LOOPS:

Loops are useful when you want a certain set of commands executed over and over and over and over and …

There are two types of loops in Matlab/Octave: (A) for loops and (B) while loops.

(A) For loops:

For loops are useful when you know you need to perform a task a certain amount of times.

```
for  index = A:C:B
   commands
end
```

The index variable will take on all the values in the array A:C:B as it goes through the loop. The first value of index will be A.

Often, it is useful to think of A, C, and B in the following terms,

```
for  index = initial value : increment : end value
    commands
end
```

Let’s say the A is 1, C is 2, B is 5.

1:2:7 yields an array of 1 3 5 7

When the for loop begins, the first value of index will be the first array element, 1. After each pass through the loop, the index changes to the value in the subsequent array element. So, after the first pass through the loop, when all the commands between the ‘for’ and ‘end’ are executed, the index changes value to 3 and the commands are re-executed. The loop would go through four passes until it is terminated.

In mfile.m:

```
n = 5;
for i = 1:1:n     % The loop will go through 5 passes
    disp(i)
end
disp(i)
```

At the command line:

```
> mfile
1
2
3
4
5
5
```

In mfile.m:

```
n = 5;
for i = 1:n   % The increment is 1
    disp(i^2)
end
```

At command line:

```
> mfile
1
4
9
16
25
```

In mfile.m:

```
for i = 1:2:5    % example with a non-1 increment
    disp(i^2)
end
```

After execution:

```
1
9
25
```

The variable **i** is 1, then 3, then 5.

You can use the iterator variable **i** to loop through elements of an array.

In mfile.m:

```
for i = 1:5
    a(i) = i^2;
end
disp(a)
```

After execution:

```
    1    4    9   16   25
```

Example: Calculate the average test score without using the mean( ) function.

In mfile.m:
```
score = [76,100,90,99,91];
summy = 0;                  % Notice that we must initialize summy to zero.
for i = 1 : numel(score)
    summy = summy + score(i);
end
average = summy/numel(score)
disp(i)
```

After execution:
```
average =  91.200
5
```

The final value of **i** is 5.

In Matlab/Octave, we have the option of using the **mean( )** function to calculate the average score.

```
score = [76,100,90,99,91];
mean(score)
```

However, many computer languages do not have this handy function and you will need to use a loop.

(B) While loops:

While loops can do anything **for** loops can do, but are more useful when you have a test condition for the loop stopping.

```
while  (test condition)
    commands
end
```

The parentheses **( )** around the test condition are not strictly necessary, but makes your code look nicer.

In mfile.m:
```
x = 0;
while (x<3)
   disp(x)
   x = x + 1;
end
```

After execution:

```
0
1
2
```

The task of summing a known amount of numbers is not a good task for **while** loops since we want to do a task a certain amount of times. A **for** loop would be better in this case. However, you can use a **while** loop if you like.

Example: Keep reading in scores from the user until he is finished.

In mfile.m:

```
answer = input('Do you want to enter another score? (y or n) ', 's');
i = 0;
while(answer=='y')
   i = i + 1;
   value = input('Enter a score: ');
   score(i) = value;
   answer = input('Do you want to enter another score? (y or n) ', 's');
end
average = 0;
for i=1:numel(score)
   average = average + score(i);
end
average = average/numel(score);
disp(['The average score is ' , num2str(average)])
```

After execution:

```
Do you want to enter another score? (y or n) y
Enter a score: 70
Do you want to enter another score? (y or n) y
Enter a score: 80
Do you want to enter another score? (y or n) n
The average score is 75
```