Module 1 Topics

- Control flow
  - Conditional statement, for-loop, while-loop
- User-defined functions
  - Script vs. function
- Arrays
  - Numeric arrays, 1-d, 2-d, vectorized arithmetic
- Syntax and basic built-in functions
  - input, fprintf, disp, elementary math functions
  - Appropriate use of random number generators

- Graphics (use of plot) will not be asked on a test
- Do not use break, continue
The **if** construct

```plaintext
if  boolean expression1
    statements to execute if  expression1 is true
elseif  boolean expression2
    statements to execute if  expression1 is false
    but  expression2 is true
:
else
    statements to execute if all previous conditions are false
end
```

Can have any number of **elseif** branches but at most one **else** branch.
Generating random numbers

- \texttt{rand(m,n)} gives an \( m \times n \) matrix of random values, each in interval \((0,1)\)

- Generate a random number in the range \((a,b)\)

- Generate a random integer in the range \([a,b]\)
Generating random numbers

- `rand(m, n)` gives an m-by-n matrix of random values, each in interval (0,1)

- Generate a random number in the range (a,b)
  \[ \text{rand} \times (b-a) + a \]

- Generate a random integer in the range [a,b]
  \[ \text{floor} (\text{rand} \times (b-a+1) + a) \]
  \[ \text{ceil} (\text{rand} \times (b-a+1) + a-1) \]
Built-in functions for creating/manipulating arrays

- **Creation**
  - zeros, ones, rand
  - linspace

- **Manipulation**
  - length
  - size
Built-in functions for creating/manipulating arrays

- **Creation**
  - zeros, ones, rand
  - linspace

- **Manipulation**
  - length
  - size

\[
\text{zeros}(3, 2)
\]

\[
\begin{bmatrix}
\frac{3}{2}, 10, \frac{8}{3}
\end{bmatrix},
\text{starting} \quad \text{end} \quad \# \text{of values}
\]

\[x = \text{linspace}\left(\begin{bmatrix}\frac{3}{2}, 10, \frac{8}{3}\end{bmatrix}\right)\]

\[
[\text{nr}, \text{nc}] = \text{size}(M)
\]

\[a = [4 \ 2 \ 3 ; \text{ones}(2,3)]\]
Common loop patterns

Do something \( n \) times

\[
\text{for } k = 1:1:n \\
\text{ % Do something } \\
\text{end}
\]

Do something an indefinite number of times

\[
\text{% Initialize loop variables} \\
\text{while ( not stopping signal )} \\
\text{ % Do something } \\
\text{% Update loop variables} \\
\text{end}
\]
**for loop examples**

for k = 2:0.5:3  
disp(k)  
end

k takes on the values 2, 2.5, 3
Non-integer increment is OK

for k = 1:4  
disp(k)  
end

k takes on the values 1, 2, 3, 4
Default increment is 1

for k = 0:-2:-6  
disp(k)  
end

“Increment” may be negative

for k = 0:-2:-7  
disp(k)  
end

Colon expression specifies a bound

for k = 5:2:1  
disp(k)  
end

The set of values for k is the empty set: the loop body won’t execute
for k = 4:6
    disp(k)
k= 9;
disp(k)
end

Not a condition (boolean expression) that checks whether k<=6.

It is an expression that specifies values:

4 5 6
Write a function \texttt{evalPoly} to evaluate an $n$\textsuperscript{th} order polynomial of $x$:

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

- Input parameter \texttt{coef} has length $n+1$, contains the coefficients of the polynomial
- \texttt{coef(1)} is the coefficient for the term $x^0$
- Input parameter \texttt{x}
- Return the value of the polynomial evaluated at \texttt{x}
- No Matlab predefined function other than \texttt{length}
$c_{01}$ | 1 | 2 | 3 | 4

$C_1 X^0 + C_2 X^1 + C_3 X^2 + C_4 X^3$
\[ c_1 x^0 + c_2 x^1 + c_3 x^2 + c_4 x^3 \]

function \( \text{val} = \text{evalPoly}(\text{coef}, x) \)

\%
val is polynomial evaluated at \( x \)
\%
coef is a vector where coef(1) is for term \( x^0 \)

\( \text{val} = \text{coef}(1) \)

for \( k = 2 : \text{length}(\text{coef}) \)

\( \text{val} = \text{val} + \text{coef}(k) \times x^{(k-1)}; \)

end
Simulation problem:
- Ann and Bob take turns flipping an unfair coin—twice as likely to be heads than tails
- In one round, each player flips once
- Ann gets 1 point if she gets heads; Bob gets 2 points if he gets tails
- Game ends after the round in which at least one player gets 10 points. Display the final scores.
Simulation problem:

- Ann and Bob take turns flipping an unfair coin—twice as likely to be heads than tails.
- In one round, each player flips once.
- Ann gets 1 point if she gets heads; Bob gets 2 points if he gets tails.
- Game ends after the round in which at least one player gets 10 points. Display the final scores.

```plaintext
pA = 0; pb = 0;
while pA < 10 & & pb < 10
    rand;
    if r < 2/3
        pA = pA + 1;
    end
    rand;
    if r
        pb = pb + 2;
    end
end
```

Display final scores.
Write a function triSums to return the column sums of the largest lower left triangular part of matrix M (same number of elements on each side of the triangle; including the main diagonal if matrix is square)
Write a function triSums to return the column sums of the largest lower left triangular part of matrix M (same number of elements on each side of the triangle; including the main diagonal if matrix is square)

```matlab
function colsums = triSums(M)

[nr, nc] = size(M);

colsums =
```
Write a function triSums to return the column sums of the largest lower left triangular part of matrix M (same number of elements on each side of the triangle; including the main diagonal if matrix is square)

```
function colsum = triSums(M)

[nr, nc] = size(M);
minD = min(nr, nc);
colsums = zeros(1, minD);
for c = 1: nc - minD
    for r = 0 : nr
        colsum(c) = colsum(c) + 
    end
end
end
```
Other notes for the test (course)

- Read questions/instructions carefully
- Use Matlab syntax
- Do not use `break, continue`
- Use `randi, rand` (and other functions) only as specified in the questions
- Many students make “index out-of-bounds” error