Outline

1. More VBA Syntax
2. Control Statements
3. Workbook and Worksheet Objects
4. Range Object
5. Applications
Outline

1. More VBA Syntax
2. Control Statements
3. Workbook and Worksheet Objects
4. Range Object
5. Applications
Global variables

The variables have three different scopes: procedure-level, private-global and public-global.

- **Local variables**: available only within the procedure. Syntax:
  Dim name_variable As Type

- **Private Global variables**:
  - Syntax: Private name_variable As Type at the top of the module
    (or use the keyword Dim)
  - Available only to other procedures within the same module.

- **Public Global variables**:
  - Syntax: Public name_variable As Type at the top of the module
  - Available only to all modules within a project.
Arrays

- Group of variables with the same type and name.
- You can specify the array’s size by using the syntax: Dim name_array(lowerbound To upperbound) As type.
- You can access elements by indexing: name_array(i).
- If you want to declare a lower bound of 1 always use: Option Base 1 at the top of the module. Then just define: Dim name_array(upperbound) As type
Multidimensional Arrays

- VBA allows to create arrays up to 60 dimensions.
- Syntax: Dim Multiarray(size_1,...,size_k)
Example

Compute two uniform(0,1) numbers, store them using an array and compute their sum.
Redimension an Array

You can change the size of an array by using the ReDim statement.

Dynamic array: you don’t define the array’s size unless you use ReDim.
Example

What is this code doing?

Sub example2()
Dim n, dvec() As Double
n=Range("dvec").Count
ReDim dvec(n)
End Sub
Functions

- Functions return values (call by a subroutine or another function).
- Syntax:
  
  Function name(arguments)
  ...
  ...
  End Function

- There are two types of functions:
  - Public: all modules can access a public function. It appears in the Insert dialog box within Excel.
  - Private: this function can be accessed by the functions within the same module. (Private Function)
Create a function `BSCallValue(S, X, r, q, T, sigma)` that computes the BS Call option formula with continuous dividends, where $S$: stock price, $X$: strike price, $r$: interest rate, $q$: dividend rate, $T$: time to maturity and $sigma$: volatility.
Outline

1. More VBA Syntax
2. Control Statements
3. Workbook and Worksheet Objects
4. Range Object
5. Applications
Conditional Statements (If)

- Syntax:
  
  ```
  If condition Then
  ...
  Else
  ...
  End If
  ```

- In order to do comparisons, these operators are important: Equal (=), greater than (>), not equal (<>), greater or equal to (>=).

- Logical operators: And, Not and Or.
Loop Statements (For)

- **Syntax:**
  
  ```plaintext
  For counter=a To b
  ...
  Next
  ```

- **You can change the increments with the statement Step:**
  
  ```plaintext
  For counter=a To b Step c
  ...
  Next
  ```
Create a function that computes $n!$, where $n \geq 1$. (Without using excel functions).
With the Select Case command, you can execute a specific code based upon a value of a statement.

Syntax:

Select Case variable
Case value1
...
Case value2
...
Case valuek
...
End Select

The Case statement is very general, you can include comparisons or ranges.
You can do the same example (factorial), with Select Case:

Function example5(n)
'computes n! for n>=1 (using select case)
Dim i As Integer
Dim tempn As Double
Select Case n
Case 1
example5 = 1
Case Is > 1
tempn = 1
For i = 1 To n
tempn = tempn * i
Next
example5 = tempn
End Select
End Function
More Control Statements

- Do While: do instructions while a condition is true
  Do While condition
  ...
  Loop

- Do Until: so instructions until a condition is satisfied
  Do Until condition
  ...
  Loop
Outline

1. More VBA Syntax
2. Control Statements
3. Workbook and Worksheet Objects
4. Range Object
5. Applications
Object variables

- Definition of Objects: `Dim object_name as Type`
- Assignment: `Set object_name=object`

Example:

```
Dim book2 As Workbook
Set book2 = ActiveWorkbook
MsgBox book2.Name
```
Workbook Object

The Workbooks collection has several methods:

- **Open**: it allows to open an existing workbook. You have to specify the workbook path if it is not within the current folder:
  
  \[ \text{Workbooks.Open("path")}. \]

- **Save**: save the current workbook: \[ \text{ThisWorkbook.Save("newpath")} \]
  or \[ \text{ActiveWorkbook.Save("newpath")} \], or a pre-specified workbook: \[ \text{Workbooks("path")}.Save("newpath") \].

- **Activate**: activate the workbook: \[ \text{Workbooks("path")}.Activate \].

- **Close**: close a specified workbook.

- **Add**: add a new workbook to Workbooks collection: \[ \text{Workbooks.Add} \].
Worksheet Object

Some methods:

- **Add**: same as before: `Worksheets.Add`
- **Delete**
- **Copy**: with its two arguments: Before and After.
- **Protect**: password protect a worksheet so others can’t modify it.
  (Unprotect)
Worksheet Object

Some properties:

- **Count**: number of Worksheets in the collection.
- **Visible**: Boolean value. (True or False)
- **Name**
“For Each” Command

Syntax:

For Each object in collection
...
Next

Example:

Sub shownames()

Dim wb As Worksheet

For Each wb In ActiveWorkbook.Worksheets
MsgBox wb.Name
Next

End Sub
“With” statement

Syntax:

With object
    .property
    .method
End With
Outline

1. More VBA Syntax
2. Control Statements
3. Workbook and Worksheet Objects
4. Range Object
5. Applications
- The Range Object can be: a single cell, an entire column/row or a selection of multiple rows.
- You can call the Range Object: Range("A1:B5") or Range("myrange").
- You can call entire columns or rows by using the Columns and Rows collections.
Methods

- Select
- Delete
- Copy
- PasteSpecial
- Resize: resize a range. Arguments: Rowsize, ColumnSize.
Properties

- **Offset**: you can define a range which is an offset of a predefined range. Example: `Range("A1:B1").Offset(2,2)`.
- **Hidden**: hide a specific range. (Boolean value)
- **Name**: name of a range within Excel worksheet.
- **Value**
- **Formula**
- **Address**
The most important is **Cells** property:

- Returns the values, methods and properties of a single cell: `Range("myrange").Cells(row,column)`
- You can change format, border style, values, names, etc. on cells.
- The Cells collection has the same properties and methods that Range.
Some examples

Use “ActiveCell” in the following two examples:

- **Example 1:** Create a macro that shows the formula and the value in the current cell.
- **Example 2:** Create a macro that adds a border to the active cell and select the cell located 2 cells down and 2 cells right from the active one.
Outline

1. More VBA Syntax
2. Control Statements
3. Workbook and Worksheet Objects
4. Range Object
5. Applications
Reminder (see Bjork, 2009):

- **Bond price process:**
  
  \[ B_0 = 1 \]
  \[ B_1 = 1 + R \]

- **Stock price:**
  
  \[ S_0 = s \]
  \[ S_1 = \begin{cases} 
  s \cdot u & \text{with prob } p_u \\
  s \cdot d & \text{with prob } p_d 
  \end{cases} \]
The market model is arbitrage free iff there exist a martingale measure $Q$:

\[
\begin{aligned}
q_u &= \frac{(1+R)-d}{u-d} \\
q_d &= \frac{u-(1+R)}{u-d}
\end{aligned}
\]

and if there exists a simple contingent claim $X$, its arbitrage-free price is:

\[
\Pi(X) = \frac{1}{1+R} E^Q[X]
\]