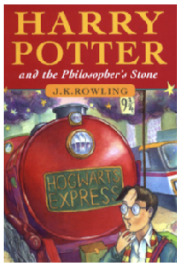
Reading External Text Files

Text files are *human readable* files that store information that may be textual or numerical. They have a file name extension of **.txt** and can be created in the MATLAB editor or in other applications

Consider the **harryPotter.txt** file that contains this content:

> Mr and Mrs Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense.



To read the content of a text file into the local workspace, we open the file with **fopen**, read its content with **textscan**, and close the file with **fclose**

```
% read the contents of harryPotter.txt into the workspace

fid = fopen('harryPotter.txt');
words = textscan(fid, '%s');
fclose(fid);
```

fopen returns a numerical ID referred to as a *file identifier*

textscan returns a cell array containing the content of the text file. In the above case, the cell array words contains a single cell array nested inside it, with the words of the text, including the punctuation, arranged in a column:

```
words = { {'Mr'
           'and'
           'Mrs'
           'Dursley, '
           'of'
           'number'
           'four, '
           ...
           'such'
           'nonsense. '} }
```

```
% view the content of the cell array contained inside words

words{1}

% view single words in the inner cell array

words{1}{8}

words{1}{37}
```

The second input to the textscan function is a ***format specifier*** that indicates the expected format of the information being read (e.g. string, integer, floating point number, etc.)

The specifier '%s' indicates that we expect to read strings of text, but *what is a string?* a single word? a full line of text?

This is determined by the 'Delimiter' property - by default, strings are separated by spaces and line breaks, but we can restrict this to line breaks as in the following example:

```
% read the text file using only line breaks as the delimiter

fid = fopen('harryPotter.txt');
words = textscan(fid, '%s', 'Delimiter', '\n');
fclose(fid);

% print out the contents of the inner cell array in words

words{1}
```

Suppose we have a text file **elements.txt** that contains a mix of numerical data and strings, in a ***fixed format***

```
1  hydrogen   H    1.01
2  helium     He   4.00
3  lithium    Li   6.94
4  beryllium  Be   9.01
5  boron      B   10.81
6  carbon     C   12.01
7  nitrogen   N   14.01
8  oxygen     O   16.00
9  fluorine   F   19.00
10 neon       Ne  20.18
```

The format string can incorporate other types:

```
fid = fopen('elements.txt');
elements = textscan(fid,'%u %s %s %f');
fclose(fid);
```

format string:
%u integer
%s string
%f float

```
% read in information about elements from the elements.txt file

fid = fopen('elements.txt');
elements = textscan(fid, '%u %s %s %f');
fclose(fid);

elements

% create new variables assigned to the components of elements

atomNums = elements{1}

names = elements{2}

symbols = elements{3}

masses = elements{4}
```

Sometimes life is not so simple...

Suppose we want to compute the total value of a toy inventory stored in a text file **toys.txt** with the following format:

name	price	quantity
mr. potato head	$3.29	80
slinky	$1.29	120
hoola hoop	$2.19	60
monopoly	$3.89	50

Let's explore a solution in the **computeToyValue.m** script...

```
% read the text file of information about the toy inventory

fid = fopen('toys.txt');
tokens = textscan(fid, '%s', 'HeaderLines', 1);
fclose(fid);

tokens{1}
```

When reading the above text file, we used the 'HeaderLines' property to skip the first line of the file. Suppose we want to *preserve* the headers instead. **textscan** has an optional third input that applies the format specification **N** times:

```
% open the file, read the three header strings, then read
% the rest of the contents into the tokens variable

fid = fopen('toys.txt');
headers = textscan(fid, '%s', 3);
tokens = textscan(fid, '%s');
fclose(fid);

headers{1}

tokens{1}
```

Writing External Text Files

Suppose we have data from our program that we want to store in an external text file that is *human readable*

```
% information about 10 chemical elements

atomNums = [1 2 3 4 5 6 7 8 9 10];

names = {'hydrogen' 'helium' 'lithium' 'beryllium' 'boron' ...
         'carbon' 'nitrogen' 'oxygen' 'fluorine' 'neon'};

symbols = {'H' 'He' 'Li' 'Be' 'B' 'C' 'N' 'O' 'F' 'Ne'};

masses = [1.01 4.0 6.94 9.01 10.81 12.01 14.01 16.0 19.0 20.18];
```

```
1  hydrogen   H    1.01
2  helium     He   4.00
3  lithium    Li   6.94
4  beryllium  Be   9.01
5  boron      B   10.81
6  carbon     C   12.01
7  nitrogen   N   14.01
8  oxygen     O   16.00
9  fluorine   F   19.00
10 neon       Ne  20.18
```

```
% print information about the elements in the Command Window
% with fprintf that takes a format specifier and values to
% insert in the individual format strings

for i = 1:length(atomNums)
    fprintf('%u %s %s %f \n', atomNums(i), names{i}, ...
            symbols{i}, masses(i));
end
```

```
% print information about the elements with explicit
% spacing in the format specifier

for i = 1:length(atomNums)
    fprintf('%3u %10s %4s %5.2f \n', atomNums(i), names{i}, ...
            symbols{i}, masses(i));
end
```

```
% left-justify the content in the first 3 columns

for i = 1:length(atomNums)
    fprintf('%-3u %-10s %-4s %5.2f \n', atomNums(i), names{i}, ...
            symbols{i}, masses(i));
end
```

Writing to a Text File

To write information to a text file:

(1) Open file for writing with **fopen**

(2) Write text to file with **fprintf**

(2) Close file with **fclose**

```
% write information about elements to a text file

fid = fopen('elementsNew.txt', 'w');
for i = 1:length(atomNums)
    fprintf(fid, '%-3u %-10s %-4s %5.2f \n', atomNums(i), ...
            names{i}, symbols{i}, masses(i));
end
fclose(fid);
```

A vector of numbers can be written to a file all at once:

```
xdata = [1.0  3.2  7.4  8.7  9.1];
ydata = [32.8  21.9  17.6  29.2  30.4];
results = [1.09  2.13  3.48  2.87  0.98];
fid = fopen('results.txt', 'w');
fprintf(fid, 'experimental results:');
fprintf(fid, '\nxdata: ');
fprintf(fid, '%6.2f', xdata);
fprintf(fid, '\nydata: ');
fprintf(fid, '%6.2f', ydata);
fprintf(fid, '\nresults: ');
fprintf(fid, '%6.2f', results);
fclose(fid);
```

results.txt

```
experimental results:
xdata:   1.00   3.20   7.40   8.70   9.10
ydata:   32.80 21.90 17.60 29.20 30.40
results:   1.09   2.13   3.48   2.87   0.98
```

```
% include literal strings within the format string

patient = {'KJD' 'GAV' 'LLJ' 'RDF' 'YKS' 'EFP'};

systolic = [120 142 117 158 137 125];

diastolic = [80 88 75 92 83 78];

fid = fopen('BP.txt', 'w');
for i = 1:length(patient)
    fprintf(fid, 'patient: %-5s systolic: %-5u diastolic: %-5u \n', ...
            patient{i}, systolic(i), diastolic(i));
end
fclose(fid);
```