Loops

Loops

- Loops are used to execute a sequence of statements more than once
- We will learn:
  - while loop
  - for loop
- They differ in how the repetition is controlled

Loops: "for" Loop

- Statements are executed a specified number of times
- `for index = array`
- `    statement 1`
- `    statement 2`
- `    ...`
- `end`

statement group

- The statements btw `for` and `end` are executed once for every column in array
- array is usually given in shortcut notation as first:increment:last

Loop Examples

- for x = 1:2:10
  x
  end
- Output:
- x =
  1
  x =
  3
  x =
  5
  x =
  7
  x =
  9

for x = [1 5 13]
  x
end

- Output:

x =
  1
x =
  5
x =
  13

Loop Examples

```
for x = [ 1 2 3; 4 5 6 ]
      x
end
```

- Output:
-
```
x =
     1
     4

x =
     2
     5

x =
     3
     6
```

Loop Examples

Adding numbers 1 through 5

```
sum = 0;
for i = 1:5
    sum = sum + i
end
```

Adding Elements of a Vector

- Given vector v, sum up its contents

```
sum = 0;
for i = 1:length(v)
    sum = sum + v(i)
end
```

Minimum Element of a Vector

```
min = inf;
for i = 1:length(v)
    if (v(i) < min)
            min = v(i)
    end
end
```

Minimum Element of a Vector

What if we need the location as well?

```
min = inf;
index = 0;
for i = 1:length(v)
    if (v(i) < min)
        min = v(i)
        index = i
    end
end
```

Loop Examples

- Example: Factorial (n!) of an integer n
- n = input('Please enter number for which factorial ... to be calculated: ');

Data validation

```
if ( ( n < 0 ) | ( fix(n) ~= n ) )
%check whether n is a positive integer
 error( 'n must be a non-negative integer' );
end
```

```
if ( ( n == 0 ) | ( n == 1 ) ),
    f = 1;
else
    f = 1;
    for i = 2:n
        f = f * i;
    end
end
```

MATLAB Examples

- Find the number of positive numbers in a vector

```
x = input( 'Enter a vector: ' );
count = 0;
for ii = 1:length(x),
    if ( x(ii) > 0 )
        count = count + 1;
    end
end
fprintf('Number of positive numbers is %d\n', count);
```

Loops: "break/continue" Statements

- Break statement terminates the execution of a loop and passes the control to the next statement after the end of the loop
- Continue statement terminates the current pass through the loop and returns control to the top of the loop

Loop Examples

- Example:

```
for i = 1:5,
    if ( i == 3 ),
        break;
    end
    fprintf( 'i = %d\n', i );
end
disp( 'End of loop' );
```

- Output:

```
i = 1
i = 2
End of loop
```

Loop Examples

- Example:

```
for i = 1:5,
    if ( i == 3 ),
        continue;
    end
    fprintf( 'i = %d\n', i );
end
disp( 'End of loop' );
```

- Output:

```
i = 1
i = 2
i = 4
i = 5
End of loop
```

State the output after the execution of the following code segment

```
x=9;
for y=x:-2:0
   if ( (rem(y,3) == 2) & (x<5) )
       break;
   end
  fprintf('y=%d and x=%d\n',y,x);
  x=x-3;
end
fprintf('Final values of x and y  are:%d %d',x,y);
```

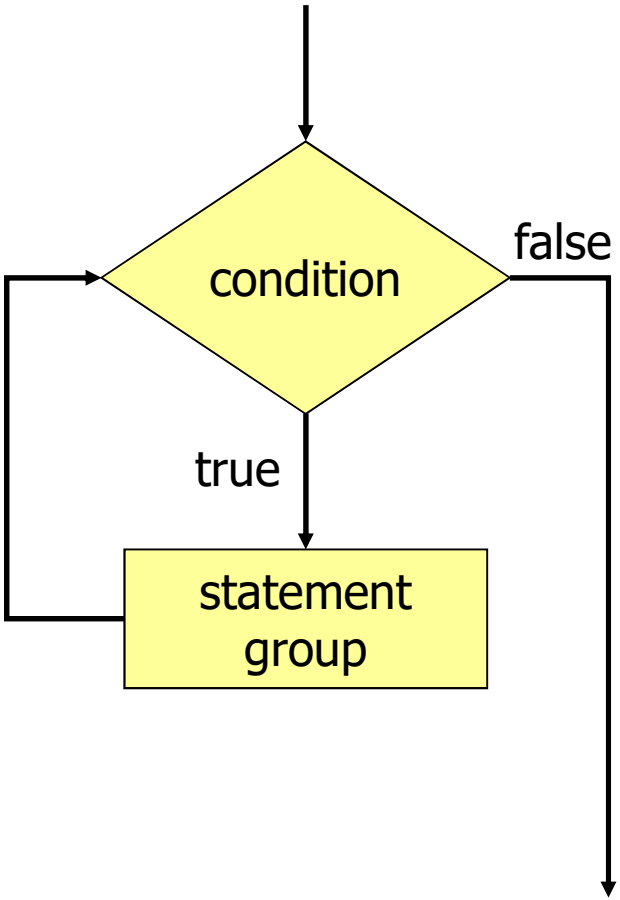
Loop Examples

- Number guessing example: User has only 3 tries
- Pseudocode:
  - Pick a random number, num, in [1,10]
  - for tries = 1:3
    Read user's new guess
    Stop if guess is correct
    end

Loop Examples

```
num = ceil(rand * 10);
for tries = 1:3
    guess = input( 'Your guess?' );
    if ( guess == num )
        disp( 'Congratulations!' );
        break;
    end
end
if ( guess ~= num )
    disp( 'You could not guess correctly' );
end
```

Loops: "while" Loop



- Statements are executed indefinitely as long as the condition is satisfied

```
while ( condition )
    statement 1
    statement 2
    ...
end
```

statement group

While Loops

- a *while* loop evaluates a group of statements an indefinite number of times in contrary to *for* loops which evaluates expressions a fixed number of times

While Loops

- the outline of a typical while loop :
  - perform initializations
  - while (condition)
    - perform the repetitive step
    - update the parameters of condition
  - perform final adjustments

Loop Examples

- Example: Guessing a number computer picks between 1 and 10
- Pseudocode:
  - Pick a random number, num, in [1,10]
  - Read user's guess
  - while (guess ~= num)
    Read user's new guess
    end

Loop Examples

```
num = ceil(rand * 10);
guess = input( 'Your guess?' );
tries = 1;
while ( guess ~= num )
    guess = input( 'Your guess?' );
    tries = tries + 1;
end
fprintf( 'You guessed correctly in %d tries', tries );
```

Initializations

Repetitive step(s)

Loop Examples

- Number guessing example modified:
  User has only 3 tries
  - Pick a random number, num, in [1,10]
  - Read user's guess
  - Initialize number of tries to 1
  - while ((guess ~= num) & (tries < 3))
    Read user's new guess
    Increment number of tries
    end

Number Guessing Game

- What are the reasons we might have gone out of the loop?

The condition must have failed :

guess ~= num and tries < 3

that means, either :

guess == num

or

tries >= 3, meaning, tries == 3

The Matlab Program

```
num = ceil(rand * 10);
guess = input('Try to guess my number ');
tries = 1;
while (guess ~= num && tries < 3)
        guess = input('Try to guess my number ');
        tries = tries + 1;
end
if (guess == num)
    disp('You got it !');
else
    disp('Maybe some other time');
end
```

NESTED LOOPS

A loop within another loop

Loop Examples

- Example: Nested loops
- Multiplication table
- ```
for i = 1:10
   for j = 1:10
      p = i * j;
      fprintf('%d x %d =...
%d\n', i, j, p);
    end
end
```
```

Loop Examples

- Replace those elements of array B greater than 5 by 0.

```
[rB cB]=size(B);
for i=1:rB
      for j=1:cB
         if B(i,j)>5
               B(i,j)=0
         end
      end
end
```

Loop Examples

- How many times **b=b+x*y** is executed? What are the values of x and y after the execution of code segment**?**

```
b=0;
for x=[-1 1 3]
    for y=[2 4 5]
        b=b+x*y;
    end
end
disp(b)
```

State the output after the execution of the following code segment

```
d=[2 3 5;-1 2 0];
s=size(d);
sum=0;
A=zeros(1,3);
for ix=1:s(2)
    for xi=1:s(1)
       sum=sum+d(xi,ix);
    end
    A(ix)=sum;
end
disp(A);
fprintf('ix=%d xi=%d sum=%d\n',ix,xi,sum);
```

MATLAB Examples

- Print a triangle of stars in n rows

```
n = input( 'Enter the number of rows: ' );
for i = 1:n
    for j = 1:i
        fprintf( '*' );
    end
    fprintf( '\n' );
end
```

MATLAB Examples

All Subsets of Size 2

- Given a vector v, list all the possible subsets of elements of v of size 2.

Idea:

For a given element, list all the other elements together with it, effectively listing all 2-element subsets that contain that element

Repeat this for all elements of the vector (set)

First Try

For all elements x of v do

list all the other elements together with x

For all elements x of v do

For all elements y of v except x do

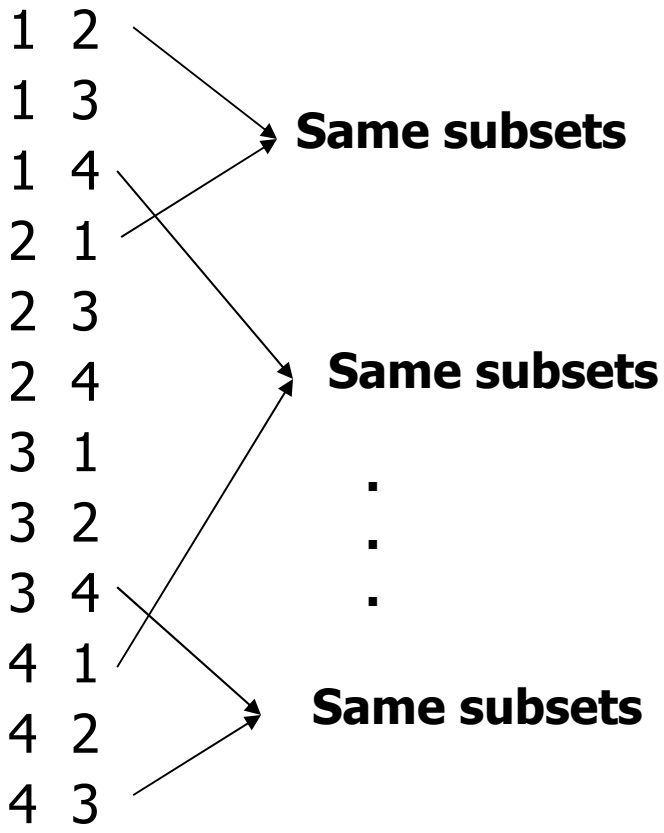
list (x, y)

First Try …

```
for i=1:length(v)
   for j=1:length(v)
        if (i ~= j) % second element is not the same as first one
                disp([v(i) v(j)])
        end
   end
end
```

First Try ...

- For v = 1:4, our program produces:



An idea …

- For two valid index values *k,l,* we list two entries : v(k) v(l) and v(l) v(k)
- One of the listings is extra and not needed
- We can put a new rule to apply before making a listing that will eliminate one of the duplicates

Second Try …

```
for i=1:length(v)
   for j=1:length(v)
        if (i ~= j && i < j)
% second element is not the same as first one and same pairs don't repeat
                disp([v(i) v(j)])
        end
   end
end
```

Eliminate useless iterations

- We don’t need to go over all the i,j combinations
- Only cases where i < j are use

Advice

- Use indentation to improve the readability of your code
- Never modify the value of a loop index inside the loop
- Allocate all arrays used in a loop before executing the loop
- If it is possible to implement a calculation either with a loop or using vectors, always use vectors
- Use built-in MATLAB functions as much as possible instead of reimplementing them

Advice

- Test & debug your code before getting graded/handing it in.
  - Test: Check that your code is running properly. Enter different values to see that it does.
  - Debug: If your code is not running correctly, add some statements to see where you have a problem.
  - Add disp or fprintf statements to see if your program enters a loop, or to see the value of a variable at some point, etc.
- Always hand in your own work!!!