One of the main points of writing code is to do calculations that would be too boring or long to do by hand. In this short tutorial, we introduce the for-loop structure as a way to automate such calculations.

Let's say that I want to multiply all numbers between one and ten by two. I could simply do

\[
\begin{align*}
1*2 \\
\text{ans} &= 2 \\
2*2 \\
\text{ans} &= 4 \\
3*2 \\
\text{ans} &= 6 \\
& \text{and so forth and so on until I get to ten. However, using for-loops we can avoid having to write the same calculation ten times. A for-loop makes it possible to iterate through these calculations. For-loops need a counter: a number that keeps track of how many iterations we've done. Let's start by defining the range we want}
\end{align*}
\]

\[
\begin{align*}
i &= 1:10 \\
\text{i} &= \text{1x10 double} \\
&= \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix}
\end{align*}
\]

Note, that this is a simple way to generate all numbers between one and ten. Now, we invoke a for-loop where in each iteration we go through one value of \( i \)

\[
\begin{align*}
\text{for Counter} &= i \\
\text{Counter*2} \\
\text{end}
\end{align*}
\]

\[
\begin{align*}
\text{ans} &= 2 \\
\text{ans} &= 4 \\
\text{ans} &= 6 \\
\text{ans} &= 8 \\
\text{ans} &= 10 \\
\text{ans} &= 12 \\
\text{ans} &= 14 \\
\text{ans} &= 16 \\
\text{ans} &= 18 \\
\text{ans} &= 20
\end{align*}
\]

Note the different elements of the for-loop. First, we defined a counter that is, in our specific example, a list of numbers that goes from 1 to 10. In each iteration of the for-loop, the variable \( \text{Counter} \) will take one of the values of \( i \). Then, in each one of these cycles, it multiplies the specific value of \( \text{Counter} \) by two. For-loops also need an end statement to tell Matlab where to stop doing iterations. Finally, we could have circumvented the definition of \( i \) by simply doing

\[
\begin{align*}
\text{for} \quad \text{Counter} &= 1:10 \\
\text{Counter*2} \\
\text{end}
\end{align*}
\]
Of course, this is a silly calculation: we could just have done

```
(1:10)*2
```

However, the point is that these for-loops will make it possible to carry out multiple repetitive tasks that will be far more complex than a simple multiplication by two.