MATLAB lesson 5: Programming
Solutions to exercises

Dr. Gerard Capes*

1 Logical operators

Based on the lesson

3.
```
% Q1.3 solution

% clear all variables
clear

% create array with 500 rows
a=rand(500,1);

% Test if any values are greater than 0.5
any(a>0.5)
```

4.
- Create a new script using the `new script` button on the `home` tab, or using the keyboard shortcut ctrl+N
- Type commands into the script window, then save the script, chosing a descriptive name (something which reflects what the script does)
- Make sure the file extension is `.m`
- You can then run the script by typing the name of the script (without the `.m` extension), clicking the `run` button on the `editor` tab, or pressing `F5`.

Using MATLAB's help

5.
```
% Q1.5 solutions

% clear all variables
clear

% create array with 500 rows
a=rand(500,1);

% 1.5 (a)
% Test if any values are greater than 0.5, 0.9, 0.99.
% Output text to explain the following:
disp('The output from any(a>0.5) is')
```

*Questions and feedback can be directed to gerard.capes@manchester.ac.uk

```
disp(any(a>0.5))

disp('The output from any(a>0.9) is')
disp(any(a>0.9))

disp('The output from any(a>0.99) is')
disp(any(a>0.99))

% 1.5 (b)
% Find indices where values are greater than 0.99.
indices=find(a>0.99);
disp('Indices where a>0.99')
disp(indices)

% 1.5 (c)
% Are all values greater than 0.5, 0.1, 0.01?
result=all(a>0.5);
disp('All values are greater than 0.5: true or false?')
disp(result)

result=all(a>0.1);
disp('All values are greater than 0.1: true or false?')
disp(result)

result=all(a>0.01);
disp('All values are greater than 0.01: true or false?')
disp(result)

% 1.5 (d)
indices=find(a>0.99);
a(indices)=1
```

6.
```
% 1.6 - repeat for 10*10 matrix
clear
a=rand(10,10);

% 1.6 (a)
% Test if any values are greater than 0.5, 0.9, 0.99.
% Output text to explain the following:
disp('The output from any(a(:)>0.5) is')
disp(any(a(:)>0.5))

disp('The output from any(a(:)>0.9) is')
disp(any(a(:)>0.9))

disp('The output from any(a(:)>0.99) is')
disp(any(a(:)>0.99))

% 1.6 (b)
% Find indices where values are greater than 0.99.
indices=find(a(:)>0.99);
```

```
disp('Indices where a(:)>0.99')
disp(indices)

% Find row and column indices (more useful)
[row_indices,col_indices]=find(a>0.99);
disp('row_indices')
disp(row_indices)
disp('col_indices')
disp(col_indices)


% 1.6 (c)
% Are all values greater than 0.5, 0.1, 0.01?
result=all(a(:)>0.5);
disp('All values are greater than 0.5: true or false?')
disp(result)

result=all(a(:)>0.1);
disp('All values are greater than 0.1: true or false?')
disp(result)

result=all(a(:)>0.01);
disp('All values are greater than 0.01: true or false?')
disp(result)

% 1.6 (d)
a(a>0.99)=1
```

7.
```
% 1.7
% Clear all variable
clear

% Create two 5*5 matrices of random numbers
r1=rand(5)
r2=rand(5)

% 1.7 (a)
% Create logical matrix showing which values are greater in r1 than r2
greaterthan = r1 > r2

% 1.7 (b)
resulta = r1 > 0.5
resultb = r1 > 0.9
resultc = r1 > 0.99
```

2 Flow control

Based on the lesson

1.
```
% 2.1
```

```
% Create a variable x and assign it a value
x=5;

% 2.1 (a)
if x > 1 && x < 2
    disp('1 < x < 2')
end

% 2.1 (b)
if x > 1 && x < 2
    disp('1 < x < 2')
elseif x <= 1
    disp('x is less than or equal to one')
end

% 2.1 (c)
if x > 1 && x < 2
    disp('1 < x < 2')
elseif x <= 1
    disp('x is less than or equal to one')
else
    disp('x is greater than or equal to 2')
end

% 2.1 (d)
x=0.5; % Then run the script from 2.1 (c)
% This tests x<=1 condition

x=1; % Then run the script from 2.1 (c)
% This tests the boundary condition for x <=1

x=1.3 % Then run the script from 2.1 (c)
% This tests the condition x > 1 && x < 2

x=2 % Then run the script from 2.1 (c)
% This tests the "else" condition
```

2.
```
% 2.2 (a)

if isa(A,'double')
    disp('A is double precision')
end

% 2.2 (b)
if isa(A,'double')
    disp('A is double precision')
elseif isa(A,'char')
    disp('A is a character')
elseif isa(A,'logical')
    disp('A is a logical')
end
```

```
% 2.2 (c)
if isa(class(A),'double')
    disp('A is double precision')
elseif isa(A,'char')
    disp('A is a character')
elseif isa(A,'logical')
    disp('A is a logical')
else
    disp('Unknown class')
end

% 2.2 (d)
% (Re-)run the script from 2.2 (c) after each of the following
   commands
A=true;
A=1.1;
A='Test';
A=single(8);
```

3.
```
% 2.3

switch class(A)
    case 'double'
        disp('A is double precision')
    case 'char'
        disp('A is character')
    case 'logical'
        disp('A is logical')
    otherwise
        disp('Unknown class')
end
```

4. In a `switch`, each `case` is looking for a match (i.e. testing the switch variable for a match to the value in each `case`). When testing an expression, you should use an `if` instead. A `switch` is best reserved for cases when you expect the result to be one of a discrete (non-continuous) series of values.

5.
```
% 2.5

% 2.5 (a)
% A loop that counts from 1 to 10
for c1 = 1:10
    fprintf('c1=%d\n', c1)

    % 2.5 (b)
    % A nested loop that counts from 10 to 1
    for c2 = 10:-1:1
        fprintf('c2=%d\n', c2)

        % 2.5 (c)
        % Exit inner loop when c1 is equal to c2
```

```
        if c1==c2
            break
        end
    end

    % Exit outer loop when c1 is equal to c2
    if c1==c2
        break
    end
end
```

6.
```
% 2.6

% Initialise variables
B = 1;
c = 0; % c is our counter

% Our loop
while B ~= Inf
    B = B * 10;
    c = c + 1;
end

% Print to screen the number of iterations
fprintf('Number of iterations = %d\n', c)
```

7. Generally speaking, you should use a `for` loop when you know in advance how many iterations you will require. If you don't know how many iterations you will need, you can use a `while` loop to iterate for as long as a condition remains true.

8.
```
% 2.8 (a)


% Initialise variables
% The first 2 variables are for use in the fprintf statement at the
   end
first=1;
last=100;
count=0;

% Loop from 1 to 100
for i=first:1:last % This would normally be "for i=1:1:100"
    % Test if the loop counter is divisbile by both 5 and 7
    if mod(i,5)==0 && mod(i,7)==0
        fprintf('%i is divisible by both 5 and 7\n',i)
        % 2.8 (b)
        % Count iterations
        count=count+1;
    end
end
```

```
% Print number of iterations
fprintf('There are %i integers between %i and %i that are divisible by
    5 and 7\n',count,first,last)
```

9.
```
% 2.9
% While loop to calculate first 10 numbers divisible by 3, 4, and 5.

% Counter to record how many numbers have met our criteria
count=0;
number=1; % This could start from 3*4*5=60 because this will be the
   first result.
while count<10
    % Test if number meets the criteria
    if mod(number,3)==0 && mod(number,4)==0 && mod(number,5)==0
        % Print text to screen when a result is found
        fprintf('%i is divisible by 3, 4 and 5.\n',number)
        % Increment counter for each result
        count=count+1;
    end
    % Increment number
    number=number+1;
end
```

10. Using a for loop

```
% 2.10 (a) - using a for loop
% Test for prime numbers

% Get user input
prompt='Enter an integer greater than 1: \n';
number=input(prompt);

% 2.10 (b)
% Check the number is greater than 1
assert(number>1,'Number must be greater than 1')

% 2.10 (b)
% Check the number is an integer
assert(mod(number,1)==0,'Number must be an integer')

% Initialise the prime flag as 1 (true)
prime=1;

% Loop through all numbers required to determine result
for i=2:1:number/2
    % Test if number isn't prime
    if mod(number,i)==0
        % Set the prime flag to false
        prime=0;
        % No need to continue the loop now the result is known
        break
    end
```

```
end

% Print result and explanation
if prime
    fprintf('%d is a prime number\n',number)
else
    fprintf('%d is not a prime number\n',number)
    fprintf('%d is divisible by %d\n',number,i)
end
```

Using a while loop

```
% 2.10 (a) - using a while loop
% Test for prime numbers

% Get user input
prompt='Enter a number: \n';
number=input(prompt);

% 2.10 (b)
% Check the number is greater than 1
assert(number>1,'Number must be greater than 1')

% 2.10 (b)
% Check the number is an integer
assert(mod(number,1)==0,'Number must be an integer')

% Initialise the prime flag as 1 (true)
prime=1;

% Loop through all numbers required to determine result
% Flag to determine whether loop should continue
i=2;
while prime && i<number/2
    % Test if the number is not prime
    if mod(number,i)==0
        prime=0;
    end
    i=i+1;
end

% Print result and explanation
if prime
    fprintf('%d is a prime number\n',number)
else
    fprintf('%d is not a prime number\n',number)
    fprintf('%d is divisible by %d\n',number,i)
end
```