ENGG1801 Week 4 Lecture 2

If Statements & Arrays

Key Revision Points:

- If Statements
  - Used to do some action(s) if some condition is true
  - Example:
    - % A student scores 90 in ENGG1801
    - mark = 90
    - % Print a nice message only if the student passed
    - **if** mark >= 50
      - disp('Pass!') ;
    - **end**
    - % Always print the mark whether pass or fail
    - disp(mark) ;
  - Always indent to make code clearer
    - Do NOT indent comments after the % symbol
    - Use only a single space
  - There are no semi-colons at the end of the lines of code with **if** and **end**
  - Immediately after the **if** keyword, there must be a logical (Boolean) expression (something that can only be true or false)
    - Number/Value Comparison Operations:
      - == is used for "equal to", as = is used to assign values
      - ~= is not equal to
    - Boolean Comparison Operations:
      - && is and
      - || is or (pipe key, not LL)
      - ~ is not
  - Complex Boolean Expressions
    - % Check if above dangerous conditions occur
    - **if** (temp >= 70 | | temp < 4) | | (pressure > 30 && temp > 50)
      - disp('Danger!') ;
    - **end**
- Else Statements
  - Use else to do something else if the condition is false
    - % A number that we will test
    - number = -3 ;
    - % Print a message depending on whether the number is positive or not
    - **if** number > 0
      - disp('Positive') ;
    - **else**
      - disp('Not positive') ;
    - **end**
  - Notice that there is **no condition** after the else keyword

- Elseif Statements
    - Use elseif to test many conditions
        - % A number that we will test
        - number = -3 ;
        - % Only 1 of these 3 print statements will occur
        - **if** number > 0
            - disp('Positive') ;
        - **elseif** number < 0
            - disp('Negative') ;
        - **else**
            - disp('Zero') ;
        - **end**
    - Condition after elseif is **only tested if the condition after if is false, otherwise it is skipped**
    - Many elseif's can be used in an if statement
- Each if must have a corresponding end
- Nested If Statements – We can put if statements inside other if statements (also works for else, elseif)
    - **if** mark >= 50
        - disp('You passed!') ;
        - **if** mark >= 85
            - disp('High Distinction! Congratulations!') ;
        - **else**
            - disp('But you missed a High Distinction') ;
    - **else**
        - disp('Failed') ;
    - **end**
- Matrices
    - Scalar is a number
    - Matrix is an array of numbers
- Arrays
    - Vectors and Matrices are represented as **arrays** in Matlab
        - % Store many marks under 1 variable
        - marks1 = [62, 78.5, 47] ;
        - % We can also leave out the commas
        - marks2 = [62 78.5 47] ;
    - Each row is separated by semi-colons
        - % Store some data in a matrix
        - data1 = [1, 2, 3; 4, 5, 6] ;
        - % We can also leave out the commas
        - data2 = [1 2 3; 4 5 6] ;
        - % Store many numbers in a column vector
        - constants = [23; 47; 7] ;
    - Appears in workspace

- We can access (find out the value of) an element in the array
  - % Store scalar in a variable
  - marks = [62 78 47 68 90 53] ;
  - % Store the 4th mark into a separate variable
  - aStudentsMark = marks(4)
  - % This prints 68 to the screen
  - disp(aStudentsMark) ;
  - % This also prints 68 to the screen
  - disp(marks(4)) ;
- To access a matrix, we need to give the row number first then the column number
  - anElement = data(2,3) ;
  - disp(anElement) ; or disp(data(2,3)) ;
- We can change a value in an array
  - marks(3) = 50 ;
- We can increase an array's length if we insert into an index (position) that is longer than the array
  - marks(9) = 82 ;
- We can remove an element from an array
  - marks(3) = [] ;
- Other ways to create arrays:
  - numbers1 = 5:12 ;
    - Only in ascending order
  - numbers2 = 1:3:20 ;
    - 1-20 with 3 increment each time
  - numbers3 = 10:-2:0 ;
    - 10-0 with 2 decrement each time
  - matrix1 = zeros(2,3) ;
    - Creates a matrix of 2 rows and 3 columns containing only 0's
  - matrix2 = ones(1,5)
    - Creates a matrix of 1 row and 5 columns containing only 1's
  - numbers = linspace(0, 10, 6) ;
    - Creates a matrix, with the first value being 0, the last value being 10 and 6 values in the array that are equal distance (linearly) spaced apart