- Today's topics
  - Review of topics for Test 1
    - Lecture up to and including Lecture 8 (9/24)
    - Exercises up to and including Lab 4 (9/22)
    - Assignments 1 and 2
    - But nothing on type char, logical indexing

- Announcements/Reminders:
  - Test 1 on Wednesday, 09/29. Time is from 14:40-15:30, 205, Thurston
  - Assignment 2 due tonight 11:59pm. Submit it tonight—don't incur the late penalty! After grading we will re-open A2 submission in CMS.

# The `if` construct

```
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

- `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)
- Generate a random integer in the range [a,b]

# Built-in functions for creating/manipulating arrays

- Creation
  - zeros, ones, rand
  - linspace
  - Colon expression
- Manipulation
  - length
  - size
  - Concatenation

# Common loop patterns

Do something *n* times

```
for k= 1:1:n
    % Do something

end
```

Do something an indefinite number of times

```
%Initialize loop variables

while ( not stopping signal )
    % Do something

    % Update loop variables

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
```

`k` takes on the values 0,-2,-4,-6

"Increment" may be negative

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

`k` takes on the values 0,-2,-4,-6

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:

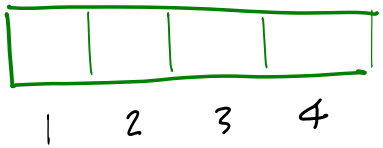
# Example

- Write a function **`evalPoly`** to evaluate an $n^{\text{th}}$ order polynomial of $x$ :

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

- Input parameter **`coef`** has length $n+1$, contains the coefficients of the polynomial
- **`coef(1)`** is the coefficient for the term $x^0$
- Input parameter **`x`**
- Return the value of the polynomial evaluated at $x$
- No Matlab predefined function other than **`length`**

coef




$$c_1 x^0 + c_2 x^1 + c_3 x^2 + c_4 x^3$$

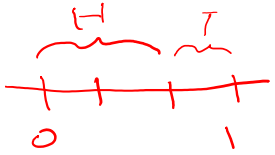
Simulation problem:

- 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.

stop: pA >= 10 ok pB >= 10

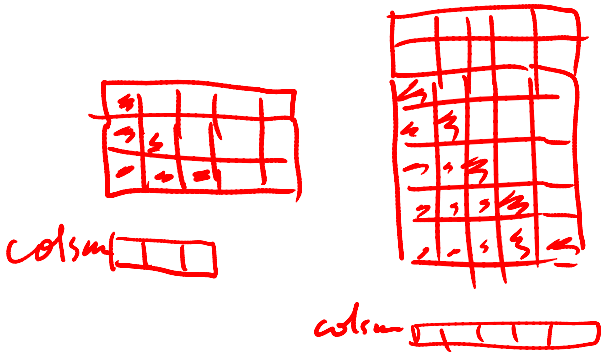
Simulation problem:

- 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.




```
pA = 0;  pB = 0;
while  pA < 10 && pB < 10
    % 1 round ; Ann flips ; Bob flips
    r = rand;
    if  r < 2/3
        pA = pA + 1;
    end
    r = rand;
    if  r >= 2/3
        pB + pB + 2;
    end
end
```

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)



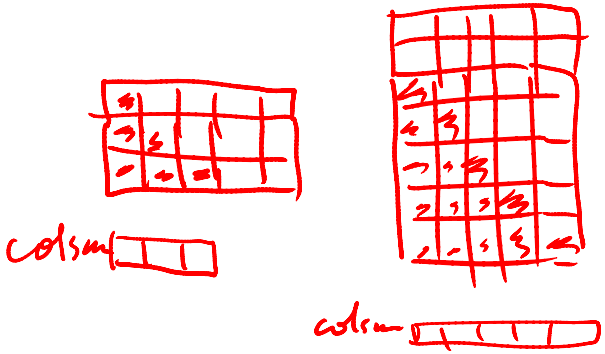
Write a function triSums to return the column sums of the largest lower left triangular part of matrix M (same number of elements on each side of the triangle; including the main diagonal if matrix is square)

```
function colSum = triSums(M)

[nr, nc] = size(M);
minDimension = min(nr, nc);
d = nr - minDimension;
colSum = zeros(1, minDimension);

for c = 1 : minDimension
    s = 0;
    for r = d + c : nr
        s = s + M(r, c);
    end
    colSum(c) = s;
end
```

c, not 1



Function header is the "contract" for how the function will be used (called)

## You have this function:

```
function  [x, y] = polar2xy(r, theta)
% Convert polar coordinates (r, theta) to
% Cartesian coordinates (x,y).  Theta in degrees.
…
```

## Code to call the above function:

```
% Convert polar (r1,t1) to Cartesian (x1,y1)
r1= 1;   t1= 30;
[x1, y1]= polar2xy(r1, t1);
plot(x1, y1, 'b*')
…
```

# Other notes for the test (course)

- Read questions/instructions carefully
- Use Matlab syntax
- Do not use `break`, `continue`, `return`
  - In general don't use functions/commands not covered in the course
- Use `randi`, `rand` (and other functions) only as specified in the questions
- Many students make "index out-of-bounds" error