Introduction to VBA
(See Chapters 1 - 4 of Albright)

Kipp Martin
University of Chicago
Booth School of Business

January 4, 2012
Excel Files

Files used in this lecture.

▶ vbIntroClean.xlsx
Outline

Programming Languages
  Compilers
  Scripting Languages

Motivation for VBA

Brief History

The VB Editor

Object Oriented Programming

Generating Code

Events
Programming Languages

Programming languages are used to create applications. For example, the Microsoft Excel application was written in the C programming language.

Learning Objectives:

1. Understand the difference between a compiled programming language and a scripting language.

2. Understand the role of Excel/VBA and MATLAB

3. Understand key tradeoffs
Programming Languages

The following are some popular high-level, languages that have been used to create applications.

▶ **COBOL** (common business-oriented language) An old language from the 1950s which was designed for accounting and transaction processing.

▶ **FORTRAN** (formula translation) An excellent language for numerically intensive work.

▶ **C, C++** Good for numerical applications and scientific computing. It is also used for writing many desktop applications. C++ is the object oriented version of C.

▶ **Java** An extremely popular language for developing Internet business applications.

▶ **C#** A language from Microsoft designed to facilitate the development of Internet business applications.
Compilers

A programmer writes a set of instructions (source code) that constitute a program. The source code must get translated into machine code.

Figure: The compilation process for different operating systems.
Compilers

A few key points.

- The executable code created by the compiler is *platform dependent*.

- The executable code generated by the compiler cannot be modified or edited – you must go back to the source code.

- The executable code is *binary* as opposed to source code which is *text*.

- When purchasing software, negotiate to get the source code if possible.
Scripting Languages

An alternative to a compiled language such as C or Java is an interpreted or scripting language. Two good examples are the MATLAB scripting language and VBA. What is the difference between a scripting language and a compiled language?

Figure: A compiled program is a process directly controlled by the OS.
Scripting Languages

A scripting language need an *interpreter* or *host*.

**Figure:** A interpreted program is a process directly controlled by host which in turn is controlled by the OS.
Scripting Languages

Scripting languages are easier than compiled languages.

- A compiled programming language generally has a much larger vocabulary/syntax to learn than a scripted language. There is more to learn.

- A compiled programming language is usually conceptually more difficult than a scripted language. Concepts of such as pointers, polymorphism, virtual classes, templating etc. are nontrivial to learn.

- There is a bigger burden placed on the programmer using a compiled language. For example, garbage collection or allocating memory when it is needed.

- A compiled language is often more “rigid.” This rigidity has its advantages, but also slows coding down.
Scripting languages give “instant gratification.”

```matlab
>> x = 10;
>> y = 0;
>> x/y;
```

MATLAB will immediately come back with the message “Warning: Divide by zero.” MATLAB is “interpreting” the commands as you type them in.

With a compiled language you must compile and link the entire problem and then run it in order to discover the problem. The compilation and linking phase can be very time consuming. You get feedback much faster with the scripting language.
Scripting Languages

**Downside:** a scripting application will not run as fast as a compiled application. The scripting language application is running inside the host which is running inside the operating system. When should you use a scripting language? More later!

Example scripting languages:

1. **Javascript** – this scripting language is very popular and is used to add functionality to a browser. Browsers such as Internet Explorer or Firefox serve as the interpreter/host for Javascript.

2. **Visual Basic for Applications (VBA)** – this scripting language is immensely popular in the Microsoft world. It is used to add functionality to the suite of Office products. Each Office product such as Excel or Word serves as the host.

3. The **MATLAB scripting language** and the MATLAB program is the host.

4. **Python** and an interpreter
Scripting Languages

Scripting languages make for easy programming.

However, there are *tradeoffs*!

A program written in Excel VBA or MATLAB script *will not* execute as fast as code written in C. There is a *performance hit* for VBA and MATLAB.

**Key Idea:** YOU must make a decision regarding the tradeoff between *execution time* and *development time*.

Computers have become so fast in recent years that execution time is becoming less relevant. Hence, software such as MATLAB and Excel/VBA that can shorten the development cycle have become more important.
Scripting Languages

Development versus execution time.

Figure: Development versus execution time.
Other things that are desirable but may slow down either execution or development time include the following.

1. extensive error checking (will slow both development and execution time)

2. putting in extensive documentation (will slow development time, but have no effect on execution time)

3. Using object oriented programming, making extensive uses of classes and polymorphism (will speed development time but slow execution time).
On another note:

A person in the software industry recently remarked to me that she thought that not all of the decline in jobs for “real programmers” was due to overseas outsourcing. Rather, she felt that because of the proliferation of easy to scripting languages many managers were actually doing programming and they didn’t need real programmers to deal with third generation languages.
Tradeoffs – Why VBA?

Excel competitors:

- MATLAB, Mathematica
- SAS, R
- Python, Perl

Among the scripting languages, why use VBA?

Think about Willie Sutton for an analogy?
Tradeoffs – Why VBA?

VBA is both powerful and flexible.

- Build models easily – take input parameters and generate a model instance.
- Provide a user interface/GUI so the user doesn’t have to do a lot of work.
- Automate tasks through Macros.
- Connect to other programs/applications, e.g. a SQL Server or Access database.
- Excel with VBA is an IDE – Integrated Development Environment
Tradeoffs – Why VBA?

Excel with VBA is an IDE. Key features include:

- There is a VBA editor. This is used for entering VB code.
- There is a generic “UserForm” for creating user interfaces.
- There a huge function library to select from. You can also write your own functions using VBA.
- Excel can be used for statistical analysis. There are routines for calculating means, variances, correlations, running regressions, forecasting models, etc.
• You can use Excel for mathematical optimization. We will discuss the GAMS package later in the quarter.

• You can use Excel for simulation.

• Excel is excellent for creating graphs and histograms.

• Excel can import and export data in various formats. It can even read and write XML.

• Excel can serve as a database. You can filter and query the data.

• You can interface with other programs using Web services.
Tradeoffs – Why VBA?

VBA, MATLAB, Python – does it really matter which one you learn?

No!!! You will learn concepts.

I am a former Booth student, and took your VBA/MAtlab class last winter. I just had a job interview that requires the use of Python ...

it didn't sound like the tasks they required were that extensive....

Is Python that different from anything we used in class?

Thanks in advance for any help you can give....

And if I hadn’t taken your class I wouldn’t even be in the running so thanks for offering it!
Brief History

The first spreadsheet program for the PC was VisiCal. Created by Dan Bricklin (Harvard MBA student) and his buddy Bob Frankston in 1978. Product was sold and marketed by Dan Fylstra’s (another Harvard MBA) company VisiCorp. It was developed for the Apple II. First “killer app” for the PC.

Lotus 1-2-3 was the first big commercial success.

- a spreadsheet
- a graphics package
- a database

Microsoft Excel is now the major player in this market.
Microsoft’s first product was a very elementary Basic compiler.

Then Microsoft created a widely successful product called Visual Basic.

Then Microsoft, in an extremely rare and unusual moment of innovative clarity, combined Visual Basic with Microsoft Office resulting in VBA (Visual Basic for Applications).
VB Flavors

- VB now VB .NET – a programming language that can be compiled into an application. Part of the Microsoft Visual Studio IDE

- VBA – a scripting language that uses one of the Office products, e.g. Excel, Word, Power Point, Access as the host. We will use VBA for Excel.

- VBScript – the browser, IE Explorer, is the host. Designed to compete with the Netscape Java Script.
The VB Editor
The VB Editor

In order to access the VB editor (Excel 2007)

► Click on the **Developer Tab** in Excel ribbon

► Click on **Visual Basic** to bring up the Editor.

If you do not see the Developer Tap click on the **Office Button** in the upper left-hand corner. Then select **Excel Options** at the bottom of the window. Make sure ”Show Developer tab in the Ribbon” is selected.
The VB Editor

Bringing up the Developer Tab
The VB Editor

- Code Window
- Properties Window
- Project Explorer
The VB Editor

The VB Editor is a set of Windows. You can get a list of available Windows by selecting the View menu item in the VB editor.

- The **Project Explorer**: sort of like the Windows Explorer and allows to explore the files structure of the files in your project.
  
  - Projects
  
  - Objects – the top level objects such as the Workbook and Worksheets
  
  - User Forms – GUIs
  
  - Modules – chunks of code that can be executed across worksheets, not necessarily specific to a worksheet.
The VB Editor

- The **Code Window**: this is a text editor. You type VBA code in the Code Window. You will spend a lot of time in this Window.

- The **Properties Window**: This will be used later, when we build GUIs. Ignore it for now (although nice if you want to change a module name).
The VB Editor

Components of the VB Editor (continued)

▶ The **Immediate Window** Where you go for instant gratification.

\[ Y = \frac{4}{5} \]

?Y
0.8

?Worksheets("Sheet1").Range("amatrix").address
$C$5:$D$8

\[ Y = \frac{2}{3} \]

?Y
0.6666666666666666

Also used for debugging as we see later.

▶ The **Watch Window** Not now – later, this is good for debugging
The VB Editor

The VBA editor also contains a set of **Toolbars**. Again, go to the menu item **View** and select **Toolbars**.

- Standard Toolbar (play around with the various options)
- Edit Toolbar (play around with the various options)
- Debug Toolbar (later)
- UserForm (later)

Go over Chapter 3 in the text several times.
Object Oriented Programming

Modern programming languages such as C++ and Java are object oriented. Even scripting languages such as VBA have object oriented features.

**KEY IDEA:** develop reusable software modules called *objects* and make it easy to modify or extend the objects for use in different applications.

Object-oriented programming is built around the concepts of a *class* and an *object*. A class is an abstraction of a real-world object and describes the essence of the object. For example the concept of a worksheet is a class. A specific worksheet with data is an object in the class.

A class is usually composed of 1) *properties* (often called primitives or data members) and 2) *methods* (often called functions) that operate on the data. In object speak we say the properties and methods are encapsulated into objects.
Object Oriented Programming

Each of the Microsoft Office family members (Excel, Access, etc.) support VBA.

The key difference between the various flavors of VBA is the object model.

The Excel VBA object model naturally focuses on the concepts of a Workbook, Worksheet, and Range.

What might be some reasonable key objects in Word VBA? For example, kind of object might correspond to a Workbook?
Object Oriented Programming

The two **KEY IDEAS** for the first part of this course:

- The concept of an object
- How to work with the range object (next week)
A fantastic learning aid is the **object browser**. Spend time with this! The object browser is useful for rookies and experienced veterans alike.
Object Oriented Programming

In the left-most **Classes** window scroll down and observe the various classes.

Select the **Worksheet** class.

In the right window you will see *Members of Worksheet*. Notice the three different icon types each denoting a different type of member

- **Properties Member** – icons with a hand
- **Sub Member** – green icon
- **Event Member** – lighting bolt icon
Select the **Worksheet** class. The Range member is a property of the Worksheet class. The Range is also a class. A class can have a member property that is also a class. Neat!
Object Oriented Programming

Key Idea: Class hierarchy

The **Workbook** class has **Worksheet** class members.

The **Worksheet** class has **Range** class members.

The **Range** class has **Font** class members.

The **Font** class has **Size** data member (an Variant).
Object Oriented Programming

The SaveAs member is a sub (I like to call these methods) of the Worksheet class.

![Object Browser](image)

![Object Model](image)
The SaveAs method has **arguments**. The arguments tell the method how to act. Some arguments are required some are optional [].

Sub SaveAs(Filename As String,
[FileFormat],
[Password],
[WriteResPassword],
[ReadOnlyRecommended],
[CreateBackup],
[AddToMru],
[TextCodepage],
[TextVisualLayout],
[Local])
Object Oriented Programming

The following code illustrates the object oriented nature of VBA.

Sub DemoObject()
    Dim ws As Worksheet
    Dim rng As Range
    Set ws = Worksheets("DemoObject")
    'A method with no arguments
    ws.Activate
    ws.Range("A1") = 99
    Set rng = ws.Range("A1")
    rng.Font.Italic = True
    'Set the value of a property
    ws.Range("A1").Font.Size = 22
    'Methods with arguments
    rng.Copy Destination:=Range("G1")
End Sub
Object Oriented Programming: “Dot Notation”

**Member Specification:** The notation to specify a member of an object is

**Object.Member**

For example, `rng.Font`

or, `rng.Calculate`

A class member can be a class that has other members. For example: `rng.Font.Italic`

To set the value of a class member that is a property do the following

`rng.Font.Italic = True`
Object Oriented Programming: Method Specification

**Method Specification:** The notation to specify a method of an object for methods with no arguments is

`Object.Method`

For example, `ws.Activate`

If the method has arguments the notation is

`Object.Method Arg1:=Arg1Value, Arg2:=Arg2Value, ..`

For example: `rng.Copy Destination:=Range("G1")`
Object Oriented Programming: Method Specification

Another way to specify is to drop the argument name, i.e.

Object.Method Arg1Value, Arg2Value, ..

rng.Copy Range("G1")

or, when used in an assignment parenthesis

Dim money as Currency
money = VBA.Format(34.124545, "Currency")
Intellisense: wow is this great, or what!

In the code window type in:

Class (object) and all of the members associated the class (object) will pop up in alphabetical order

If you type in the name of the method, Intellisense will list the arguments needed, e.g. type

MsgBox and hit the space, the arguments to MsgBox will pop up.
Object Oriented Programming

KEY CONCEPTS:

▶ Class

▶ Object

▶ Class Members – the members *define* the class (properties and methods/procedures)
  
  ▶ Method (also called subroutines, procedures, or functions)
  
  ▶ Property – data describing the class

▶ Arguments – inputs to a method/function
Generating Code

There are three basic ways to generate code:

▶ Import code from a file

▶ Enter code by hand into the code window

▶ Create a macro
Import Code

Open an Excel file that has VBA code and select the Visual Basic Icon in the Developer tab. You will see the code in in the VB editor.

Important – Under Macro Security you will need to check the box for Trust Access to the VBA project model.
Generating Code

There are two ways to generate new code. Either type the code into the Code Window by hand or record a Macro. For now we focus on recording macros.

Macro – a recording of key strokes and mouse clicks
Macros

After clicking on the record button, the figure below appears. Name the macro and click OK to start recording. When you are done recording your mouse clicks, click **Stop Recording** icon.
Macros

We can also make a macro available to all workbooks.
Macros

- Download the file `vblIntro.xls` and select the `copyMacro` worksheet.
- Turn on Macro Recording
- Name the macro `copyMacro`
- Highlight the data for ages 30 through 40
- Take the “cross” in the lower left hand side and copy through the age of 60.
- Copy `cntrl-C`
- Select Cell D2
- Paste `cntrl-V`
- Select cell A1
- Stop Recording
Now go into the VB editor and examine the code. You should have:

Sub copyMacro()
    Range("A2:A12").Select
    Selection.AutoFill Destination:=Range("A2:A32"), Type:=xlFillDefault
    Range("A2:A32").Select
    Selection.Copy
    Range("D2").Select
    ActiveSheet.Paste
    Range("A1").Select
End Sub
Macros

1. What is **Range**? (Object, method, property)

2. What is **Select**? (Object, method, property)

3. What is **Selection**? (Object, method, property)

4. What is **AutoFill**? (Object, method, property)

5. What are the arguments for **AutoFill**?

6. What are the arguments for **Paste**
Macros

- A macro corresponds to code in a module. In particular the macro is a procedure.

- The default Module is Module1 – you can change this name in the Properties Window.

- A module is code “seen” by all of the spreadsheets.

- The name of the procedure corresponds to the name of the macro.

- Comments are preceded in VB by a single quote.

- We can continue code from one line to the next using an _ (underscore)
A sub procedure has the format:

Sub SubName()
    ... VBA Code ...
End Sub
Download the file **vbIntro.xls** and select the **flightMacro** worksheet.

First understand filtering and sorting.

Highlight the entire flight data table.

Make sure the **Home** tab is selected. Then in the ribbon Select Data>Filter.

Play around and do some sorting and filtering.

Now deselect Filter so we are back to the original worksheet.
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Destination</td>
<td>Stops</td>
<td>Class</td>
<td>Price</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Hong Kong</td>
<td>2</td>
<td>Business</td>
<td>$2,500</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Hong Kong</td>
<td>3</td>
<td>Business</td>
<td>$2,200</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Hong Kong</td>
<td>3</td>
<td>Business</td>
<td>$2,100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Beijing</td>
<td>2</td>
<td>Business</td>
<td>$2,000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Beijing</td>
<td>3</td>
<td>Business</td>
<td>$1,900</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Hong Kong</td>
<td>3</td>
<td>Economy</td>
<td>$1,800</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Hong Kong</td>
<td>4</td>
<td>Business</td>
<td>$1,750</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Hong Kong</td>
<td>4</td>
<td>Economy</td>
<td>$1,400</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Beijing</td>
<td>3</td>
<td>Economy</td>
<td>$1,300</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Hong Kong</td>
<td>5</td>
<td>Economy</td>
<td>$1,100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Beijing</td>
<td>4</td>
<td>Economy</td>
<td>$900</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Beijing</td>
<td>5</td>
<td>Economy</td>
<td>$700</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Macros

GENERATING FILTERING MACROS

▶ **Step 1**: Deselect Filter so we are back to the original worksheet.

▶ **Step 2**: Start recording a macro named ViewHongKongFlights

▶ **Step 3**: Highlight the range A3:D15.

▶ **Step 4**: Select Data>Filter

▶ **Step 5**: In the Destination column select Hong Kong

▶ **Step 6**: Select cell A1

▶ **Step 7**: Stop Recording
GENERATING FILTERING MACROS

Your code should look like:

Sub ViewHongKongFlights()
    Range("A3:D15").Select
    Selection.AutoFilter
    Selection.AutoFilter Field:=1, Criteria1:="Hong Kong"
    Range("A1").Select
End Sub

Edit the ViewHongKongFlights() Sub Procedure so that Beijing flights are viewed instead of Hong Kong flights.
Do the following:

- Based on your understanding of VB create a procedure (without recording a macro) called ViewEconomy that will show all of the available Economy flights.

- Test and run these macros
Events: actions that take place in the Excel window that are linked to code (often called handlers).

- Open a workbook
- Click on a command button
- Activate a worksheet

An event procedure is a procedure except that it is associated with an excel event.
## Events

![Excel Spreadsheet](image)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td><strong>Destination</strong></td>
<td><strong>Stops</strong></td>
</tr>
<tr>
<td>4</td>
<td>Hong Kong</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>Hong Kong</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>Hong Kong</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>Beijing</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>Beijing</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>Hong Kong</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>Hong Kong</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>Hong Kong</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>Beijing</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>Hong Kong</td>
<td>5</td>
</tr>
<tr>
<td>14</td>
<td>Beijing</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>Beijing</td>
<td>5</td>
</tr>
</tbody>
</table>
Events

Event Example: the Command Button object.

See Developer and Insert

Click on the Command Button object and then click somewhere on the spreadsheet.

Click on New

Create the following code:

Sub Button1_Click()
    MsgBox "Hello World"
End Sub
**Review:** An object is made up of?