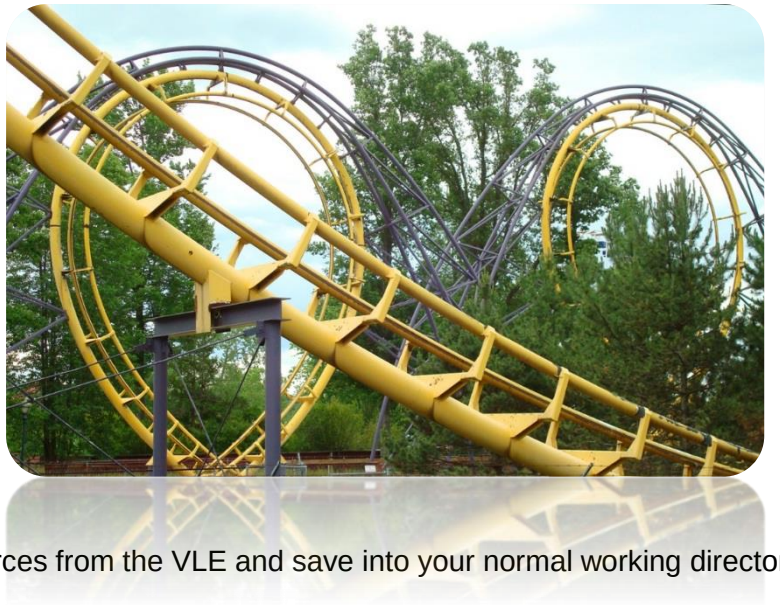


Download all resources from the VLE and save into your normal working directory.

Exercise 3.1 – For loops

In this exercise you'll look at examples of for loops and program a few loops for yourself as well as look at nested for loops.

1. Ensure the loopexample.m file from the VLE is in your working directory and remember to set the current directory in the MATLAB window to the same directory.
   a. Open the m-file in Editor and read through it.

   b. Run the program by either typing loopexample in the Command Window or pressing the run button in editor. Note there are 3 for loops in the code, you will have to press a key between each one.

   c. Does it work as expected? (QG 9)

2. Add a further 3 further for loops after the final pause. These should:
   a. count from 20 to 30 in steps of 3

   b. count down from 30 to 20 in ones (a negative step)

   c. Count from 0 to pi in steps of pi/8

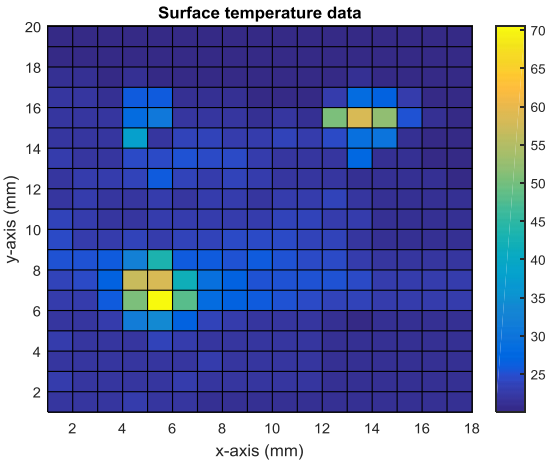
3. Now we are going to try a nested loop structure. Add a Nested loop to the m-file. This is done by adding one for loop inside another for loop.
   a. To do this add the following code to the m-file:

```
disp('Nest loops')
for counter1 =1:5 %this is the start of the first loop
   for counter2 =1:6 %this is the start of the second loop
     disp(['Value of counter 1 (outer loop) ' num2str(counter1) '
     value of counter 2 (inner loop) ' num2str(counter2)]);
   end %this is the end  of the second loop
end %this is the end of the first loop
```

   b. Save the file and run the script, does it do as expected?

   c. Now add some code to the inner for loop that will build a matrix called *result* containing the value of counter1*counter2. You can use *counter1* and *counter2* for indexing. Run your code and check you have created a 5x6 matrix.

Hot Surfaces



4. You are now going to undertake some analysis on some collected temperature data. **surface_sample.mat** contains some temperature data measured across a flat surface with measurements provided for every mm for a 20mm x 18mm sample. We are only interested in areas where temperature has elevated above the room temperature of 24°C. For this task you will use nested for loops to index each sample in turn, checking if the temperature is below 24°C and if so setting the value to zero. Follow the steps to undertake the task:

a. Create a new file called temperature_process.m and add appropriate comments. Remember to clear the command window and workspace.

b. Load surface_sample.mat.into the workspace programmatically (QG 11.1).

c. Find the **size** of the matrix *temperature* and set to a variable called *dimensions*. (QG 3)

d. Now create some nested for loops. The outer loop will be used to index the row data and the inner loop will index the columns. You need to set the outer loop to count from 1 up to the total number of rows and the inner one to count from 1 up to the number of columns. (QG 9)

```
for row =1:??? %this will be used to index the row

      for column =1:??? %this will be used to index the column

         [place your logic for setting temperature to zero when less than
          24deg, you will need to use row and column to index the
          temperature matrix]

      end %this is the end  of the second loop

end %this is the end of the first loop
```

Within the middle for loop, you need to check if the value is less than 24, if it is, you should replace the corresponding value in the temperature matrix with a zero.

e. Include in your loop a method of counting how many points are below 24deg. To do this you'll need to initialise a variable *count* as zero before the **for loop** and increment *count* appropriately inside the loop.

f. Calculate the percentage of the surface area that is at or above room temperature and display the result to the user appropriately.

g. Save your program and run it. Compare your new temperature matrix to the previous one, you should see the majority of values are now zero.

End of exercise 3.1

Exercise 3.2 – While loops

In this exercise you'll look at examples of while loops and program a few loops for yourself. You'll then look at a more complicated program that contains if/elseif/else structures within a while loop.

1. Ensure the whileprogram.m file from the VLE is in your working directory and remember to set the current directory in the MATLAB window to the same directory.
   a. Open the m-file in Editor and read through it. Run the file and notice what happens in the first and second examples.

   b. This snippet repeatedly asks for a number until it lies outside 2 values. Run it as it is - notice it doesn't do what you expect..... Why not? Now change inputnum=0 to inputnum=8 and run again. (QG 7)

   c. Does it work now? Run it a few times using a range of inputs.

   d. Alter the first example so that it repeatedly asks for a number until it lies between the 2 values. You may need to use a different logical operator. (QG 4.2)

   e. Next alter the first example again so that it only gives you 3 chances. Display an appropriate message at the end: either you got between the limits in 3 goes or you didn't! Let the person know how many goes they did it in if they got it right. You need to consider precedence of logical operators, an && will execute before an ||.

      Hint, you'll have to use some of the code from the second example in whileprogram.m.

Guessing Game

2. Now for a bit for of a challenge. Try and complete a guessing game for two players, one player must secretly input a number and the second player must then guess it. The best method of doing this is to combine both a while loop and if/else structures.

   a. Open guess_my_number.m in the Editor

   b. Adapt the program so it will ask a first user to input a number between 0 and 50 (QG 12.1). Then clear the Command Window and allow a second user to guess the number the first user inputted. (Note you'll have to manually close the command history and workspace windows to prevent cheating) If the guess is higher, display a message saying so, if the guess is lower, display a message saying so. (QG 12.2). Count the number of guesses until they get the correct number.

   c. This will take some thinking about. You could also add further logic for if the guesses are getting closer or not by storing the distance between guesses and the answer, how could you implement this? You'll need a number of elseif cases for if the number is greater but getting closer or further away and likewise if less. (QG 8) You'll also need to check if it is the first guess or not.

End of exercise 3.2

Bonus challenge

Shuffle program

Download shuffle.m from the VLE and attempt to rectify the program such that the same song cannot be selected multiple times. You will need to make a change to the logic and undertake some indexing. As this is a challenge, these are the only instructions.