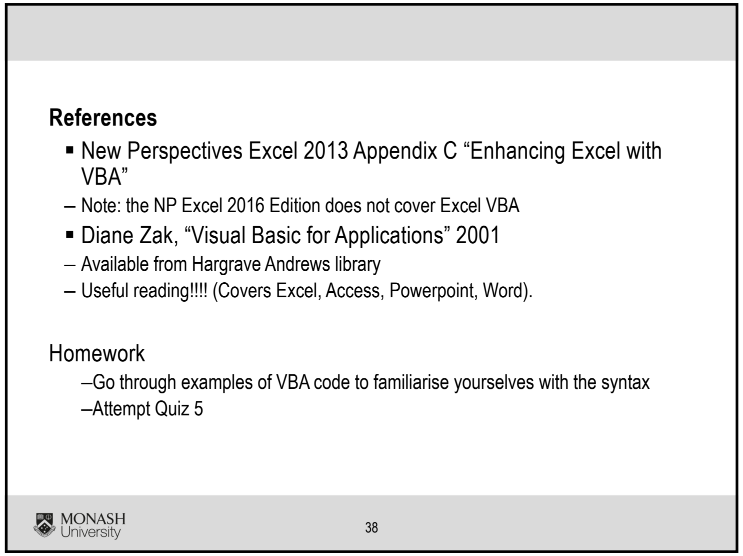
Dim wksPay as Worksheet
Set wksPay = Application.Workbooks(1).WorkSheets(1)

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



Tutorial Activities

- Use the InputBox function to get information from the user at the keyboard
- Concatenate strings
- Using the MsgBox function
- Using the Val function
- Code a workbook’s Open Event procedure
- Scope of variables



References

- 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).

Homework

–Go through examples of VBA code to familiarise yourselves with the syntax

–Attempt Quiz 5