Matlab Exercise 1: Basic Concepts

Note: In the following exercises, the notation `...` or `?` indicates that some output has been omitted. Where the output is expected to be a number, vector, or matrix, you should **first** try to determine what the result will be, noting it on the exercise sheet, **then** execute the command to verify your answer.

1. Start Matlab and check the the version number with the `ver` command when the Command Window appears. This tells you what version of Matlab you are running, on which operating system, and what extension toolboxes are available. Now try `version` which gives a string containing the short version.

   ```
   >> ver
   >> version
   ```

2. Matlab can perform basic calculator operations. Try the following and check the output:

   ```
   >> 3 * 4
   ans =
       12

   >> 20/8
   ans =
       2.5000

   >> 8^2
   ans =
       64

   >> 1 + 3 + 7
   ans =
       11

   >> 2 * 5 + 6
   ans =
       16

   >> 2 * (5 + 6)
   ans =
       22
   ```

   Notice that round brackets have the normal (i.e. highest) precedence rules. The precedence of other operators will be covered later.

3. In the last exercise you saw Matlab print out `ans =` before the result of each operation. This is because the result of the last operation Matlab peforms is automatically assigned to the variable named `ans` if one is not explicitly assigned. You can use this variable in your following operations, and in this way `ans` acts as an accumlator or default variable. Note that it will be overwritten the first time you do an operation which does not explicitly assign the result to a named variable. Try the following operations to see how this works. The next exercise will discuss variables at greater length.

   ```
   >> 5 * 3
   ans =
       15

   >> ans - 6
   ans =
        9

   >> ans / 3
   ans =
        3

   >> temp = 42
   temp =
       42

   >> ans + 10
   ans =
       13

   >> temp = 16 / 4
   temp =
        4
   ```

```
>> ans
ans =
    13
```

4. Variables are essential for referencing data and data sets. There are four main types: scalars, vectors, matrices, and strings. By convention scalar and vector numeric variables are given names starting with a lower case letter, while matrices are given names starting with an upper case letter. There is no particular convention for strings, however when assigning a string to a variable the string must be enclosed in single quotes such as `'string'`. Making use of some helper functions which will be explained more later, try the following commands to create and use different sorts of variables.

```
>> a = 5
a =
     5

>> b = 3
b =
     3

>> a * b
ans =
    15

>> r = rand(1, 3)
r =
    0.0975    0.2785    0.5469

>> M = magic(3)
M =
     8     1     6
     3     5     7
     4     9     2

>> s = 'A simple string'
s =
A simple string
```

Note that Matlab uses double precision floating point values for all numerics, and is a "dynamically typed" language. This means you do not need to distinguish between integer and floating point values, and you can assign a string value to a variable name which previously held a numeric value (or vice versa) at will. Most of the exercises here will use integer values for simplicity, but all the operations are equally valid with floating point (and, for that matter, complex) numbers. More details of the implications of floating point numbers will be discussed later.

5. Another important thing to know about Matlab is that all variable assignments are copy by value, rather than copy by reference. If you aren't sure what this means, look at the following example:

```
>> a = 42;
>> b = a;
>> b = b / 2
b =
    21
>> a
a =
    42
```

In many other programming languages assigning `b = a` would mean the two variable **names** referred to the same in-memory variable **value**, so that changing one value would, in effect, change both.

6. Matrices are entered using square brackets to mark the beginning and end of the matrix, spaces or commas between row items, and a semi-colon to mark the end of a row. Create a 2×5 matrix named "EvenNumbers" containing the even values from 0 to 18.

7. Some important notes about variable names are:

   - They must not begin with a number;
   - They may not contain "-" (dash), since this is interpreted as a minus sign;
   - They may not contain special characters (underscore "_", however, is OK).

   Effectively, this means the best rules for choosing variable names are:

   - Start with an upper or lower case letter (underscore "_" is also possible);
   - Followed by any combination of `[a-z A-Z 0-9 _]`

   Additionally, variables take precedence over functions if the same name is used for both. Above you used

the `rand` function to create a vector of random values. Try the following to see what happens when you create a variable with the same name as the function `rand`:

```
>> rand
ans =
    0.9706

>> rand
ans =
    0.1576

>> rand = 'my random string'
rand =
my random string

>> rand
rand =
my random string
```

Later you will see how to remedy this situation. To avoid confusion you should not create variables with names which match function names.

Try creating some Matlab variables with illegal names to see what error messages you are given.

```
>> final-result = 32
...
>> 2008_03_14_data = [1 2 3]
...
```

8. It is often necessary to find out information about what variables are currently active in memory. This may be to find out the size of the variables, their type (or class), or to see how much memory they are consuming in case the system slows down or you get low memory warnings. Try the following commands to see a list of variables and then to see details about the variables

```
>> who
Your variables are:
M a ans b r rand s temp x

>> whos
...
```

You can remove some of the variables using the `clear` command. Without any arguments `clear` will remove all variables.

```
>> clear rand temp x
Your variables are:
M a ans b r rand s temp x

>> who
Your variables are:
M a ans b r s
```

Notice if you try the `rand` function now it will return a random value again, rather than the string `'my random string'`.

If you try to access a variable which has been cleared (or which was never declared) you will get an error message:

```
>> temp
temp =
     4

>> clear temp

>> temp
??? Undefined function or variable 'temp'.
```

9. Some operations return too much information. In these cases you have two options:

   - complete the operation but don't output anything by terminating the operation with a `;` (semicolon)
   - pause after each screen of output by turning on the pager with `more on`.

Try the following commands to see these in effect:

```
>> a = 18;
>> b = 3;
>> a
a =
    18

>> a;

>> c = a / b;

>> c
c =
     6

>> c = a / 9
c =
     2
```

```
>> rand(100,3)
...
>> rand(100,3);
>> more on
>> rand(100,3)
...
>> more off
>> rand(100,3)
...
```

10. You have already seen two functions which generate values for you: `rand()` and `magic`. Matlab has a rich set of documentation for all its built-in functions and also topical documentation. There are three ways to access this:

    - Quick documentation can be found using `help <function-name>`. Try this for `rand` and `magic`:

      ```
      >> help rand
      ...

      >> help magic
      ```

    - More detailed documentation can be found using `doc <function-name>`. Try this for `rand` and `magic`. It will bring up a graphical browser with examples and hyper-links to more documentation:

      ```
      >> doc rand
      ...

      >> doc magic
      ...
      ```

    - The graphical documentation can also be found through the the *Help→Product Help* menu bar.

    Refer to the course notes for information regarding local support options for Matlab, such as mailing lists and help desk contact details.

11. **Extra:** The `rand` function returns a normally distributed random value. Write a Matlab expression which returns a Gaussian random value with a mean of 80 and standard deviation of 5.5.