Week 6
Iteration; Sequential Algorithms; Processing Cells
Why loops in programming?

- Let us hear from Mark Zuckerberg (founder of Facebook) on why you need loops in programming
- https://www.youtube.com/watch?v=mgoogyWMTxk
Iteration (Repetition)

- Often need to execute statements repeatedly
- **Loops** are statements that can do this
- Process is called *iteration*
- Kinds of loop:
  - **For** (iterate a fixed number of times)
  - **While** (iterate as long as something is True)
  - **Do-Loop** (general iteration)
G’day, mate!

- I wish to say G’day to each student in this session’s ENGG1811 class
- Column A of the worksheet Greetings contains the first names of all the ENGG1811 students enrolled in 15s2. (There are 411 unique names. Order randomized.)
We need two skills

1. We need to know how to read from and write to a worksheet using OO Basic

2. Loops

• Let’s first look at how we can use OO Basic to interact with the worksheet
  - Worksheet: Interacting with worksheet
  - OO Basic module: BaiscWorksheetDemo
Reading from or writing to worksheets

- The objects we are most interested in are **ranges**, including single cells, on the active sheet
  
  **ActiveCell**
  
  **ActiveSheet.Cells(row,col)**
  
  - any integer expression for **row** and **col**
  - row counts from 1 to 1000000+
  - col counts from 1 (= A) to 1000+

  **ActiveSheet.Range(name)**
  
  - **name** is a string variable or literal, say "limit"

  **ActiveSheet.Range(address)**
  
  - **address** contains, say "A6" (single cell) or "$B5:D $11" (range)

- Can assign to a variable of type **Range** or more commonly act on directly
**Demo: get values from cells, write to cells**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Menu: Tools &gt; Macros &gt; Run Macro &gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Expand week06demo-orig.ods, and expand Standard</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Click on BasicWorkSheetDemo and click the Run button</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>23</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>x = ActiveSheet.Cells(5,1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>y = ActiveSheet.Cells(8,3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>sum = x + y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ActiveSheet.Cells(12,2) = sum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>ActiveSheet.Range(&quot;B14&quot;) =</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ActiveSheet.Range(&quot;A5&quot;) + ActiveSheet.Range(&quot;C8&quot;)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>ActiveSheet.Cells(16,2) = 2 * ActiveSheet.Range(&quot;HighFive&quot;)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>ENGG1811</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>ActiveSheet.Cells(18,2) = &quot;&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The code is in BasicWorkSheetDemo – see next slide
'Get values from two cells, add them up, and put the result in another cell
x = ActiveSheet.Cells(5,1)
y = ActiveSheet.Cells(8,3)
sum = x + y
ActiveSheet.Cells(12,2) = sum

'Use activesheet.range this time
ActiveSheet.Range("B14") = _
    ActiveSheet.Range("A5") + ActiveSheet.Range("C8")

'Use a name defined on the worksheet
ActiveSheet.Cells(16,2) = 2 * ActiveSheet.Range("HighFive")

'Clear a cell
ActiveSheet.Cells(18,2) = ""
Writing greetings manually (1)

Sub GreetingsManual1()
    'Put greetings in Column B manually
    'using names explicitly
    'ActiveSheet.Cells(1,2) = "G'day, name"

    ActiveSheet.Cells(1,2) = "G'day, Zishan"
    ActiveSheet.Cells(2,2) = "G'day, Leanne"
    ActiveSheet.Cells(3,2) = "G'day, Brenda"
    ActiveSheet.Cells(4,2) = "G'day, Ismail"

End Sub

Code in Module: GreetingsManual
Sub: GreetingsManual1
Writing greetings manually (2)

Sub GreetingsManual2()
    'Put greetings in Column B manually one by one
    'using the names in Column A
    'ActiveSheet.Cells(1,2) = "G'day, " & ActiveSheet.Cells(1,1)

    ActiveSheet.Cells(1,2) = "G'day, " & ActiveSheet.Cells(1,1)
    ActiveSheet.Cells(2,2) = "G'day, " & ActiveSheet.Cells(2,1)
    ActiveSheet.Cells(3,2) = "G'day, " & ActiveSheet.Cells(3,1)
    ActiveSheet.Cells(4,2) = "G'day, " & ActiveSheet.Cells(4,1)
    ActiveSheet.Cells(5,2) = "G'day, " & ActiveSheet.Cells(5,1)
End Sub

Questions: Among these lines:
What is the same? What is sequential?

• We need to advance sequentially 1, 2, 3, ...

Sub SimpleFor()
    Dim k As Long
    For k = 1 To 6
        MsgBox "k = " & k
    Next k
End Sub
Automating greetings

- Using for-loop to perform iterations

```
Const NAMES_COL = 1
Const GREETINGS_COL = 2
Const LAST_ROW = 411

Sub GreetingsAutomatic()
    Dim row As Long
    For row = 1 To LAST_ROW
        ActiveSheet.Cells(row, 2) = "G'day, " & ActiveSheet.Cells(row, 1)
    Next row
End Sub
```

Should replace these explicit numbers by named constants
Simple form of for-loop

- **For** statement sets an integer **loop variable** to each value in a range and executes statements forming the **loop body** for each value:

  ```plaintext
  For intvar = start To finish
    statements
  Next intvar
  ```

- **start** and **finish** are expressions, evaluated once

  ```plaintext
  For k = 4 To 9
    MsgBox "k = " & k
  Next k
  ```
How to write a loop?

For intvar = start To finish
    Tasks to be repeated
Next intvar

• Need to figure out these two aspects:
  – Loop variable: starting value, finishing value
  – What task is to be repeated?

• Recommendation
  – Write out the long winded statements on paper
  – Observe: What is the same? What is sequential?
  – Translate the observation to a loop
How to write a for loop? (Example)

ActiveSheet.Cells(1,2) = "G'day, " & ActiveSheet.Cells(1,1)
ActiveSheet.Cells(2,2) = "G'day, " & ActiveSheet.Cells(2,1)
ActiveSheet.Cells(3,2) = "G'day, " & ActiveSheet.Cells(3,1)
...
ActiveSheet.Cells(411,2) = "G'day, " & ActiveSheet.Cells(411,1)

ActiveSheet.Cells(row,2) = "G'day, " & ActiveSheet.Cells(row,1)

The loop variable row goes from 1 to 411

Note: This may look like a lot of work in the beginning but once you have enough practice, you may be able to write the loop directly.
How does a loop work?

For row = 1 To LAST_ROW
    ActiveSheet.Cells(row, 2) = "G'day, " & 
    ActiveSheet.Cells(row, 1)
Next row

The loop variable row goes from 1 to 411
Set row = 1, execute the statements in the loop
Set row = 2, execute the statements in the loop
etc
More complicated for-loop

- We have used a for-loop to make greetings automatically
- Note that the task performed inside the loop does not depend on the result of the previous iteration
- We will be able to do something a lot more powerful if we can make use of the result from the previous iteration
Quiz: Summing a sequence of numbers

- I will roll a 12-sided die 100 times
- You are not allowed to write any of the numbers down (Hopefully you are not a mnemonicist!)
- I want you to tell me what the sum of those 100 numbers are
- How will you do it?

Let us have a go

Do you find yourselves repeating some actions?

https://openclipart.org/detail/92041/dice
Algorithm

(Roll 1\textsuperscript{st} time)
Remember the number from the die

(Roll 2\textsuperscript{nd} time)
Add the number from the die to the number you have remembered. Remember the new total.

(Roll 3\textsuperscript{rd} time)
Add the number from the die to the number you have remembered. Remember the new total

(Roll 4\textsuperscript{th} time)
Add the number from the die to the number you have remembered. Remember the new total
• Let us assume that you use a variable called `runningTotal` to remember the total so far.

Add the number from the die to the number you have remembered (i.e. `runningTotal`).

Update the value of the variable `runningTotal`.

• Question: How will you write the above task using one line of pseudo-code?

```
runningTotal = runningTotal + theNumberFromTheDie
```
We want to sum the numbers in Column A.

```
runningTotal = 0
runningTotal = runningTotal + ActiveSheet.Cells(2,1)
runningTotal = runningTotal + ActiveSheet.Cells(3,1)
runningTotal = runningTotal + ActiveSheet.Cells(4,1)
runningTotal = runningTotal + ActiveSheet.Cells(5,1)
```

```
runningTotal = 0
For row = 2 To 5
    runningTotal = runningTotal + ActiveSheet.Cells(row,1)
Next row
```
For statement sets an integer loop variable to each value in a range and executes statements forming the loop body for each value:

```
For intvar = start To finish Step amount
  statements
Next intvar
```

- `start` and `finish` are expressions, evaluated once
- `Step amount` is optional (default 1)
- `statements` may use (but not change) the value of the loop variable
- loop must terminate (may not iterate at all, depending on `start`, `finish` and `amount`
Quiz: want to compute \((5^4)*(8^4)*\ldots*(26^4)\)

1. How to initialize?
   
   \[
   \text{prod} = 1 \\
   \text{prod} = \text{prod} * 5^4 \\
   \text{prod} = \text{prod} * 8^4 \\
   \text{prod} = \text{prod} * 11^4 \\
   \text{prod} = \text{prod} * 14^4 \\
   \text{prod} = \text{prod} * 17^4 \\
   \text{prod} = \text{prod} * 20^4 \\
   \text{prod} = \text{prod} * 23^4 \\
   \text{prod} = \text{prod} * 26^4
   \]

2. Try to complete the for loop

   \[
   \text{prod} = 1 \\
   \text{For } i = 5 \text{ To 26 Step 3 } \\
   \quad \text{prod} = \text{prod} * i^4 \\
   \text{Next } i
   \]

Alternative answer:
   
   \[
   \text{For } i = 26 \text{ To 5 Step -3}
   \]
Quiz: want to compute \((5^4)(8^4)\ldots(26^4)\) (cont’d)

<table>
<thead>
<tr>
<th>prod = prod * 5^4</th>
</tr>
</thead>
<tbody>
<tr>
<td>prod = prod * 8^4</td>
</tr>
<tr>
<td>prod = prod * 11^4</td>
</tr>
<tr>
<td>prod = prod * 14^4</td>
</tr>
<tr>
<td>prod = prod * 17^4</td>
</tr>
<tr>
<td>prod = prod * 20^4</td>
</tr>
<tr>
<td>prod = prod * 23^4</td>
</tr>
<tr>
<td>prod = prod * 26^4</td>
</tr>
</tbody>
</table>

You can also write the for-loop in the following manner. You need to figure out what to put here.

```
For i = 1 To 8
    prod = prod *
Next i
```

You can post the answer on the forum.
How to teach dumb robots to tighten a screw

1. Put the screw in
2. Put the tip of the screwdriver to the head of the screw
3. **While** the screw is still loose:
   keep turning in the clockwise direction
More precise instructions for dumb robots

• Notes:
  • Diamond block
    – Test a condition
  • Square block
    – Actions

Still loose?

N

Y

Turn the screwdriver by half a degree
Iteration – While

- **While** statement continues to execute statements as long as a Boolean expression is True
  
  
  ```
  While boolean-expression
  statements
  Wend*
  ```

- Loop guard is evaluated
  - If it is **True** execute the loop body and go back to start of loop to re-test the guard
  - Otherwise (i.e., it is **False**) exit loop and continue with the statement following the loop

- Loop body must change state so that loop guard can eventually become False (else **infinite loop**)

* Why not **End While**? Beats me.
While loop: example 1

' 1. Print all powers of 2 less than LIMIT

Sub WhileDemo()
    Const LIMIT = 3
    Dim x As Integer

    x = 1
    While x < LIMIT
        MsgBox "x = " & x
        x = 2 * x
    Wend

    MsgBox "Out of the loop!"
End Sub
Walking through example 1

\[ x = 1 \]

- If \( x < \text{limit} \):
  - If \( Y \):
    - MsgBox \( x = x \times 2 \)
    - Go to x is 2
  - If \( N \):
    - MsgBox \( x = x \times 2 \)
    - Exit the loop

\( x \) is 2

\( x < \text{limit} \)

\( x \) is 4

- If \( x < \text{limit} \):
  - If \( Y \):
    - MsgBox \( x = x \times 2 \)
    - Go to x is 4
  - If \( N \):
    - MsgBox \( x = x \times 2 \)
    - Exit the loop

Leave the loop
Iteration – termination

- Loop body must change state so that loop guard can eventually become False (otherwise we have produced an *infinite loop*).

MC Escher (1948) *Handteckning* [Drawing Hands]; lithograph
http://cs.nyu.edu/courses/spring04/V22.0002-001/Escher_hands_2.jpg
Debugging Loops

- Always have the editor open when testing a procedure that has a loop
- \textit{Always} save the document from within the editor before testing anything with iteration
- If a program fails to terminate, the red stop button is your friend
  - VBA: have to use Ctrl-Alt-Break (Break may be labelled Pause), or \textbf{Esc}, UI is frozen until then
- If the stop button is greyed out, typing into the editor also works
- Set a breakpoint, rerun and single step
Quiz

This is the example earlier, the outputs are 1, 2

Sub DemoWhile()
    Const LIMIT = 3
    Dim x As Integer

    x = 1
    While x < LIMIT
        MsgBox "x = " & x
        x = 2 * x
    Wend
End Sub

Question: What is the output of the following program?

Sub WhileQ()
    Const LIMIT = 8
    Dim x As Integer
    x = 1
    While x < LIMIT
        x = 2 * x
        MsgBox "x = " & x
    Wend
End Sub

(a) 1 2 4
(b) 2 4
(c) 2 4 8
(d) 2 4 8 16
Learning to reason with while (1)

Problem: For a given number \( L \), you want to find the largest positive integer \( n \) such that

\[
1^3 + 2^3 + \ldots + n^3 \leq L
\]

Const \( L = 12345 \)

Dim \( n \) as long
Dim sumCube as long

\( n = 1 \)
\( \text{sumCube} = 0 \)

\textbf{while} \ \text{sumCube} \leq L
\ 
\text{sumCube} = \text{sumCube} + n^3
\ 
\( n = n + 1 \)
\textbf{wend}

msgBox "n = " \& n

Question: Can you use the code on the left to solve this problem? Does this \( n \) give the correct answer?

Code in WhileCube
Learning to reason with while (2)

• In order to reason whether the program returns the correct $n$, it’s best to deal with a small example.
• It’s hard to reason with $L = 12345$ (the value in the code) because we don’t even know what the correct $n$ should be.
• Let us choose $L = 12$.

• Question: What is the largest integral $n$ such that the following inequality holds for $L = 12$?
  a. 1
  b. 2
  c. 3
  d. 4

$$1^3 + 2^3 + \ldots + n^3 \leq L$$

Implication: If the program gives $n = 2$ for $L = 12$, then it returns the correct answer.
Learning to reason with while (3)

• We need to know the value of $n$ after the while loop has been executed for the last time
• This means we need to know
  – the program state just before the loop-guard becomes invalid
  – Or, equivalently, the last time the statements within the while-loop are executed

• Question: Consider the program with $L = 12$. What is the value of sumCube when the loop guard becomes invalid?
  a. $1$
  b. $1 + 2^3$
  c. $1 + 2^3 + 3^3$
  d. $1 + 2^3 + 3^3 + 4^3$
sumCube = sumcube + n^3
n = n + 1

sumCube = 1 + 2^3 + 3^3
n = 4
How to fix the program?

Problem: For a given number $L$, you want to find the largest positive integer $n$ such that

$$1^3 + 2^3 + ... + n^3 \leq L$$

Question: How will you fix the program to give the right $n$?

```
Const L = 12345

Dim n as long
Dim sumCube as long

n = 1
sumCube = 0

while sumCube <= L
    sumCube = sumCube + n^3
    n = n + 1
wend

msgBox "n = " & n
```
A few tips

• Use small examples, which you know the correct answers, to test your algorithm

• Think forward: Trace the loop and observe the pattern

• Think backward: Determine the condition under which the loop guard becomes invalid and then work backwards
When to use for? When to use while?

- No hard rules
  - For and while are largely interchangeable
  - Sometimes it’s personal preference

- One suggestion is:
  - Use for when the number of iterations is known in advance, otherwise use while

```plaintext
sum = 0
For i = 1 To 10
    sum = sum + i
Next i
```

```plaintext
sum = 0
i = 1
While (i <= 10)
    sum = sum + i
    i = i + 1
Wend
```
For – known number of iterations

- Marge is going on holiday for 5 days
- The number of days is known in advance

For day = 1 To 5
Clean the toilet
Next day

While – unknown number of iterations

- Marge is going on holiday but doesn’t know for how long

While (I am away on day)
Clean the toilet that day
\[ \text{day} = \text{day} + 1 \]
Wend
Other kinds of loop

Do-Loop statement is similar to While, but the loop body is always executed at least once:

\[
\text{Do} \quad \text{statements} \quad \text{Loop While } \text{boolean-expression} \\
\text{Loop guard}
\]

Alternative form uses complemented guard:

\[
\text{Do} \quad \text{statements} \quad \text{Loop Until } \text{boolean-expression} \\
\text{Loop guard}
\]

Can terminate the loop early with the statement

\[
\text{Exit Do} \quad \text{(also Exit For but not Exit While)}
\]
Worksheet Programming

- We can use values on the active worksheet as inputs to a program, and overwrite cells to show results and calculated values
- We can iterate over rows or columns of data
- We can detect empty cells to stop the iteration
- Example: running sum to the right of a column
AddRunningSum

- Running sum to the right of a column

Tools – Macros – Run Macro...
AddRunningSum

<table>
<thead>
<tr>
<th>A Values</th>
<th>B Running sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>-6</td>
</tr>
<tr>
<td>6</td>
<td>-2</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>-0.4</td>
</tr>
<tr>
<td>12</td>
<td>409.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A Values</th>
<th>B Running sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>-6</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>-2</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>18.6</td>
</tr>
<tr>
<td>12</td>
<td>427.7</td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
</tbody>
</table>
What you want to realize is:

- B2 is A2
- B3 is A2 + A3
- B4 is A2 + A3 + A4
- B5 is A2 + A3 + A4 + A5
- ...

It’s possible to implement these relationships in the form written above, but you need 2 loops – one inside another. We won’t try that.

We can do it with one loop. Can you rewrite these relationships so that B3, B4, … is the sum of two terms only, i.e.

- B3 is something + something
- B4 is something + something
- …
Rewriting the relationships

- B2 is A2
- B3 is A2 + A3
- B4 is A2 + A3 + A4
- B5 is A2 + A3 + A4 + A5
- ...

Rewrite so that B3, B4, etc are sum of 2 terms

- B2 is A2
- B3 is B2 + A3
- B4 is B3 + A4
- B5 is B4 + A5
- ...

Writing the iteration (1)

- B2 is A2
- B3 is B2 + A3
- B4 is B3 + A4
- B5 is B4 + A5
- ...

Express these relationships in terms of `ActiveSheet.Cells`

- `ActiveSheet.Cells(2,2) = ActiveSheet.Cells(2,1)`
- `ActiveSheet.Cells(3,2) = ActiveSheet.Cells(2,2) + ActiveSheet.Cells(3,1)`
- `ActiveSheet.Cells(4,2) = ActiveSheet.Cells(3,2) + ActiveSheet.Cells(4,1)`
- `ActiveSheet.Cells(5,2) = ActiveSheet.Cells(4,2) + ActiveSheet.Cells(5,1)`
- ...

This part goes in the loop. Let \( k \) be the loop variable.

\[
\text{ActiveSheet.Cells}(k,2) = _
\]

\[
\text{ActiveSheet.Cells}(k-1,2) + \text{ActiveSheet.Cells}(k,1)
\]
Running sum

Sub AddRunningSum()
  Dim row As Long  ' index always integral
  Dim sum As Double  ' numeric values normally real

  row = 2

  While ActiveSheet.Cells(row, COL_NUM) <> ""
    If row = 2 Then
      'Copy the value in column 1 to column 2
      ActiveSheet.Cells(row, COL_SUM) = _
      ActiveSheet.Cells(row, COL_NUM)
    Else
      ActiveSheet.Cells(row, COL_SUM) = _
      ActiveSheet.Cells(row - 1, COL_SUM) + _
      ActiveSheet.Cells(row, COL_NUM)
    End If
    row = row + 1
  Wend
End Sub
Running sum: Alternative solution

```vba
Sub AddRunningSum()
    Dim row As Long    ' index always integral
    Dim col As Long
    Dim sum As Double  ' numeric values normally real

    row = 2: col = 1: sum = 0
    ' FYI, the : symbol separates multiple simple statements
    While ActiveSheet.Cells(row, col) <> ""
        sum = sum + ActiveSheet.Cells(row, col)
        ActiveSheet.Cells(row, col+1) = sum
        row = row + 1
    Wend

End Sub
```
Sequential Algorithms

OO Basic and VBA are particularly useful for tasks such as
• Processing rows or columns on one pass
• Locating cells with particular characteristics
• Identifying extreme values (max/min)

These are called **sequential tasks**, because they treat the data as a *sequence* of items, usually stored in cells.

Standard sequential algorithms can be adapted to solve these kinds of problems.
I want to find the largest number in the rectangle

1. If the largest number behind the circle is 699, what is the largest number in the rectangle?
2. What if the largest number behind the circle is 934?
   a. Cover the rectangle
Finding the maximum of a column of numbers

Pseudo code

- max = biggest found so far
- newNumber = the next number in the column

If newNumber > max Then
    max = newNumber
End If

Code in FindMax
Maximum/minimum

- Max/min calculations maintain a “best so far” variable
- May also need to know the position of the (first or last occurrence of) the maximum

**Pseudocode for maximum of a column:**

```plaintext
set col to the required column number
set row to the row where the sequence starts
set max to the value in that cell
While cell at (row, col) is not empty
    If cell value is greater than max Then
        set max to cell value
    End If
    move row one cell down
Wend
```
More examples

- The demo workbook week04-heartbeat.ods shows you can use sequential algorithms to automatically count heart beats.
Exercises

- In order for you to understand loops, it would be good if you can read a program and tell what it does. Here are a number of exercises for you. We won’t talk about these in the lecture. You can discuss them and post your solution on the forum.

- Exercise 0 is a sample and has been done.

- You can try out Exercises 1-3.
Exercise 0: Sample

`(assume n is set beforehand and n >= 1)

sum = 0
For k = 1 To n
    sum = sum + k
Next k

Answer: Sum of integers from 1 up to n

Question: What happens if n < 1
Exercise 1

'(assume n is set beforehand)

sum = 0
For k = 1 To n
    sum = sum + 2*k
Next k

Answer: ??
Exercise 2

`(assume n is set beforehand and is a positive integer >= 2)

sum = 0  
k = 1  
While k < n  
    k = k + 1  
    sum = sum + 2*k  
Wend

Answer: ??
Exercise 3

' (Assume $n$ is set beforehand and is a positive integer)

```plaintext
sum = 0
For j = 1 To n
  For k = 1 To j
    sum = sum + k
  Next k
Next j
```

Answer: ??
### String and Character Functions

<table>
<thead>
<tr>
<th>Function call</th>
<th>Returns…</th>
</tr>
</thead>
<tbody>
<tr>
<td>Len(str)</td>
<td>number of characters in str</td>
</tr>
<tr>
<td>Mid(str,start,len)</td>
<td>len-char substring of str, pos @ start; also Left(str,len) and Right(str,len)</td>
</tr>
<tr>
<td>Asc(str)</td>
<td>char code for first char in str</td>
</tr>
<tr>
<td>Chr(code)</td>
<td>Unit string containing char with code</td>
</tr>
<tr>
<td>LCase(str)</td>
<td>str with all upper-case chars converted to lower case (also UCcase)</td>
</tr>
<tr>
<td>Trim(str)</td>
<td>str with spaces removed from both ends (also LTrim, RTrim)</td>
</tr>
<tr>
<td>InStr(str,match)</td>
<td>first position in str where match occurs, or 0</td>
</tr>
<tr>
<td>Replace(str,match,rep1)</td>
<td>str with all occurrences of match replaced by repl (other args possible)</td>
</tr>
<tr>
<td>String(str,n)</td>
<td>n copies of first char of str</td>
</tr>
</tbody>
</table>
Can mix up numeric types (angle + intOffset is evaluated as Double)

When explicit conversion is needed, use function:

<table>
<thead>
<tr>
<th>Function name</th>
<th>Accepts as argument</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>CStr</td>
<td>number</td>
<td>String (also <strong>Format</strong> next slide)</td>
</tr>
<tr>
<td>CLng</td>
<td>number or string</td>
<td>Long Integer (rounded if from Double)</td>
</tr>
<tr>
<td>CDbl</td>
<td>number or string</td>
<td>Double</td>
</tr>
<tr>
<td>Val</td>
<td>string</td>
<td>Integer or Double, ignores trailing junk</td>
</tr>
<tr>
<td>Round((v,d))</td>
<td>number</td>
<td>$v$ Rounded to $d$ decimal places</td>
</tr>
<tr>
<td>Hex</td>
<td>integer</td>
<td>Hexadecimal string form</td>
</tr>
</tbody>
</table>
Format Function

Specialised formatting of numbers and dates

Format(expr, fmt) returns a formatted string

fmt mixes ordinary characters and formatting codes:

<table>
<thead>
<tr>
<th>Predefined code</th>
<th>Displays…</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>digit or zero</td>
</tr>
<tr>
<td>#</td>
<td>digit or nothing</td>
</tr>
<tr>
<td>%</td>
<td>percentage placeholder ($\times 100$, show %)</td>
</tr>
<tr>
<td>e+ e- E+ E-</td>
<td>scientific notation</td>
</tr>
<tr>
<td>@</td>
<td>single char or space</td>
</tr>
<tr>
<td>&lt; &gt;</td>
<td>force lower-case or upper-case</td>
</tr>
<tr>
<td>ddd mmm yy hh nn</td>
<td>date/time components (see text)</td>
</tr>
</tbody>
</table>

Can also use predefined formats such as Scientific, Percent, Long Date etc. See Help for details.
“Give me 26 soldiers of lead and I will conquer the world.” (Benjamin Franklin)

- Assignment, if-then-else, functions, for, while are the most fundamental elements of programming. OO Basic has them and so do many other programming languages, though the syntax may be different.

- “Give me the programming fundamentals, and I will compute the world.”
Summary

- Reference parameters allow a subprogram to change the value of variables declared outside.
- Use pseudocode to express algorithm prior to coding in OO Basic.
- Active sheet and its cells can be addressed directly.
- Sequential algorithms process a row or column of data systematically.
- Some algorithms accumulate values, others process cells locally and order-independently.
- Strings store simple character sequences.
- Built-in functions evaluate commonly-required quantities (string and numeric).
- Relatively complex tasks can usually be subdivided into easily managed subtasks and solutions combined.