MONASH University

FIT1013 Digital Futures: IT for Business
Week 9: *Date Variables*

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

- *Use Date and related variables*
- *Use VBA's date and time functions*

GROUP OF EIGHT AUSTRALIA

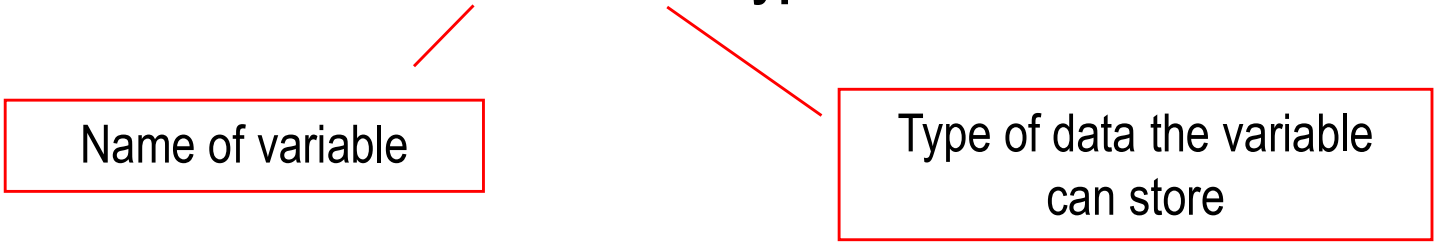
Examples

Internal storage	Represents
567.0	20th July 1901
1299.0	22nd July 1903
0.3	7.12am
0.8	7.12pm
.5692	1.39.39pm
6788.673	1st August 1918, 4.09.07pm

Reserving date variables

Recall to reserve a procedure level variable:

Dim VariableName As DataType



To reserve a procedure level Date variable:

Dim VariableName As Date

e.g.

Dim dtmStart As Date

Dim dtmBirth As Date

Examples of Dim Statements that Reserve Date Variables

- Dim dtmPay as Date
- Dim dtmEmploy as Date
- Dim dtmStart as Date
- Dim dtmEnd as Date
- Dim dtmBirth as Date

Assigning a value to a date variable

Recall the assignment statement that assigns a value to a variable:

Variablename* = *value

Examples for date variables:

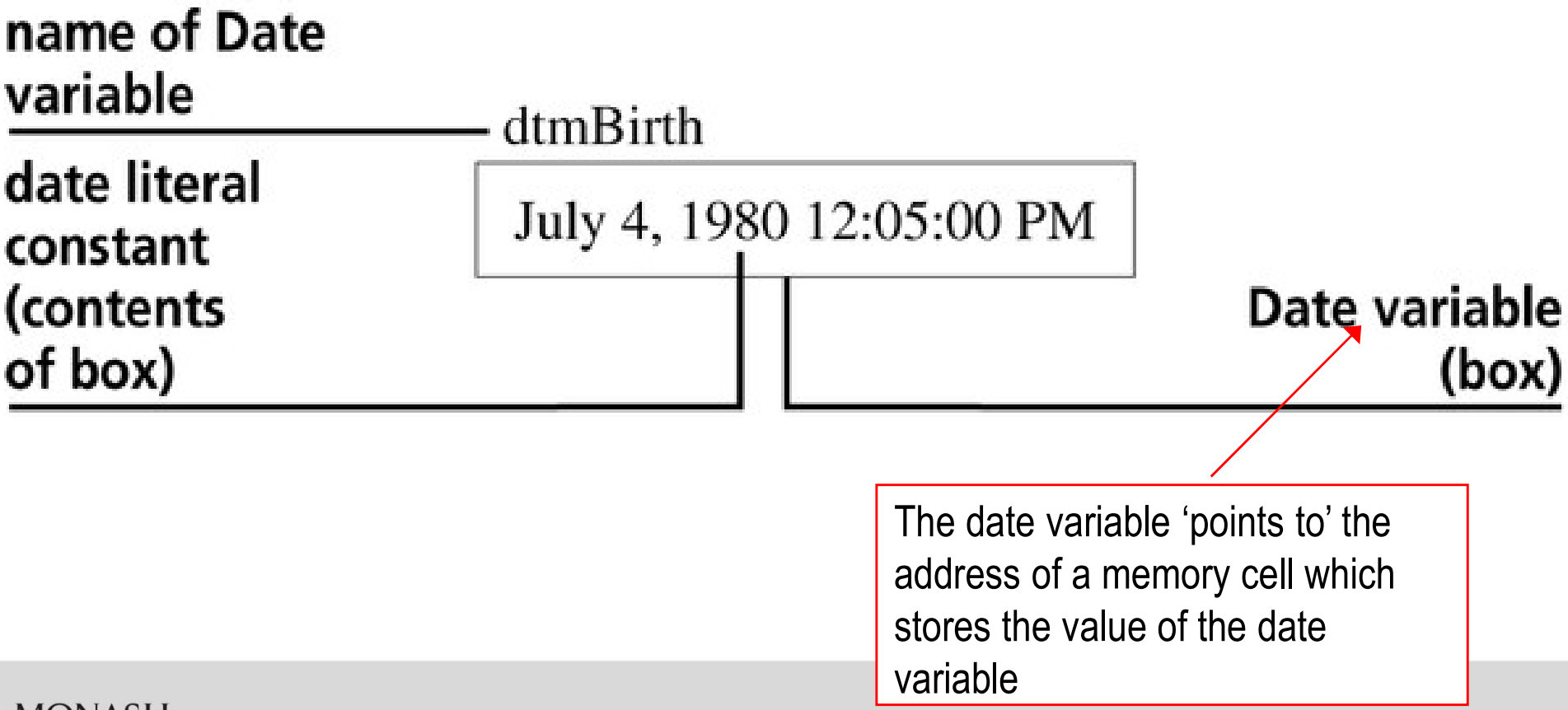
dtmBirth = #June 10, 1981#

dtmFinish = #6:48:07 PM#

Date variables store date literal constants….

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

Illustration of date literal constant stored in a date variable



Using VBA's Date, Time, and Now Functions

In addition to assigning date literal constants to Date variables, you also can assign the value returned by VBA's Date, Time, and Now functions:

- VBA's **Date** function returns the system date, which is the date maintained by your computer's internal clock
- VBA's **Time** function returns the system time, which is the time maintained by your computer's internal clock
- VBA's **Now** function returns both the system date and time

The AssignDisplayDate Procedure

```
Public Sub AssignDisplayDate()
'declare date variables
Dim dtmCurDate As Date
Dim dtmCurTime As Date
Dim dtmCurDateTime As Date
'assign values to date variables
dtmCurDate = Date
dtmCurTime = Time
dtmCurDateTime = Now
'display contents of date variables
MsgBox Prompt:=dtmCurDate & vbNewLine _
    & dtmCurTime & vbNewLine & dtmCurDateTime
End Sub
```

reserves three Date variables named dtmDurDate, dtmCurTime, and dtmCurDateTime

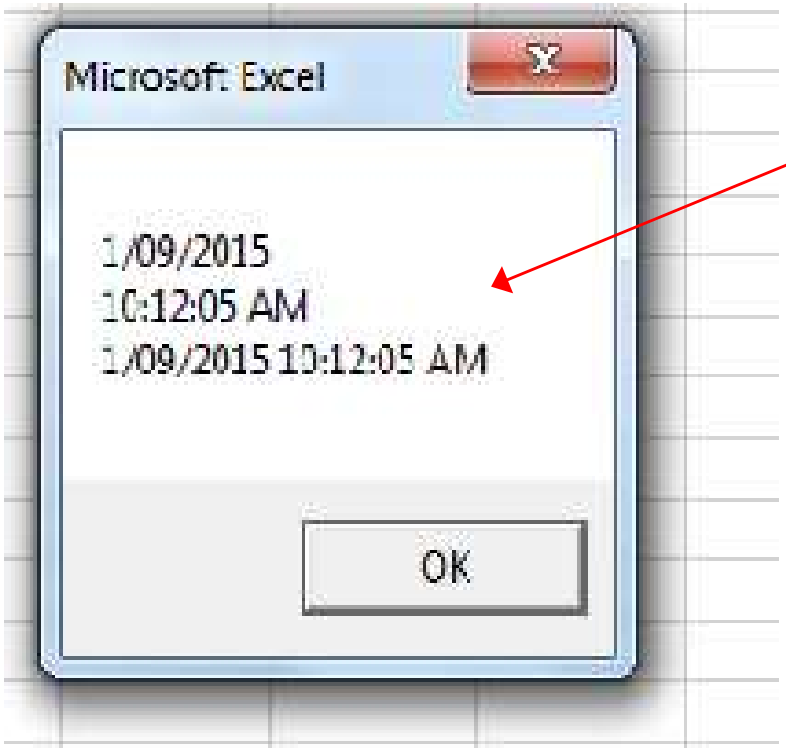
Assign values to the date variables using the **Date**, **Time** and **Now** functions

Use the underscore to indicate the code continues onto the next line

Display the values of the date variables using the MsgBox function

vbNewLine - Visual Basic Constant

Message Box Displayed by the AssignDisplayDate Procedure AssignDisplayDate.xls



Each date is displayed on a separate line

Using the Format Function

- Use the VBA **Format** function to control the appearance of dates and times
- The syntax of the Format function is:

  Format(Expression:=expression, Format:=format)

- In the syntax, **expression** specifies the number, date, time, or string whose appearance you want to format, and **format** is the name of a predefined VBA format
- E.g.

  Format(Expression:=#1/03/2004#, Format:="short date")

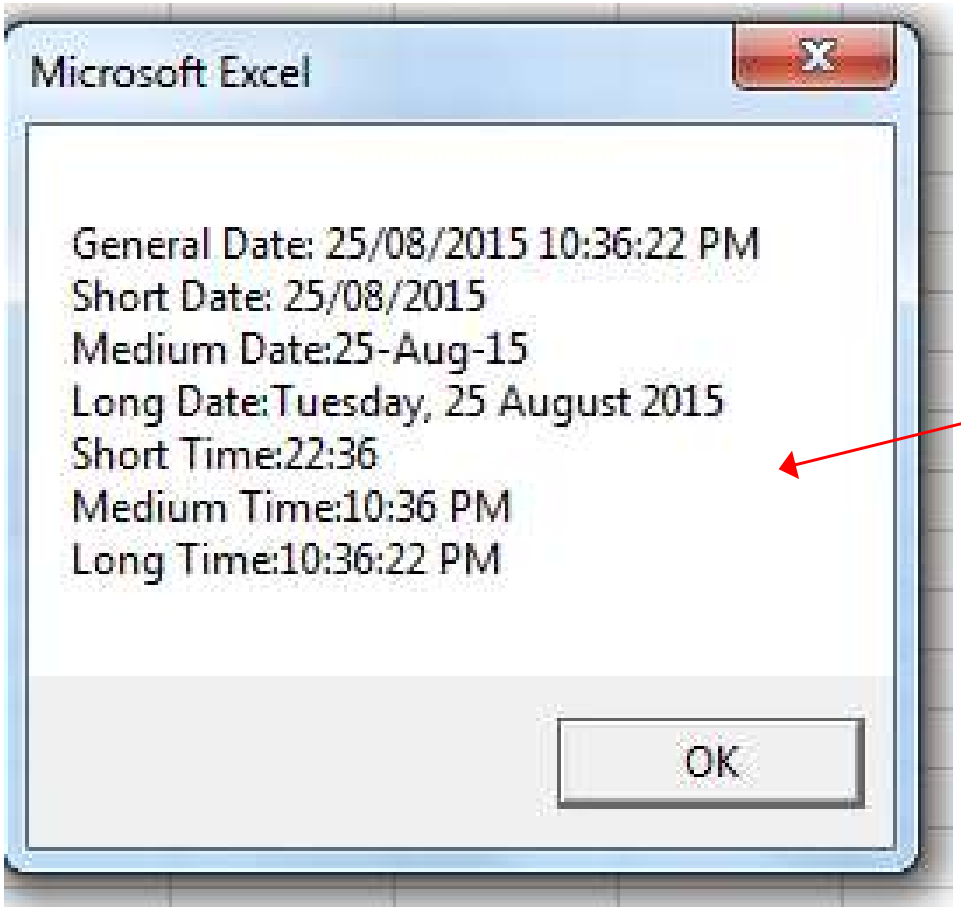
Using the Format Function

E.g. AssignDisplayDate.xls – see dateFormats() procedure

```
Public Sub dateFormats()
'declare date variables
Dim dtmEgDate As Date
'assign values to date variables
dtmEgDate = #2/18/1991 10:36:22 PM#
MsgBox Prompt:= _
    Format(Expression:=dtmEgDate, Format:="General Date") & vbNewLine _
    & Format(Expression:=dtmEgDate, Format:="Short Date") & vbNewLine _
    & Format(Expression:=dtmEgDate, Format:="Medium Date") & vbNewLine _
    & Format(Expression:=dtmEgDate, Format:="Long Date") & vbNewLine _
    & Format(Expression:=dtmEgDate, Format:="Short Time") & vbNewLine _
    & Format(Expression:=dtmEgDate, Format:="Medium Time") & vbNewLine _
    & Format(Expression:=dtmEgDate, Format:="Long Time")
End Sub
```

Note different date formats

Results of Date Format function



Note different date formats

Using Dates and Times in Calculations

- You may need to include date and time calculations in your procedures
- VBA provides two functions called **DateAdd** and **DateDiff** that you can use to perform calculations involving dates and times
- The **DateAdd** function allows you to add a specified time interval to a date or time, and it returns the new date or time
- The **DateDiff** function allows you to determine the time interval that occurs between two dates

The DateAdd function

Syntax:

DateAdd(Interval:=*interval*, **Number:=***number*, D**ate:=***date*)

Interval specifies the time units: e.g. hours, minutes, years etc..

Number specifies how many time units to add on to the date. Can be positive or negative

Date argument – can be any format

Adds 3 days to the value of the date variable dtmEgDate

E.g.

DateAdd(interval:="d", Number:=3, Date:=dtmEgDate)

AssignDisplayDate.xls – see DateAddEg() procedure

Valid Settings for the Interval Argument

interval setting	Description
"yyyy"	Year
"q"	Quarter
"m"	Month
"y"	Day of year
"d"	Day
"w"	Weekday
"ww"	Week
"h"	Hour
"n"	Minute
"s"	Second

Examples of the DateAdd Function

DateAdd function and result
`dtmNew = DateAdd(Interval:="yyyy", Number:=2, Date:=#1/1/2001#)` Result: Assigns 1/1/2003 to the dtmNew variable
`dtmDue = DateAdd(Interval:="d", Number:=15, Date:=dtmInvDate)` Result: If the dtmInvDate variable contains 1/1/2002, then 1/16/2002 is assigned to the dtmDue variable
`dtmFinish = DateAdd(Interval:="h", Number:=4, Date:=Time)` Result: If the current time is 3:54:11 PM, then 7:54:11 PM is assigned to the dtmFinish variable

`MsgBox Prompt:=DateAdd(Interval:="n", Number:=-5, _`
`Date:=#10:25:00 AM#)`
Result: Displays 10:20:00 AM in a message box

Using Dates and Times in Calculations

- The **DateDiff** function allows you to determine the time interval that occurs between two dates
- Unlike the **DateAdd** function, which returns either a future or past date or time, the **DateDiff** function returns an integer that represents the number of time intervals between two specified dates or times

The DateDiff function

Syntax

DateDiff(Interval:=*interval*, **Date1:=***date1*, **Date2:=***date2*)

Interval specifies the time units: e.g. hours, minutes, years etc..

date1 and *date2 :* dates needed in the calculation.

E.g.

MsgBox prompt:="Date diff: " & DateDiff("yyyy", #2/18/1991#, #1/27/2015 10:36:22 PM #)

Examples of the DateDiff Function

DateDiff function and result

`MsgBox Prompt:=DateDiff(Interval:="yyyy", Date1:=#1/1/2001#, _`
`Date2:=#1/1/2003#)`
Result: Displays 2 in a message box

`MsgBox Prompt:=DateDiff(Interval:="yyyy", Date1:=#1/1/2003#, _`
`Date2:=#1/1/2001#)`
Result: Displays -2 in a message box

`intDay = DateDiff(Interval:="d", Date1:=dtmInvDate, _`
`Date2:=dtmDue)`
Result: If the dtmInvDate variable contains 1/1/2002 and the dtmDue variable contains 1/31/2002, then 30 is assigned to the intDay variable

`intHour = DateDiff(Interval:="h", Date1:=#3:54:11 PM#, _`
`Date2:=Time)`
Result: If the current time is 7:54:00 PM, then 4 is assigned to the intHour variable

`MsgBox Prompt:=DateDiff(Interval:="n", Date1:=#10:25:00 AM#, _`
`Date2:=#10:20:00 AM#)`
Result: Displays -5 in a message box

Examples of Using the DateValue and TimeValue Functions to Convert Strings to Dates and Times

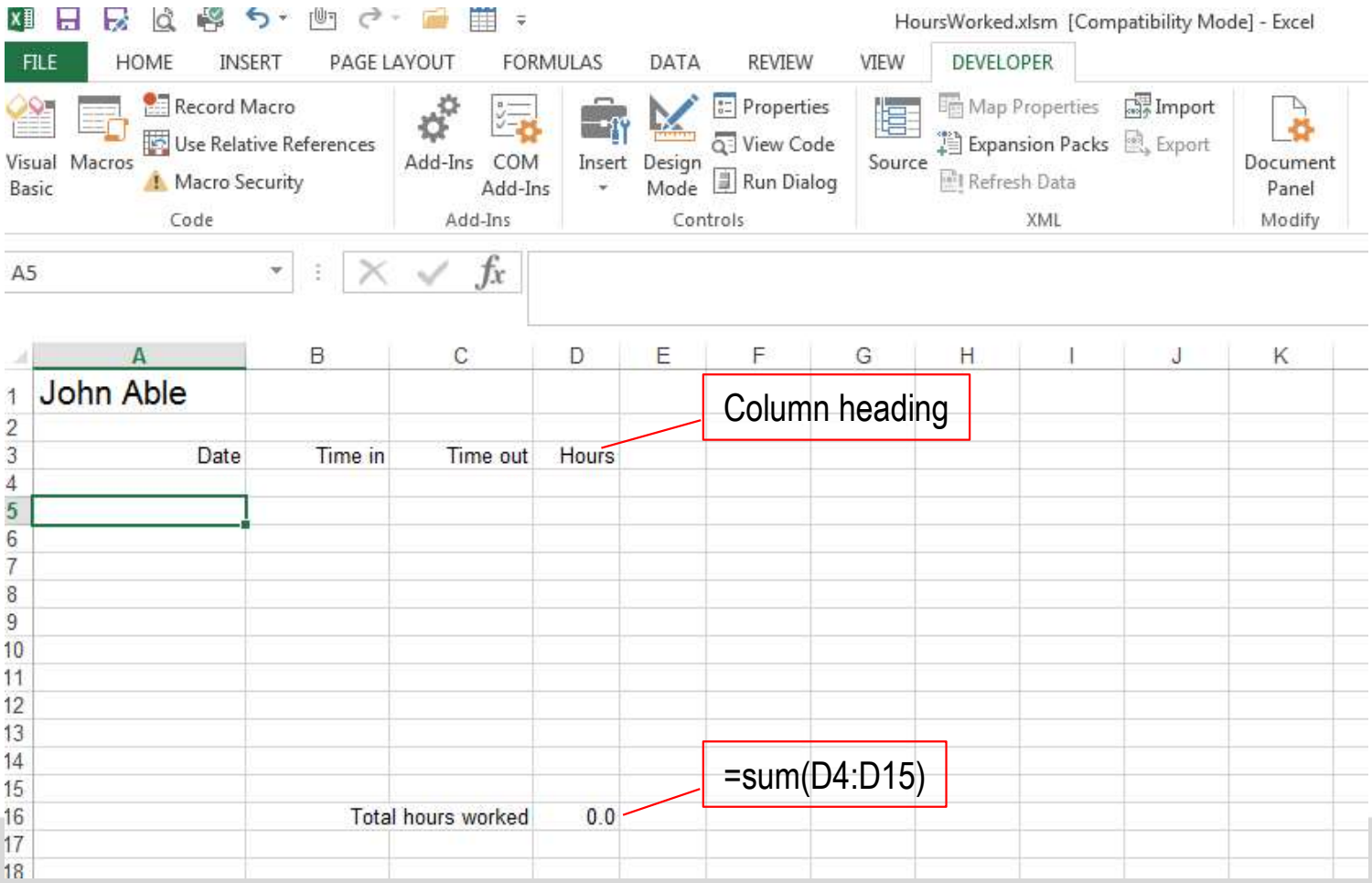
DateValue function	Result
`dtmShip = DateValue(Date:="3/5/2002")`	Converts the "3/5/2002" string to a date, and then assigns the resulting date, 3/5/2002, to the dtmShip Date variable
`dtmBirth = DateValue(Date:=strBirth)`	Assuming the strBirth variable contains the string "October 11, 1950", the statement converts the string to a date and then assigns the result, 10/11/1950, to the dtmBirth Date variable
TimeValue function	**Result**
`dtmIn = TimeValue(Time:="5:30pm")`	Converts the "5:30pm" string to a time, and then assigns the resulting time, 5:30:00 PM, to the dtmIn Date variable
`dtmOut = TimeValue(Time:=strOut)`	Assuming the strOut variable contains the string "3:45am", the statement converts the string to a time and then assigns the result, 3:45:00 AM, to the dtmOut Date variable

Excel Example: Creating the CalcHours Macro Procedure

This exercise involves:

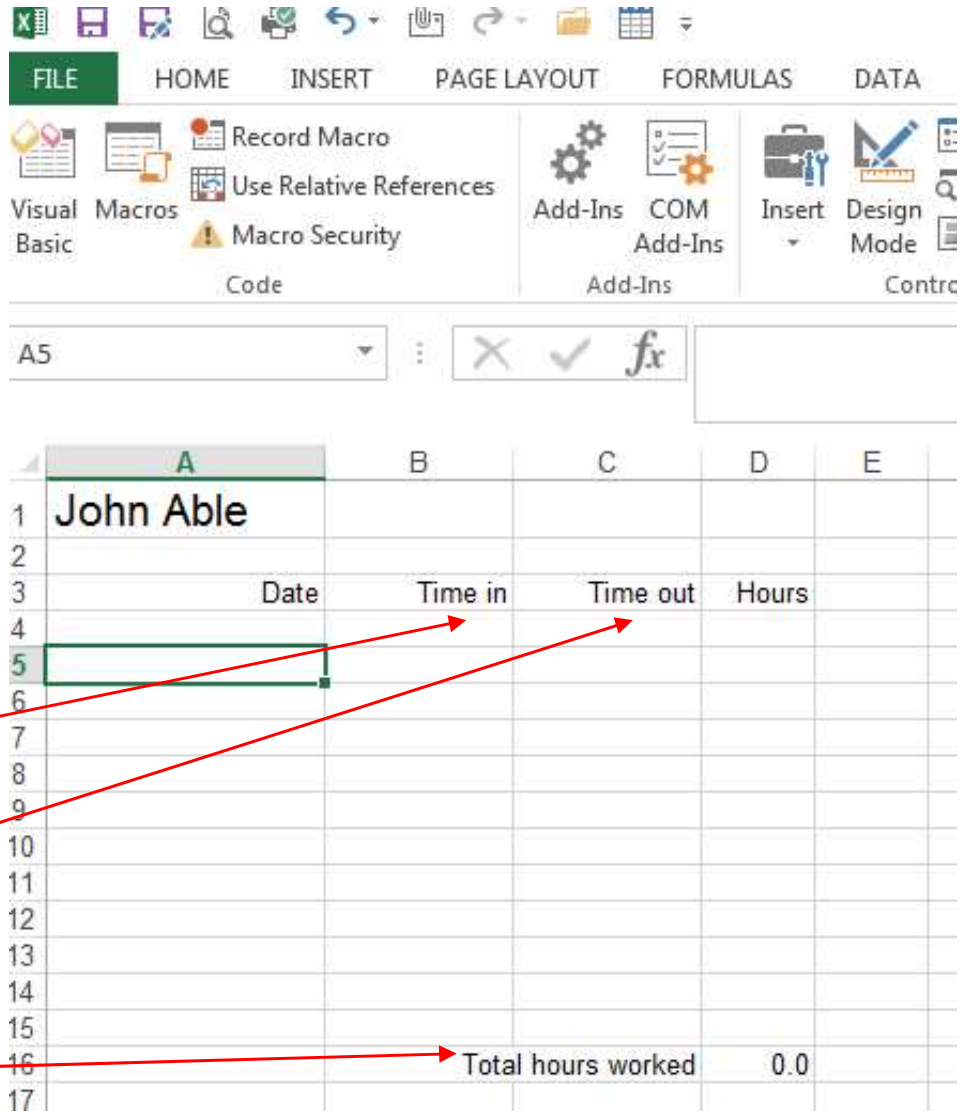
- Finding the total number of hours worked each day
- Calculating the total hours worked per fortnight for each employee

Hours Worked.xls



Pseudocode for the CalcHours Procedure

1. Use the **InputBox** function to prompt the user to enter the **starting time**. Store the response in a string variable named **strIn**
2. Use the **InputBox** function to prompt the user to enter the **ending time**. Store the response in a string variable named **strOut**
3. Use the **TimeValue** function to convert the string stored in **strIn** to a time, then assign the result to a date variable named **dtmIn**
4. Use the **TimeValue** function to convert the string stored in **strOut** to a time, then assign the result to a date variable named **dtmOut**
5. assign the system date to the active cell in column A
6. assign the starting time (stored in **dtmIn**) to the cell located one column to the right of the active cell. I.e. in column B
7. assign the ending time (stored in **dtmOut**) to the cell located two columns to the right of the active cell. I.e. in column C
8. use the **DateDiff** function to calculate the number of hours worked. Assign the result to the cell located three columns to the right



Creating the CalcHours Macro Procedure

Declare string and object vars, set the object variables:

```
Public Sub CalcHours()
 'declare variables and assign address to object variable
    Dim strIn As String
    Dim strOut As String
    Dim dtmIn As Date
    Dim dtmOut As Date
    Dim rngActive As Range
    Set rngActive = Application.ActiveCell
End Sub
```

User entered times are Stored as strings

The date variables are used to store the actual times in the 'time' format

ActiveCell Returns a Range object that represents the active cell in the active window

This range variable stores the active cell Address in the worksheet

Partially Completed CalcHours Procedure

```
Public Sub CalcHours()
    'declare variables and assign address to object variable
    Dim strIn As String, strOut As String, dtmIn As Date, dtmOut As D
    Dim rngActive As Range
    Set rngActive = Application.ActiveCell
    'enter starting and ending time
    strIn = InputBox(prompt:="Enter the starting time:", _
       Title:="Start Time", Default:=#9:00:00 AM#)
    strOut = InputBox(prompt:="Enter the ending time:", _
       Title:="End Time", Default:=#5:00:00 PM#)
    'convert strings to times
    dtmIn = TimeValue(Time:=strIn)
    dtmOut = TimeValue(Time:=strOut)
    'assign values to worksheet cells
    rngActive.Value = Date
End Sub
```

Prompts user for Start/Finish time and stores response in strIn/strOut

Convert the string values to Dates (times)

Assign the System Date to the active cell

The Offset Property of the Range object

- You can use a Range object's **Offset** property to refer to a cell located a certain number of rows or columns away from the range itself
- The syntax of the Offset property is

  rangeObject.Offset([rowOffset] [,columnOffset])
- You use a **positive** rowOffset to refer to rows found below the rangeObject, and you use a **negative** rowOffset to refer to rows above the rangeObject
- You use a **positive** columnOffset to refer to columns found to the right of the rangeObject, and you use a **negative** columnOffset to refer to columns to the left of the rangeObject

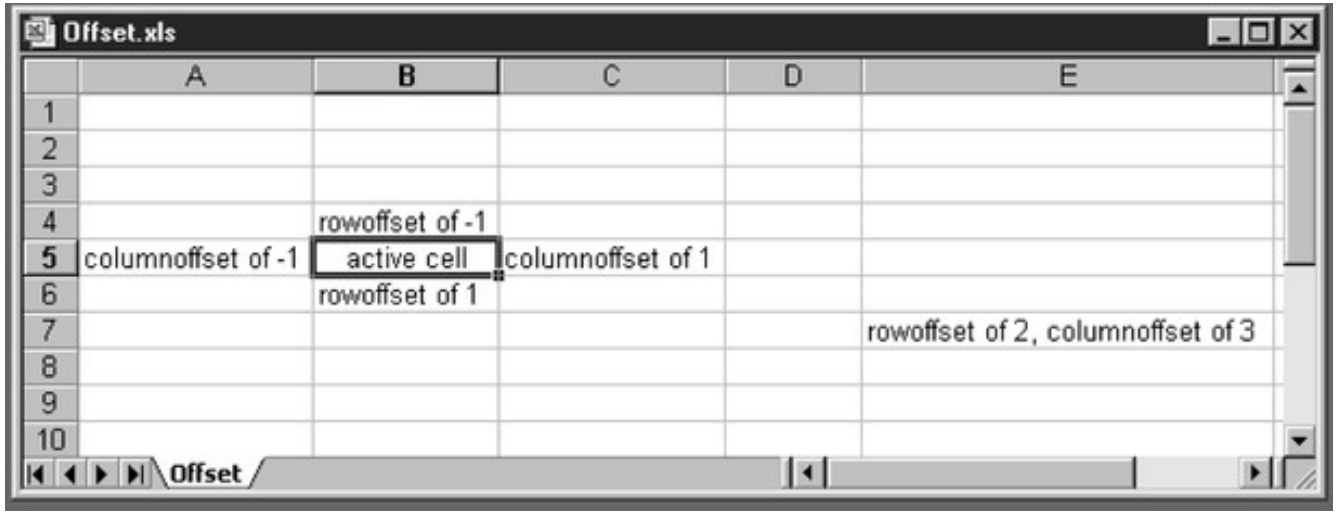
Illustration of the Offset Property

For example:

If rangeObject (I.e. active cell) is B5 then

- rowOffset of 1 refers to B6
- rowOffset of –1 refers to B4
- columnOffset of 1 refers to C5
- columnOffset of –1 refers to A5

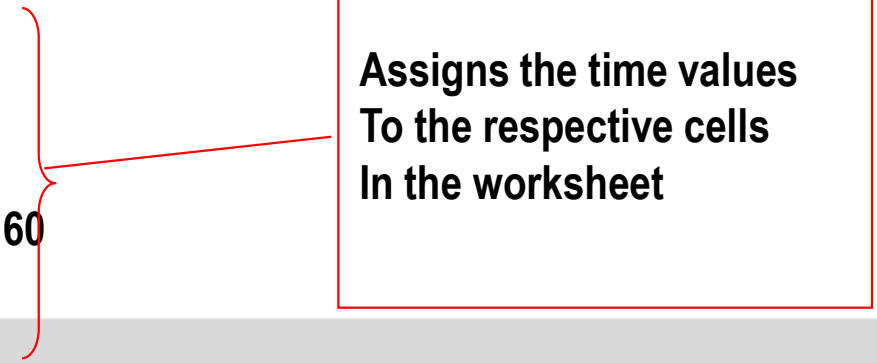
What does rangeObject.Offset(2,3) refer to?



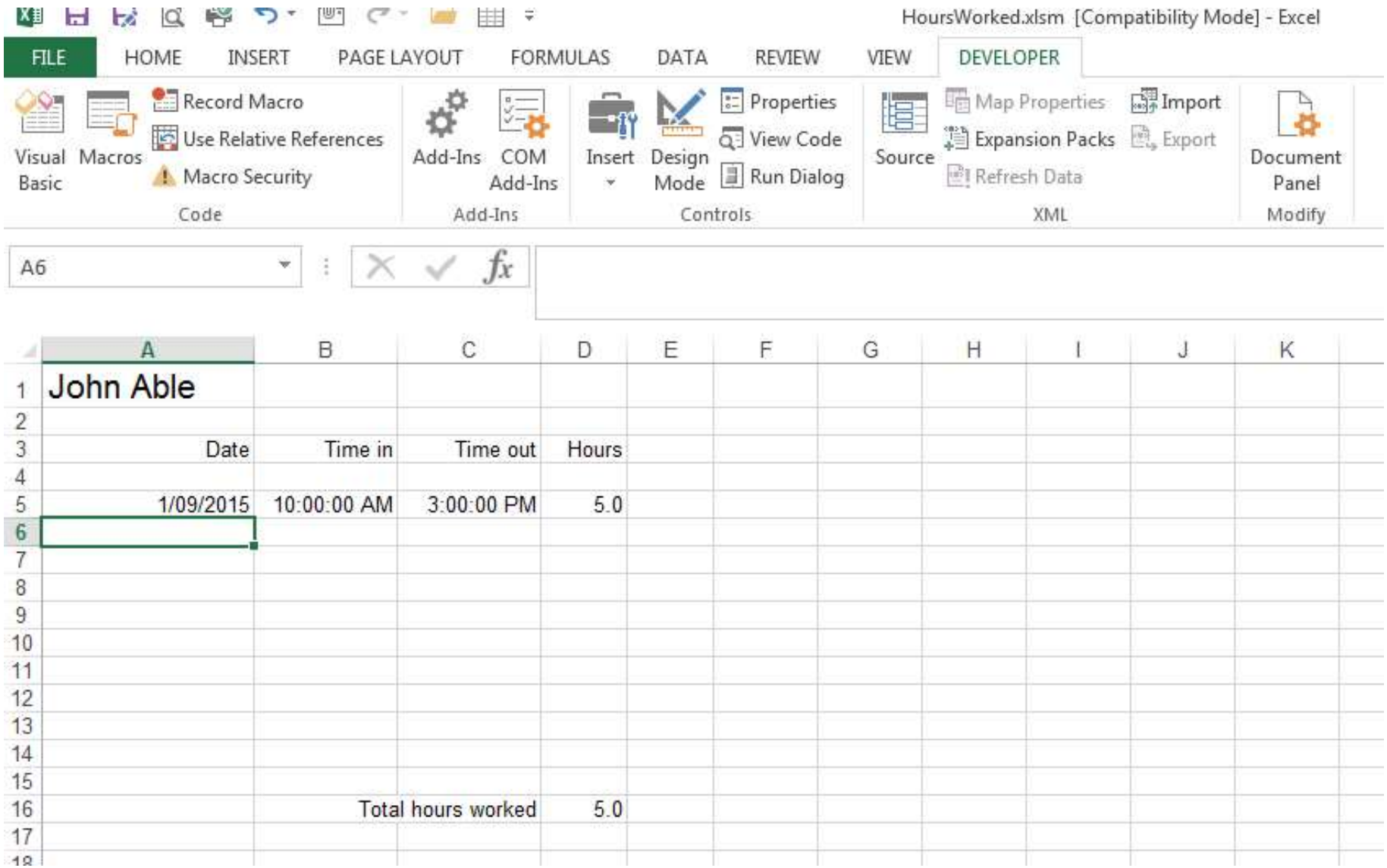
E7

Completed CalcHours Procedure

```
Public Sub CalcHours()
    'declare variables and assign address to object variable
    Dim strIn As String, strOut As String, dtmIn As Date, dtmOut As Date
    Dim rngActive As Range
    Set rngActive = Application.ActiveCell
    'enter starting and ending time
    strIn = InputBox(prompt:="Enter the starting time:", _
        Title:="Start Time", Default:=#9:00:00 AM#)
    strOut = InputBox(prompt:="Enter the ending time:", _
        Title:="End Time", Default:=#5:00:00 PM#)
    'convert strings to times
    dtmIn = TimeValue(Time:=strIn)
    dtmOut = TimeValue(Time:=strOut)
    'assign values to worksheet cells
    rngActive.Value = Date
    rngActive.Offset(columnoffset:=1).Value = dtmIn
    rngActive.Offset(columnoffset:=2).Value = dtmOut
    rngActive.Offset(columnoffset:=3).Value = _
        DateDiff(interval:="n", date1:=dtmIn, date2:=dtmOut) / 60
End Sub
```



Worksheet after running the procedure



The IsDate() function

To check whether the InputBox function has returned a valid date use the IsDate function.

Syntax:

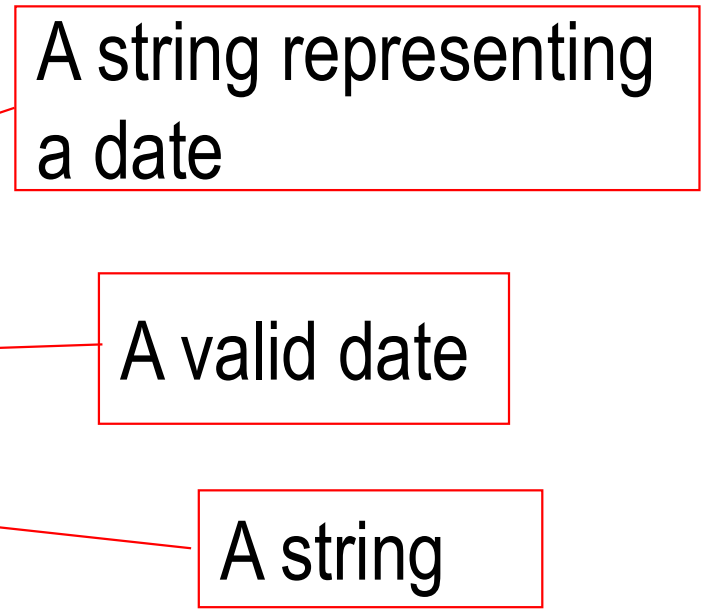
IsDate(*expression*)

The required *expression* argument is a Variant containing a date expression or string expression recognizable as a date or time.

IsDate returns either True or False depending on whether the *expression* represents a valid date.

IsDate() example

```
Dim strDate1 As String
Dim dtmDate2 As Date
Dim strDate3 As String
Dim blnCheck As Boolean
strDate1 = "February 12, 2010"
dtmDate2 = #2/12/2009#
strDate3 = "Hello"
blnCheck = IsDate(strDate1)
Debug.Print blnCheck ‘returns True
blnCheck = IsDate(dtmDate2)
Debug.Print blnCheck ‘returns True
blnCheck = IsDate(strDate3)
Debug.Print blnCheck ‘returns false
```



Updated CalcHours procedure

```
'enter starting and ending time     Hours Worked.xls
    strIn = InputBox(prompt:="Enter the starting time:", _
        Title:="Start Time", Default:=#9:00:00 AM#)
    Debug.Print IsDate(strIn)
    strOut = InputBox(prompt:="Enter the ending time:", _
        Title:="End Time", Default:=#5:00:00 PM#)
    Debug.Print IsDate(strOut)
    If Not (IsDate(strIn)) Or Not (IsDate(strOut)) Then
        MsgBox ("invalid times")
    Else
    'convert strings to times
        dtmIn = TimeValue(Time:=strIn)
        dtmOut = TimeValue(Time:=strOut)
    'assign values to worksheet cells
        rngActive.Value = Date
        rngActive.Offset(columnoffset:=1).Value = dtmIn
        rngActive.Offset(columnoffset:=2).Value = dtmOut
        rngActive.Offset(columnoffset:=3).Value = _
            DateDiff(interval:="n", date1:=dtmIn, date2:=dtmOut) / 60
    End If
End Sub
```