If, elseif, else

❑ Execute the statements if certain condition is true

❖ Syntax:

```
if expression
    statements
elseif expression
    statements
else
    statements
end
```

- ✓ The elseif and else blocks are optional.
- ✓ The statements execute only if previous expressions in the if...end block are false.
- ✓ An if block can include multiple elseif blocks
- ✓ You can nest any number of if statements. Each if statement requires an end keyword.
- ✓ Avoid adding a space after else within the elseif keyword (else if). The space creates a nested if statement that requires its own end keyword

Tips

- You can nest any number of if statements. Each if statement requires an end keyword.

- Avoid adding a space after else within the elseif keyword (else if). The space creates a nested if statement that requires its own end keyword.

Let see some worked out examples…

Switch, case, otherwise

❑ Execute one of several group of statements

❖ **Syntax:**

```
switch switch_expression
  case case_expression
    statements
  case case_expression
    statements
   ...
  otherwise
    statements
end
```

The switch block tests each case until one of the case expressions is true. A case is true when:

- ✓ For numbers, case_expression == switch_expression.
- ✓ For character vectors, strcmp(case_expression,switch_expression) == 1.
- ✓ For objects that support the eq function, case_expression == switch_expression. The output of the overloaded eq function must be either a logical value or convertible to a logical value
- ✓ For a cell array case_expression, at least one of the elements of the cell array matches switch_expression, as defined above for numbers, character vectors, and objects.

Tips

- A case_expression cannot include relational operators such as <, or > for comparison against the switch_expression. To test for inequality, use if, elseif, else statements.
- The MATLAB switch statement does not fall through like a C language switch statement. If the first case statement is true, MATLAB does not execute the other case statements. For example:

```
result = 52;

switch(result)
   case 52
      disp('result is 52')
   case {52, 78}
      disp('result is 52 or 78')
end
```

```
>> switch_example
result is 52
```

Let see some worked out examples…

For

❑ Perform iterative tasks for specified number of times

❖ **Syntax:**

```
for index = values
   statements
end
```

for *index* = *values*, *statements*, end executes a group of statements in a loop for a specified number of times. values has one of the following forms:

- ✓ *initVal*:*endVal* — Increment the index variable from *initVal* to *endVal* by 1, and repeat execution of statements until index is greater than *endVal*.
- ✓ *initVal*:*step*:*endVal* — Increment index by the value step on each iteration, or decrements index when step is negative.
- ✓ *valArray* — Create a column vector, index, from subsequent columns of array *valArray* on each iteration. For example, on the first iteration, *index* = *valArray(:,1)*. The loop executes a maximum of n times, where n is the number of columns of *valArray*, given by numel*(valArray(1,:))*

Tips

- To programmatically exit the loop, use a break statement. To skip the rest of the instructions in the loop and begin the next iteration, use a continue statement
- Avoid assigning a value to the index variable within the loop statements. The for statement overrides any changes made to index within the loop
- To iterate over the values of a single column vector, first transpose it to create a row vector.

Let see some worked out examples…

While

❑ While loop to repeat when condition is true

❖ **Syntax:**

while expression
statements
end

- ✓ while expression, statements, end evaluates an expression, and repeats the execution of a group of statements in a loop while the expression is true.
- ✓ An expression is true when its result is nonempty and contains only nonzero elements (logical or real numeric). Otherwise, the expression is false.

Tips

- If you inadvertently create an infinite loop (that is, a loop that never ends on its own), stop execution of the loop by pressing Ctrl +C
- If the conditional expression evaluates to a matrix, MATLAB evaluates the statements only if all elements in the matrix are true (nonzero). To execute statements if any element is true, wrap the expression in the any function.
- To programmatically exit the loop, use a break statement. To skip the rest of the instructions in the loop and begin the next iteration, use a continue statement.
- When nesting a number of while statements, each while statement requires an end keyword
- The MATLAB while loop is similar to a do...while loop in other programming languages, such as C and C++. However, while evaluates the conditional expression at the beginning of the loop rather than the end.

Let see some worked out examples…

Try, catch

❑ Execute statements and catch resulting errors

❖ **Syntax:**

```
try
   statements
catch exception
   statements
end
```

- ✓ *try statements*, *catch statements* end executes the statements in the try block and catches resulting errors in the catch block.
- ✓ This approach allows you to override the default error behavior for a set of program statements.
- ✓ If any statement in a try block generates an error, program control goes immediately to the catch block, which contains your error handling statements

Let see some worked out examples…

Pause

❑ Stop MATLAB execution temporarily

❖ Syntax:

Pause
Pause(n)
Puse(‘off’)
pause(‘query’)
Pause(‘on’)

Let see some worked out examples…

Thanks for your kind attention…