# Storing it away for safe keeping

## Reading and writing text files

**CS112 Scientific Computation**
Department of Computer Science
Wellesley College

## Reading a text file into a cell array

Information can be stored in text files, with a .txt extension

mobydick.txt:

> Call me Ishmael. Some years ago – never mind how long precisely – having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen and regulating the circulation…

The contents of a text file can be read into a cell array using textscan:

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

→ { {'Call' 'me' 'Ishmael.' 'Some' … } }

(arranged in a column)

```
fid = fopen('mobydick.txt');
lines = textscan(fid, '%s', 'delimiter', '\n');
fclose(fid);
```

→ { {'Call me Ishmael. Some years ago - … having'
'little or no money in my purse, and … me on shore,'
'I thought I would sail about a little … world. It' … } }

# Reading formatted text with textscan

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
...
```

3

# Reading formatted data

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

Recall our text file elements.txt with numerical data & strings:

```
>> atomNums = elements{1}'
      1   2   3   4   5   6   7   8   9   10  ...
>> names = elements{2}'
  'hydrogen'  'helium'  'lithium'  'beryllium'  'boron'  'carbon'  ...
>> symbols = elements{3}'
  'H'  'He'  'Li'  'Be'  'B'  'C'  'N'  'O'  'F'  'Ne'  ...
>> masses = elements{4}'
   1.0100   4.0000   6.9400   9.0100   10.8100   12.0100   14.0100  ...
```

' is the transpose operator

4

## Sometimes life is not so simple…

Suppose we want to compute total value of our toy inventory, from 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 |

```
fid = fopen('inventory.txt');
tokens = textscan(fid, '%s', 'headerlines', 1);
tokens = tokens{1};
fclose(fid);
totalValue = 0.0;
for index = 1:length(tokens)
    token = tokens{index};
    …
end
disp(['total: $' num2str(totalValue)])
```

5

## *vice versa:* Writing formatted data

Suppose you have data in the MATLAB workspace that you want to store in a text file in a desired format

```
>> atomNums = [1 2 3 4 …];
>> names = {'hydrogen' 'helium' 'lithium' 'beryllium' …};
>> symbols = {'H' 'He' 'Li' 'Be' …};
>> masses =[1.01 4.00 6.94 9.01 …];
```

↓

```
1 hydrogen H 1.01
2 helium He 4.00
3 lithium Li 6.94
4 beryllium Be 9.01
…
```

elements.txt

**Exercise:** write a loop that prints these lines of data

6

## Formatting strings with sprintf

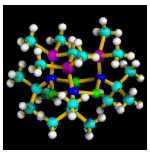
```
>> sprintf( '%u %s %s %f' , atomNums(1), names{1}, symbols{1}, masses(1))
ans =
1 hydrogen H 1.010000
```

```
for i = 1:4
    disp(sprintf('%u %s %s %f',  atomNums(i), names{i}, symbols{i}, masses(i)))
end
```

```
1 hydrogen H 1.010000
2 helium He 4.000000
3 lithium Li 6.940000
4 beryllium Be 9.010000
```

```
for i = 1:4
    disp(sprintf('%4u %12s %4s %8.2f',  atomNums(i), names{i}, symbols{i}, ...
                 masses(i)))
end
```

```
   1    hydrogen    H    1.01
   2      helium   He    4.00
   3     lithium   Li    6.94
   4   beryllium   Be    9.01
```



7

## Finally… Writing data to a text file

(1) Open file for writing
(2) Write text to file
(3) Close file

```
fid = fopen('elements.txt', 'w') ;
for i = 1:length(atomNums)
    fprintf(fid, '%4u %12s %4s %8.2f \n', atomNums(i), ...
            names{i}, symbols{i}, masses(i));
end
fclose(fid);
```

8

# Write a vector of numbers all at once

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
```

9

# Writing files with literal strings

```
data = {{'kayla'   5   6.2   85.7} ...
        {'emilia'  3   5.5   89.8} ...
        {'malai'   7   3.4   92.4} ...

        ... };
```

→ cs112.txt

```
Data for CS112 assignment work
Spring 2018

kayla
used drop-in 5 times
average assignment time 6.2 hours
assignment grade 85.7

emilia
used drop-in 3 times
average assignment time 5.5 hours
assignment grade 89.8
...
```

```
fid = fopen('cs112.txt',  'w');
fprintf(fid, 'Data for CS112 assignment work \nSpring 2018 \n\n');
for i = 1:length(data)
   fprintf(fid, ...
    '%s\nused drop-in %u times\naverage assignment time %3.1f hours\nassignment grade %4.1f\n\n', ...
     data{i}{1}, data{i}{2}, data{i}{3}, data{i}{4});
end
fclose(fid);
```

10

# Other file formats

MATLAB also supports a variety of industry standard formats and custom file formats

This allows MATLAB to exchange data with other programs

Text: MAT, CSV, DLM, TAB

Scientific data: CDF, FITS, HDF

Spreadsheet: XLS, WK1

The urlread function reads in the html code for an input URL into a string

11