



## Week 2:

- Problem solving with conditional constructs
  - Solution formulation with linear equations
- Use of HTML for web page interface
- ActiveX Services
  - Excel.Application – spread sheet application

---

---

---

---

---

---

---

---



## Conditional Constructs

- Positive Logic: test for “positive condition”

```

bool lamLate = TrafficOn405()    ' TrafficOn405() is
    a function
    ' it returns "true" or "false"

If (lamLate) Then
    Walk in quietly
Else
    Sing as I walk in, greeting my friends
EndIf
  
```

---

---

---

---

---

---

---

---



## Nested Conditional construct

- There is another set of conditions: either my friend is in class or she is not.
- Test on this condition only if it affects our action

---

---

---

---

---

---

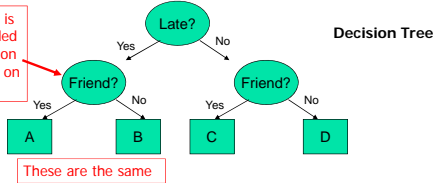
---

---

## More complicated: Is my friend in class?

I am Late	Best friend in class	A: Go in quietly
	Best friend not there	B: Go in quietly
I am not late	Best friend in class	C: high five her
	Best friend not in class	D: sing to self, and greet others

This test is not needed - no action depends on it.




---

---

---

---

---

---

---

---

---

---

## Nested IF statements

```

bool IamLate = TrafficOn405() ' TrafficOn405() is
a function

If (IamLate) Then
  Walk in quietly
Else
  If (MyBestFriendIsInClass) Then
    HighFiveWithHer as I walk in
  Else
    Sing as I walk in, greeting other friends
  EndIf
EndIf
  
```

If I am not late for class, there are now two cases. Choice depends on whether my friend is there.

---

---

---

---

---

---

---

---

---

---

## Default action ...

- A more introverted but courteous student will neither sing nor high five people. But, if late, will apologize for entering late.
 

```

bool IamLate = TrafficOn405() ' TrafficOn405()
is a function
If (IamLate) Then
  Apologize to the class
EndIf
Walk in quietly ' something we will do in any
case
      
```
- No need for "else" in this case

---

---

---

---

---

---

---

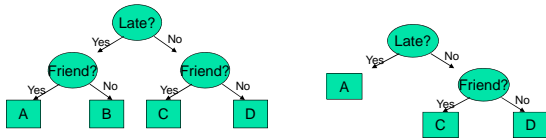
---

---

---

## Take away lesson: Conditions

- ALWAYS: draw the decision tree!
- Every condition at the *bottom* should be taken care of!



---

---

---

---

---

---

---

---

## Flow Charts for IF tests ...

---

---

---

---

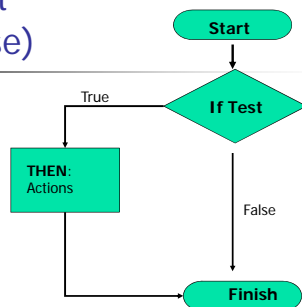
---

---

---

---

## IF test (no else)



---

---

---

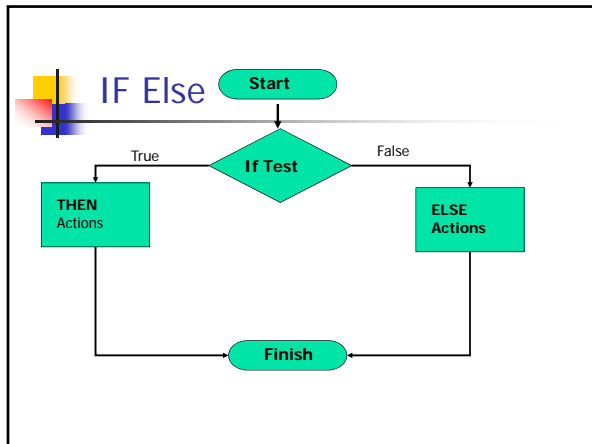
---

---

---

---

---




---

---

---

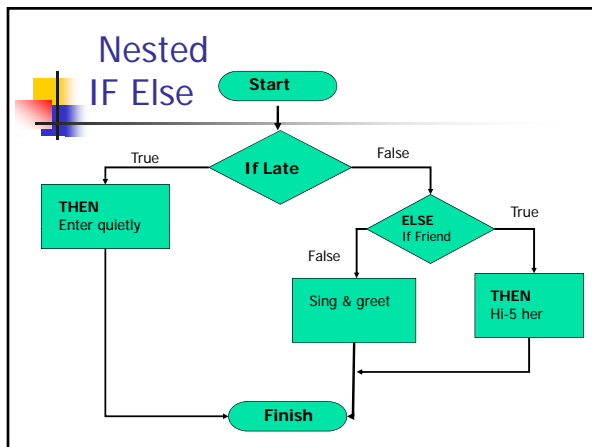
---

---

---

---

---




---

---

---

---

---

---

---

---

### With Partner

- Learning Activity #2

---

---

---

---

---

---

---

---



## Conditional Constructs

- OR (reversed, but still positive logic)

```

bool IamNotLate = NoTrafficOn405() '
NoTrafficOn405() is a function

If (IamNotLate) Then
  Sing as I walk in, greeting friends
Else
  Walk in quietly
EndIf

```

---

---

---

---

---

---

---

---



## Negative Logic (more awkward to me)

- Negative Logic:

```

bool IamLate = TrafficOn405() ' TrafficOn405()
is a function

If (Not IamLate) Then
  Sing as I walk in, greeting friends
Else
  Walk in quietly
EndIf

```

Here, Not is a logical operator

---

---

---

---

---

---

---

---



## Find the hundreds digit

- Look at *HundrethDigit.vbs*
- Easy to work with numbers ...
  - $43 = 4 * 10 + 3 * 1$
  - $543 = 5 * 100 + 4 * 10 + 3 * 1$
- Integer division vs. float division
  - Ordinary division (/) is floating only
- What is the value of  $543/100$ ?
- What is the value of  $553/100$ ?
- How can you use `CInt ()` to extract the "5"?

---

---

---

---

---

---

---

---



## Now you have the "5"...

- How to get the "43" from "543"?
  - You already have the "5"
  - and then what?
- Calculating a "remainder"
- Assumes:
  - We do not have integer division available
  - The algorithm is correct in all cases.

---

---

---

---

---

---

---

---



## Big Idea?

- Look for relationships
  - Our algorithm produces 5 and 43 from 543  
*what are the relationships here?*
  - What about producing 4 and 3 from 43 ?

---

---

---

---

---

---

---

---



## Big Idea?

- Use the relationships to devise an algorithm for a more general case.
- For a 5-digit number (e.g. 57324) find each digit (10G, thousands, hundreds, etc.)
- Look at *FiveDigit.vbs*
- **Activity 3 (tests and revision)**

---

---

---

---

---

---

---

---



## HTML (FirstHTML)

- Use html for Input/output
- A set of formatting tags
  - <Tag> and </Tag>
  - <!-- is comment
- There are many tags – look them up
- Web browser (IE, or Firefox) *interprets* HTML to output formatted text.
  - Look at *FirstHTMLExample.html*
  - Look at *LargestNum.html*




---

---

---

---

---

---

---

---



## <SCRIPT> Tag: (LargestNum)

`<SCRIPT Language="VBSCRIPT">`

- Language="" is option to the tag
- </SCRIPT> is the end of tag
- Let's ignore everything in between the <SCRIPT> tags for now

---

---

---

---

---

---

---

---



## Other tags

- the HTML programming help page (course web-site again)
- <input> tag: an input *intrinsic control object* (*Lomax, pp 170ff*)
  - Type, Name, Value
  - Type: button, text, checkbox, ...
  - Name: give the "input element" a name
  - Value: the content of the input element (*is set by user*)
- HTML Programming: Reference **B**:
  - Input is under (*HTML element*)
  - Button is an object with event (under Objects->objects)

*Note: the inputs are outside the /SCRIPT tags  
 Nothing happens when we fill in the boxes.  
 Until – we click "submit"*

---

---

---

---

---

---

---

---



## Servicing Events from tags

- Sub *SOMETHING\_OnClick* (e.g.: `submitButton_OnClick`)
  - When we click on the "SubmitButton" the `_OnClick` Sub is called
- We use
  - `Num1.value`: what is this?
- Reference:
  - A->VBScript User's Guide
    - -> VBScript in Internet Explorer
  - Chapter 8 in Lomax

---

---

---

---

---

---

---

---



## More about MsgBox

- `msgBox arg1, arg2, arg3`
  - Arg1: "string" with & concatenation
  - Arg2: `vbOKOnly`
  - Arg3: "another string"
- What is going on?
- How to find out more?
- A:Language Reference->MsgBox or Lomax p. 356

---

---

---

---

---

---

---

---



## LargestNumberWithFormat.html

- `LargestNumberWithFormat.html`
- Nicer looking input, no functionality difference!
- Tags
  - `<Table>` - begin of table
  - `<TR>` - a row in table
  - `<TD>` a data element of a row
- Notice the *indentation* of the source file!!

---

---

---

---

---

---

---

---

## Debugging Issues

### ■ Activity 4

---

---

---

---

---

---

---

---

## Income Tax Problem

- **Problem:** Calculate Income Tax given income.
- **Keystone City has three tax brackets**
  - 2% on income \$1000 or less
  - 10% on income in range \$1001-2000
  - 25% on income in range \$2001 -
- **Mathematical Model:**
  - Tax has three linear ranges ( $y = mx + c$ )
  - Tax can be calculated for each range and added
  - Test understanding by working out by hand
- **Programming Approach:**
  - Write vbscript within html page that utilizes this model.
  - *Standard approach to problems.*

---

---

---

---

---

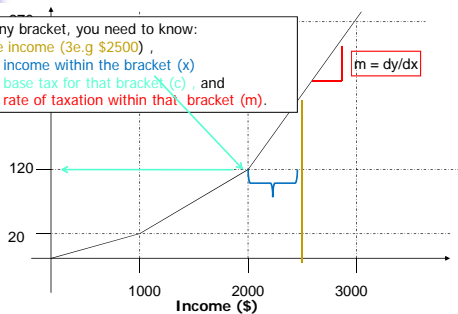
---

---

---

## Solution Modeling: Linear

- In any bracket, you need to know:
- the income (3e.g \$2500)
  - the income within the bracket ( $x$ )
  - the base tax for that bracket ( $c$ ) and
  - the rate of taxation within that bracket ( $m$ ).



---

---

---

---

---

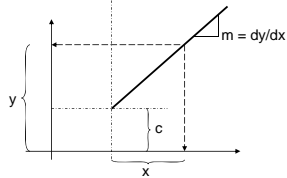
---

---

---

## In English ...

- $y = mx + c$
- Total Tax = sum of the following two
  - Tax Base from previous bracket (c)
  - Amount in this bracket (x) \* tax rate (m)



---

---

---

---

---

---

---

---

## With partner – test by hand

Brackets	\$0-1000	\$1001-2000	\$2001-
Rates	2%	10%	25%

- Calculate (by hand):
  - Total tax for an income of \$600
  - Total tax for an income of \$1,400
  - Total tax for an income of \$3300
- Answers:
  - \$12
  - \$60
  - \$445

---

---

---

---

---

---

---

---

## HTML + Script + Conditional Construct

- `Incometax.html` example ...
- Run the same data you just did by hand.

---

---

---

---

---

---

---

---



## HTML + Script + Conditional Construct

- **Incometax.html** example ...
- **WATCH OUT!!** When embed scripts in html ...
  - Some errors just cause it not to run when the errors are before the event routine
- **Conditional Construct:** nested if-then-else
  - Natural to determine correct tax bracket and rate
  - Proper indentation
  - Make sure all cases are handled
  - Make sure no unnecessary cases
- **Const:** never use numerals ...
  - All constants numbers and constant strings should be defined before used in program code. *Lomax p. 218*

---

---

---

---

---

---

---

---



## Compute Rate: Commission.html

- Keystone Daily pays commission based on total sales in three brackets:
  - For \$5000 in sales, commission = \$100
  - For \$6500 in sales, commission = \$300
  - For \$10,000 sales, commission = \$800
  - Over \$10,000 sales -> fraud assumed
  - Within these brackets, commission will be *proportional* to the amounts above.

---

---

---

---

---

---

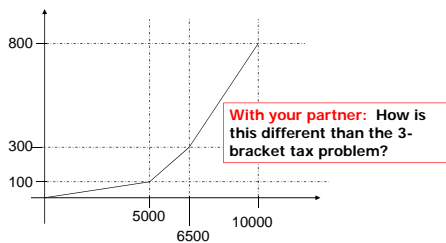
---

---



## Compute Rate: Commission.html

- **Proportion:** what does this mean?




---

---

---

---

---

---

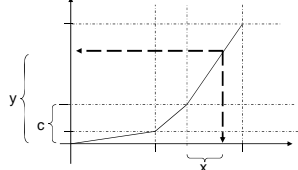
---

---



## What do we need?

■  $y = mx + c$



- What is **m**? (*tax rate previously –still a rate*)

---

---

---

---

---

---

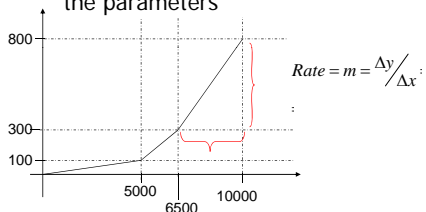
---

---



## Rate from graph

- **Rate:** express in terms of the parameters



---

---

---

---

---

---

---

---



## Why are we doing this?

- Solution derivation based on mathematical models
  - Standard approach
  - See how its implemented.
- Note how it is similar to previously solved problem
- Point? ...
  - In many cases ... it's not that difficult!
  - See *Commission.html*

---

---

---

---

---

---

---

---



## MP1: due Oct 17 (Sat)

- We now have what we need to solve MP1.
- Start on it right away.
- Design the solution before starting to code.
- Be sure to leave time for testing and revision

---

---

---

---

---

---

---

---



## Spreadsheets – why here?

- Easy to use – do rather complicated calculations and manipulations for us.
- Using VBScript, we can automate this process, perhaps applying a spreadsheet function to multiple data sets without intervention – much easier than programming these complicated activities.

---

---

---

---

---

---

---

---



## Vbs code:

```
Dim xlApp!  
Set xlApp! = CreateObject("Excel.Application")
```

Page 220-21 of Lomax gives details and list of some programmatic identifiers.

---

---

---

---

---

---

---

---



## OpenExcel.vbs

Ref: pp 220-23 of Lomax

- **CreateObject()** – function call
  - Activate “Active X” services (Excel)
- **Set** – VB script reserve word
  - Only use with “Objects”
  - Do not use for basic data type
- Excel.Application
  - Registered (with the WindowsXP)
  - ActiveX service

---

---

---

---

---

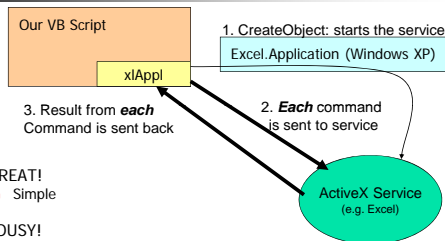
---

---

---



## Execution Model



- GREAT!
  - Simple
- LOUSY!
  - Efficiency (each command is a round trip)
  - Almost no error handling!
    - Command are sent into the unknown!

---

---

---

---

---

---

---

---



## Classes and Services of Excel

- From OpenExcel.vbs
  - xlAppl is the handle to the activeX service
  - Application object (xlAppl.Application.something)
  - Refer to manual: [D](#) to find out more!
- Strategies on learning to use the interface
  - Find simple VBScripts and understand how they work
  - Open MS Excel: View->Macros:
    - Microsoft Visual basic Editor

---

---

---

---

---

---

---

---



## How to USE an ActiveX

- From our program (our VBScript)
  - Create a handle to represent the service
  - The CreateObject() function call
  
- Based on the created object
  - Reference to "objects" within the handle
  - Execute what we need ... typically ...
    - Open an existing/new document
    - Edit the content (examine/insert/modify)
    - Save the document
    - The actual operations we perform are application dependent. (e.g. different operations for WORD vs Excel)

---

---

---

---

---

---

---

---