Exercises

1. Write a `for` loop to display N asterisks (*), one on each line, where N is a number given at the beginning of your script. For example, if N is 5, your script should print out:

```
*
*
*
*
*
```

This `for` loop will display N asterisks on N different lines. Compare this with the `while` loop that solves the same problem.

```
for n = 1 : N
   disp('*');
end
```

2. What will be the values of n and counter at the end of the execution of the Octave script given in (a) below?

```
n = 0;
while n ~= 5
    n=input('Enter a number:');
    counter = 2*n;
end
```

```
n = 5
counter = 10
```

The loop will not stop instantly when n reaches 5. It continues till it reaches "end".

3. Write a `for` loop to display N asterisks (*) on one line. Assume that N has already been assigned a number. If N is greater than 80, display only 80 asterisks. If N is less than 1, display no asterisks.

For example, if N is 5, your script should print out

```
*****
```

Here is a fragment that will work if N has previously had its value assigned. Notice how the variable `starline` with N asterisks is created in the loop and it is printed after the loop ends. This is because the `disp()` command prints a new line after its argument.

```
if N > 80
   N = 80;
end
starline = '';
for n = 1 : N
   starline = [starline '*'];
end
disp(starline);
```

4. Write a code fragment that uses a `for` loop to repeatedly prompt the user for positive integer or zero. As long as negative values or non-integers are entered, your loop should ask the user for a positive integer.

Here is a script that will work but has undesireable properties as described following the code.

```
val = -1;
for n = 1 : 100000;
    val = input('Enter a positive integer or zero: ');
    if val >= 0 & rem(val,1) == 0
        break;    % break out of the loop when a valid value is found
    end
end
```

The use of `break` statements is generally discouraged due to the increased difficulty in following the control flow of programs that use `break` statements. Such loops have two (or more) ways to exit or stop the loop, this makes them harder to debug than loops that have only one exit point. Also, the use of the value `100000` or other similarily large value is logically incorrect and this also makes the code harder to maintain. We really don't want this loop to execute a large number or infinite number of times only until the user enters a valid value. Iteration that should repeat until a condition is met (or while a bad condition exists) should be implemented with a `while` loop as shown here:

```
val = -1;
while val < 0 | rem(val,1) ~= 0
    val = input('Enter a positive integer or zero: ');
end
```

5. Write a script that uses a `for` loop to compute and plot the finite Fourier series given by the sum:

$$y(x) = \sum_{n=1}^{1000} \frac{\sin(nx)}{n}$$

```
x = input('Enter x:');
y = 0;            % total starts at 0
% Compute the fournier series for x
for n = 1:1000
    y = y + sin(n*x)/n;
end
```

6. Write a `script` to display all integers from 1 to N, where N is a positive number given at the beginning of your script. Each integer will be displayed on a new line and for numbers which are multiples of 6, the message "This number is a multiple of 6" will also be displayed next to the number.

```
N=input('Enter a number');
for i = 1:N
    if (rem(i,6)==0)
        disp([num2str(i) ' This number is a multiple of 6.']);
    else
        disp(num2str(i));
    end
end
```

7. Write a script to display all the prime numbers from 1 to 100.

```
for i=2:100
   for j=2:100
      if (rem(i,j)==0) % You can also write (mod(i,j)==0)
         break;
      end
   end
   if(j == i)
      disp([num2str(i) ' is prime.']);
   end
end
```