Introduction to Scripting Languages

Course Mechanics

End User Programming and VBA

What is Scripting

Practice

- Rapid prototyping (Perl, Tcl, Python)
- End-user programming (shells, macros, config files, spreadsheets)

Domain

- Text processing (awk, Perl)
- Application extension (VBA, AppleScript)
- Web (JavaScript, PHP, Ruby)

Scripting Language

Language whose features favor scripting

- Low boiler-plate
  `print "Ice cream!\n"`
- Dynamic typing
  `$amount = 20 . " grams";`
- Interpretation
  `eval "print 'egg'\n";`
- String manipulation
  `$x = "food"; $x =~ s/o/e/g;`
- Associative arrays
  `$group{pasta} = 'carbs';`
- Properties
  `document.im1.src="meal.jpg";`
- Call-backs
  `<input ... onClick="stir()">`

Why Scripting

- Simplicity
- Productivity
- Domain

Language Popularity Index


http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html
Introduction to Scripting Languages  
Course Mechanics  
End User Programming and VBA
Course Goals

• **Short-term**
  – Have skills in Perl, VBA, PHP, JavaScript
  – Understand scripting PL concepts

• **Long-term**
  – Use languages effectively
  – Quickly learn new languages on your own
  – Design or improve scripting languages

Grading

• 25% hw + 35% quizzes + 40% final
• Homeworks
  – Due each Friday at 6pm
  – >= 1 minute late: 50% points
  – >= 3 hours late: 0% points
  – You can collaborate, but must indicate so

Recommended Books

About VBA

- Visual Basic for Applications
  - Visual = create GUI by drag’n’drop
  - Basic = simple, for end users
  - for Applications = embedded in Word, Excel, PowerPoint, AutoCAD, Corel Draw, Acrobat
- Object-oriented, event-driven
- Aimed at end user programming
  - Automate task user does by hand otherwise
  - Same niche as Emacs Lisp, AppleScript

Related Languages

- VB6
  - Same language as VBA, different libraries
  - “End of life”, not supported on new Microsoft platforms
- VBScript
  - Subset of VB 6.0, for server-side scripts
- VB.NET
  - Not backward compatible, based on CLR

How to Write + Run Code

- Open Word, PowerPoint, or Excel
- Visual Basic Editor (Alt-F11)
  - Immediate window (edit-eval-print loop)
  - Code editor, including auto-completion
  - Debugger, object browser
  - Help system
- Macro Recorder
  - Recorded macros are useful code blueprint
- Customization
  - Add hand-written code to toolbars or menus

Lexical Peculiarities

- Case insensitive
- Line break sensitive
  - One statement, multiple lines: _
  - One line, multiple statements: :
  - Single-line comments: ' , Rem
- Literals
  - Boolean: True, False
  - Date: #April 20, 2008#, #10:15pm#
  - Variant: Empty, Error, Nothing, Null
  - Character: vbCr, vbTab, vbLf, Chr(149), ...

Types

- Variant
- String
- Integer
- Long
- Decimal
- Currency
- Single
- Double
- Enum

Variable Declarations

<table>
<thead>
<tr>
<th>Implicit, untyped</th>
<th>fruit = &quot;apple&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implicit, typed</td>
<td>fruit$ = &quot;apple&quot;</td>
</tr>
<tr>
<td>Explicit, untyped</td>
<td>Dim fruit</td>
</tr>
<tr>
<td>Explicit, typed</td>
<td>Dim f As String</td>
</tr>
<tr>
<td>Constant</td>
<td>Dim Const pi As Byte = 3</td>
</tr>
<tr>
<td>Constant</td>
<td>Dim x As Integer</td>
</tr>
<tr>
<td>Constant</td>
<td>Dim a(10)</td>
</tr>
<tr>
<td>Constant</td>
<td>Dim x(5 To 10)</td>
</tr>
<tr>
<td>Constant</td>
<td>Dim a(10, 5)</td>
</tr>
<tr>
<td>Array</td>
<td>Optional As type</td>
</tr>
</tbody>
</table>

© Martin Hirzel
Concepts

Gradual and Dynamic Typing

- **Gradual typing**: optional in declaration
- If not specified: **Variant**
  - **Dynamic typing**: check at the last moment whether runtime value legal for operation
  - E.g., same variable can hold an `Integer` at one time, and a `String` at another
- If specified:
  - **Static typing**: check at compile time
    - Compiler can optimize time+space

Arrays

- Indexing with round parentheses `()`
- Statements
  - `ReDim Preserve id ([new sizes/bounds])`
  - `Erase id`
- Functions
  - `IsArray(id)`
  - `LBound(id[, dim])`, `UBound(id[, dim])`
- Options
  - "Option Base 1" overrides default base-0 indexing

Concepts

BNF = Backus Naur Form

<table>
<thead>
<tr>
<th>Construct</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal</td>
<td>concrete code</td>
<td>Courier Bold</td>
</tr>
<tr>
<td>Non-terminal</td>
<td>placeholder</td>
<td>Times Italic</td>
</tr>
<tr>
<td>Ellipsis</td>
<td>omitted code</td>
<td>...</td>
</tr>
<tr>
<td>Rule</td>
<td>non-terminal definition</td>
<td>Lhs ::= Rhs</td>
</tr>
<tr>
<td>Alternative</td>
<td>choose one</td>
<td>A1</td>
</tr>
<tr>
<td>Optional</td>
<td>zero or one times</td>
<td>[Square brackets]</td>
</tr>
<tr>
<td>Repeat</td>
<td>zero or more times</td>
<td>Kleene star*</td>
</tr>
<tr>
<td>Grouping</td>
<td>treat as unit</td>
<td>(parentheses)</td>
</tr>
</tbody>
</table>

Input and Output

- `Debug.Print "Hello, world!"`
- `Application.StatusBar = "Hello, world!"`
- `MsgBox "Hello, world!"`
  - `MsgBox(prompt[, buttons[, title]][, default][, xpos][, ypos]) As Long`
  - Buttons: 0 `vbOkOnly`, 1 `vbOkCancel`, 2 `vbAbortRetryIgnore`, 3 `vbYesNoCancel`, 4 `vbYesNo`, 5 `vbRetryCancel`
  - Result: 1 `vbOk`, 2 `vbCancel`, 3 `vbAbort`, 4 `vbRetry`, 5 `vbIgnore`, 6 `vbYes`, 7 `vbNo`
- `userName = InputBox("Who are you?")`
  - `InputBox(prompt[, title][, default][, xpos][, ypos]) As String`

Operator Characterization

<table>
<thead>
<tr>
<th>Operator</th>
<th>Arity</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>()</code></td>
<td>0</td>
<td>left</td>
<td>indexing</td>
</tr>
<tr>
<td><code>-</code></td>
<td>1</td>
<td>unary</td>
<td><code>-</code></td>
</tr>
<tr>
<td><code>+</code></td>
<td>1</td>
<td>binary</td>
<td><code>+</code></td>
</tr>
<tr>
<td><code>*</code></td>
<td>2</td>
<td>binary</td>
<td><code>*</code></td>
</tr>
<tr>
<td><code>/</code></td>
<td>2</td>
<td>division</td>
<td><code>/</code></td>
</tr>
<tr>
<td><code>&lt;</code>, <code>&gt;</code></td>
<td>1</td>
<td>relational</td>
<td><code>&lt;</code>, <code>&gt;</code></td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>1</td>
<td>logical</td>
<td><code>&lt;</code></td>
</tr>
<tr>
<td><code>&lt;=</code>, <code>&lt;=</code></td>
<td>2</td>
<td>relational</td>
<td><code>&lt;=</code>, <code>&lt;=</code></td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>1</td>
<td>logical</td>
<td><code>&gt;=</code></td>
</tr>
<tr>
<td><code>&amp;</code></td>
<td>2</td>
<td>logical</td>
<td><code>&amp;</code></td>
</tr>
<tr>
<td>`</td>
<td>`</td>
<td>2</td>
<td>`</td>
</tr>
<tr>
<td><code>&lt;</code>, <code>&gt;</code></td>
<td>1</td>
<td>relational</td>
<td><code>&lt;</code>, <code>&gt;</code></td>
</tr>
</tbody>
</table>

Arity, Precedence, Associativity

<table>
<thead>
<tr>
<th>Arity</th>
<th>Number of operands</th>
<th>Precedence</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Precedence and associativity in programming usually follows the conventions from math.
Operators

- **Exponentiation** (^)
- **Identity, negation** (+, -)
- **Multiplicative** (*, /)
- **Integer division** (\)
- **Modulus** (Mod)
- **Additive; String concatenation** (+, &)
- **Bit shift** (<<, >>)
- **Comparison** (=, <>, <, >, Is)
- **Negation** (Not)
- **Logic** (And, Or, Xor, Eqv, Imp)
- **Assignment statement** ([Set] … = …)

Control Statements

- **Conditional** (If … Then … ElseIf … Else … End If)
- **Select Case**
- **Fixed/iteration loops**
  - For … To … Step … Next …
  - For Each … In … Next …
- **Indefinite loops**
  - Do … Loop While …
  - While … Loop
  - Do Until … Loop
  - Do … Loop Until …
- **GoTo**
- **Exit Do**
- **Exit For**
- ** Exit Function**
- **On Error GoTo**
- **Err.Raise**
- **Resume**

Writing Subroutines

- **Declaration**
  - mods Sub id [(arg*)] … End Sub
  - mods Function id [(arg*)] [As type] … End Function
  - To return a value, assign to function name, else default
- **Function modifiers**
  - Public, Private: visibility outside module
  - Static: make all locals static
- **Arguments**
  - mods id [(arg*)] [As type] = expr
- **Argument modifiers**
  - Optional: after first, rest must also be Optional
  - ByRef, ByVal: default ByRef
  - ParamArray: must be last, allows for var-args

Library Functions

- **Simple data conversion**: CBool, CByte, CCur, CDate, CDbl, CInt, CLng, CStr, CVar
- **Complex data conversion**: Asc, Chr, Format, Hex, Oct, RGB, QBColor, Str, Val
- **String**: InStr, InStrRev, LCase, UCase, Left, Len, LTrim, RTrim, Mid, Right, Space, StrComp, StrConv, StrReverse, Trim
- **Math**: Abs, Atn, Cos, Exp, Fix, Log, Rnd, Sgn, Sin, Sqr, Tan, IsNumeric
- And many more

Example, Revisited

```vba
Option Explicit
Sub LemonStar()
    Dim S As PowerPoint.Slide
    Set S = ActivePresentation.Slides(1 - ActivePresentation.Slides.Count)
    Dim I As Integer
    For I = 0 To 8
        Dim L As PowerPoint.Shape
        Const Dpi As Integer = 72 ' 72 dots per inch
        Set L = S.Shapes.AddLine(BeginX:=Dpi*5, BeginY:=Dpi*3.75+I*Dpi/8, EndX :=Dpi*6, EndY  :=Dpi*4.75-I*Dpi/8)
        L.Line.ForeColor.RGB = RGB(I * 31, I * 31, 0)
    Next I
End Sub
```
Reference

VBA Documentation

- Included:
  - Macro recorder
  - Auto-completion
  - Help system
  - Object browser
- Online: MSDN Library → Development Tools + Languages → Visual Studio 6.0 → VB 6.0

Soap-box

Evaluating VBA

**Strengths**
- Development environment
- Simplicity
- Availability
- Popularity
- Best tool for end users of MS Office

**Weaknesses**
- Single platform/vendor
- Binary format
- Security
- Missing features
  - Structured exceptions
  - Implementation inheritance
  - Regular expressions
  - Associative arrays

Administrative

Last Slide

- Please register
- Please subscribe to mailing list
- First homework is due on Friday 1 June
- Today’s lecture
  - Introduction to scripting
  - Dynamic typing
  - Precedence and associativity
  - Basics of VBA
- Next lecture
  - Putting the V and the A in VBA
  - Application extension
  - Properties
  - Call-backs