- Today's topics
  - Review of topics for Test 1
  - Characters and strings

- Announcements/Reminders:
  - Assignment 1b due tonight 11:59pm
  - Test 1 in class on Thursday
  - Check grading comments in CMS

The **if** construct

if $\boxed{\textit{boolean expression1}}$

statements to execute if $\boxed{\textit{expression1}}$ *is true*

elseif $\boxed{\textit{boolean expression2}}$

statements to execute if $\boxed{\textit{expression1}}$ *is false*

but $\boxed{\textit{expression2}}$ *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]

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.

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:

4	5	6

Built-in functions for creating/manipulating arrays

- Creation
  - zeros, ones, rand
  - linspace
- Manipulation
  - length
  - size

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*’)
…
```

Given this function:

```
function m = convertLength(ft,in)
% Convert length from feet (ft) and inches (in)
% to meters (m).

   . . .
```

How many proper calls to convertLength are shown below?

```
% Given f and n
d= convertLength(f,n);
d= convertLength(f*12+n);
d= convertLength(f+n/12);
x= min(convertLength(f,n), 1);
y= convertLength(pi*(f+n/12)^2);
```

A: 1 B: 2 C: 3 D: 4 E: 5 or 0

Other notes for the test (course)

- Read questions/instructions carefully
- Use Matlab syntax
- Do not use `break`, `continue`
- Do not use `randi`, instead use `rand`
- Many students make “index out-of-bounds” error

Characters & strings

- We have used strings already:
  - `n= input('Next number: ')`
  - `sprintf('Answer is %d', ans)`
- A string is made up of individual characters, so a string is a 1-d array of characters
- '`CS1112 rocks!`' is a character array of length 13; it has 7 letters, 4 digits, 1 space, and 1 symbol.

'C'	'S'	'1'	'1'	'1'	'2'	' '	'r'	'o'	'c'	'k'	's'	'!'

- Can have 2-d array of characters as well

'C'	'S'	'1'	'1'	'1'	'2'
'r'	'o'	'c'	'k'	's'	'!'

2×6 matrix

Matlab types: `char`, `double`, `uint8`, `logical`

There is not a type "string"! What we call a string is a *1-d array of chars*

a	'C'	'S'	'1'

a is a 1-d array with type `char` components. We call **a** a "string" or "char array"

```
b = [3 9]
```

b is a 1-d array with type `double` components. `double` is the default type for numbers in Matlab. We call **b** a "numeric array"

```
d = rand > .5
```

d is a scalar of the type `logical`. We call **d** a "boolean value"

Single quotes enclose strings in Matlab

Anything enclosed in single quotes is a string (*even if it looks like something else*)

`‘100’`	is a character array (string) of length 3
`100`	is a numeric value
`‘pi’`	is a character array of length 2
`pi`	is the built-in constant 3.1416…
`‘x’`	is a character (vector of length 1)
`x`	may be a variable name in your program

Strings are vectors

Vectors

- Assignment

```
v= [7 0 5];
```

- Indexing

```
x= v(3);     % x is 5
v(1)= 1;     % v is [1 0 5]
w= v(2:3);   % w is [0 5]
```

- : notation

```
v= 2:5;      % v is [2 3 4 5]
```

- Appending

```
v= [7 0 5];
v(4)= 2;     % v is [7 0 5 2]
```

- Concatenation

```
v= [v [4 6]];
                 % v is [7 0 5 2 4 6]
```

Strings

- Assignment

```
s= 'hello';
```

- Indexing

```
c= s(2);     % c is 'e'
s(1)= 'J';   % s is 'Jello'
t= s(2:4);   % t is 'ell'
```

- : notation

```
s= 'a':'g';  % s is 'abcdefg'
```

- Appending

```
s= 'duck';
s(5)= 's';   % s is 'ducks'
```

- Concatenation

```
s= [s ' quack'];
                 % s is 'ducks quack'
```

Some useful string functions

```
str= ‘Cs 1112’;

length(str)   % 7
isletter(str) % [1 1 0 0 0 0 0]
isspace(str)  % [0 0 1 0 0 0 0]
lower(str)    % ‘cs 1112’
upper(str)    % ‘CS 1112’

ischar(str)
  % Is str a char array? True (1)
strcmp(str(1:2),‘cs’)
  % Compare strings str(1:2) & ‘cs’. False (0)
strcmp(str(1:3),‘CS’)
  % False (0)
```

Example: capitalize 1st letter

Write a function to capitalize the first letter of each word in a string. Assume that the string has lower case letters and blanks only. (OK to use built-in function **`upper`**)

```
function [str, nCaps] = caps(str)
% Post: Capitalize first letter of each word.
%     str = partially capitalized string
%     nCaps = no. of capital letters
% Pre: str = string with lower case letters & blanks only
```

```
look for   the  spaces

Look For   The  Spaces
```

See **`caps.m`**

ASCII characters

(American Standard Code for Information Interchange)

ascii code	*Character*	*ascii code*	*Character*
:	:	:	:
:	:	:	:
65	‘A’	48	‘0’
66	‘B’	49	‘1’
67	‘C’	50	‘2’
:	:	:	:
90	‘Z’	57	‘9’
:	:	:	:

Character vs ASCII code

```
str= ‘Age 19’
    %a 1-d array of characters
code= double(str)
    %convert chars to ascii values
str1= char(code)
    %convert ascii values to chars
```

Arithmetic and relational ops on characters

- `‘c’-‘a’` gives 2
- `‘6’-‘5’` gives 1
- `letter1=‘e’; letter2=‘f’;`
- `letter1-letter2` gives -1

- `‘c’>’a’` gives true
- `letter1==letter2` gives false

- `‘A’ + 2` gives 67
- `char(‘A’+2)` gives ‘C’

What is in variable g (if it gets created)?

```
d1= 'Mar 3';   d2= 'Mar 9';
x1= d1(5);   x2= d2(5);
g= x2-x1;
```

A: the character '**6**'

B: the numeric value 6

C: Error in the subtraction operation

D: Error in assigning variables **x1**, **x2**

E: Some other value or error

What is in variable g (if it gets created)?

```
d1= 'Mar 13';  d2= 'Mar 29';
x1= d1(5:6);  x2= d2(5:6);
g= x2-x1;
```

A: the string '**16**'

B: the numeric value 16

C: Error in the subtraction operation

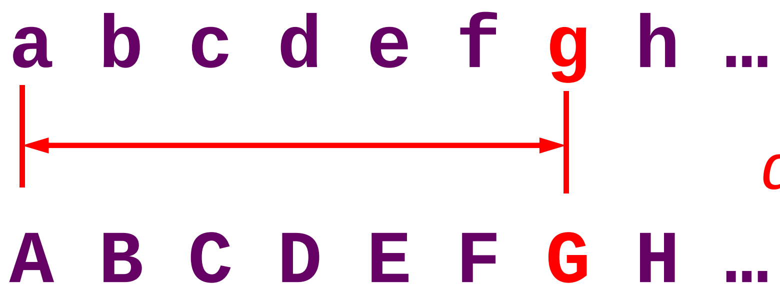
D: Error in assigning variables **x1**, **x2**

E: Some other value or error

Example: toUpper

Write a function toUpper(cha) to convert character cha to upper case if cha is a lower case letter. Return the converted letter. If cha is not a lower case letter, simply return the character cha.

Hint: Think about the distance between a letter and the base letter ‘a’ (or ‘A’). E.g.,



distance = ‘g’-‘a’ = 6 = ‘G’-‘A’

Of course, do not use Matlab function upper!

```
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.
```

```
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.

up= cha;
```

cha is lower case if it is between 'a' and 'z'

```
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.

up= cha;

if ( cha >= 'a' && cha <= 'z' )

    % Find distance of cha from ‘a’



end
```

```
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.

up= cha;

if ( cha >= 'a' && cha <= 'z' )

    % Find distance of cha from ‘a’
    offset= cha - 'a';

    % Go same distance from ‘A’

end
```

```
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.

up= cha;

if ( cha >= 'a' && cha <= 'z' )

    % Find distance of cha from ‘a’
    offset= cha - 'a';

    % Go same distance from ‘A’
    up= char('A' + offset);
end
```