What is IML

- SAS/IML is an interactive programming language. I=Interactive, M=Matrix, L=Language. Matrices are the fundamental objects.
- The programming language is dynamic, interactive because necessary activities such as memory allocation and dimensioning of matrices are performed automatically.
Characteristics of IML

IML
- operates on matrices (numeric or character);
- possesses powerful operators, functions, and call routines;
- is interactive;
- allows dynamic allocation of storage for matrices, modules;
- allows access to multiple files all at once;
- provides full access to SAS data files;
- produces graphics.

Defining Matrix

- Matrices can be either numeric (stored as double precision) or character (of same length up to 32,767 bytes).
- Matrices are stored in named objects (SAS names) by using assignment operator =.
- Matrices are defined row-wise (elements separated by spaces, and rows separated by commas) and are enclosed in braces {}.
- Character elements, when enclosed in matching quotes, are case sensitive. Otherwise, all characters are converted to upper case.
Matrix Operations
Element-wise Operations

- **arithmetics**  
  - (prefix, reversing sign)  
  +  −  #  ##  /  
  Note that + operator also allows character operands for string concatenation operation which works exactly as CONCAT function. Missing value results when at least one operand has missing value

- **comparisons**  
  >  <  =  >=  <=  ^=  
  Missing element is the smallest in comparison.

- **logical**  
  ^ (prefix)  &  |  

- **element-wise extrema**  
  <> (element-wise maxima)  
  >> (element-wise minima)

Matrix Operations
Matrix Operations

- **algebraic**  
  * (matrix product)  ** (matrix power)  ' (back quote, transpose)  @ (direct product or Kronecker product)

- **concatenation**  
  || (binds columns)  // (binds rows)

- **special operator**  
  : (index creation operator) creates row vector value1:value2 (numeric) or ’prefix1:’prefixn’ (character)
  Use DO function for numeric vectors with increments other than 1 or -1.
**Subscripting**

**Subscript**

```plaintext
operand[rows, cols] operand[elements] where

- **operand**: usually a matrix name, but it can also be an expression or literal (i.e., constant).
- **rows**: an expression, either scalar or vector, for selecting one or more rows from the operand.
- **cols**: an expression, either scalar or vector, for selecting one or more columns from the operand.

Note:
1. **rows** and **cols** can be numeric or character (names)
2. subscripted matrix can appear on the left of an assignment statement

**Subscripting**

**Subscript reduction**

- **row reduction**: `operand[operator,]`
- **column reduction**: `operand[,operator]`
- **row (first) and column (second) reduction**: `operand[operator,operator]`
- **entire matrix reduction**: `operand[operator]`

Available reduction operators are `+` `#` `##` `<>` `<<` `<>:` `>:<`. Can combine subscripting with subscript reduction. Unless all missing, missing values are excluded from the reduction operation. See examples ProcIML01.sas–ProcIML08.sas.
RESET Statement

to set processing options

RESET <options>;

Some options (defaults first)

- **overall printing**: CENTER|NOCENTER, NOPRINT|PRINT, NOPRINTALL|PRINTALL, LINESIZE=n(default=78), PAGESIZE=n(default=21), FW=n(default=9), SPACES=n(default=4)

- **printing matrices**: NAME|NONAME, NOAUTONAME|AUTONAME, NOFUZZ|FUZZ,

- **printing destination**: NOLOG|LOG

Use **SHOW OPTIONS**; statement to show options currently in effect

PRINT Statement

PRINT <matrices> <(expression)> <$"message"$> <pointer-controls> <[options]>;

where (in any sensible order)

- **matrices**: names of matrices

- **(expression)**: expression in parentheses that is evaluated

- **"message"**: message in matching quotes

- **pointer-controls**: control the pointer for printing—using a comma (,) to skip a single line and using a slash (/) to skip to a new page.

- **[options]**: options enclosed in square brackets and separated by blanks— COLNAME=matrix, ROWNAME=matrix, FORMAT=format, LABEL=label
Selected Functions
for creating matrices

- \( \mathbf{I} \) (dimension) \& \( \mathbf{J} \)(nrow\(<\), ncol\(<\), value\(>\))
- \( \text{REPEAT} \) (matrix, nrow, ncol), \( \text{SHAPE} \) (matrix\(<\), nrow\(<\), ncol\(<\), pad-value\(>\>) and \( \text{CSHAPE} \) (matrix, nrow, ncol, size\(<\), padchar\(>\))
- \( \text{BLOCK} \) (matrix1\(<\), matrix2, ..., matrix15\(>\))
- \( \text{DO} \) (start, stop, increment)
- \( \text{DIAG} \) (argument) and \( \text{VECDIAG} \) (square-matrix)
- \( \text{DESIGN} \) (column-vector) and \( \text{DESIGNF} \) (column-vector)

Selected Functions
for information about matrices

- \( \text{NCOL} \) (matrix) and \( \text{NROW} \) (matrix)
- \( \text{LENGTH} \) (matrix) and \( \text{NLENG} \) (matrix)
- \( \text{LOC} \) (matrix)
- \( \text{DET} \) (square-matrix), \( \text{EIGVAL} \) (square-matrix) and \( \text{EIGVEC} \) (square-matrix)