Morteza Kazemi | 97243054
Amir Masoud Shaker | 97243081
12/19/2021

Homework #8

Q1

Syntax	Meaning	Description
1) fileID = fopen(filename) 2) fileID = fopen(filename,permission) 3) fileID = fopen(filename,permission,machinefmt,encodingIn) 4) [fileID,errmsg] = fopen(___) 5) fIDs = fopen('all') 6) filename = fopen(fileID) 7) [filename,permission,machinefmt,encodingOut] = fopen(fileID)	Open file, or obtain information about open files	1) This syntax opens the file, filename, for binary read access, and returns an integer file identifier equal to or greater than 3. If fopen cannot open the file, then fileID is -1. 2) This syntax opens the file with the type of access specified by permission.

		3) This syntax additionally specifies the order for reading or writing bytes or bits in the file using the machinefmt argument. The optional encodingIn argument specifies the character encoding scheme associated with the file. 4) This syntax additionally returns a system-dependent error message if fopen fails to open the file. Otherwise, errmsg is an empty character vector.

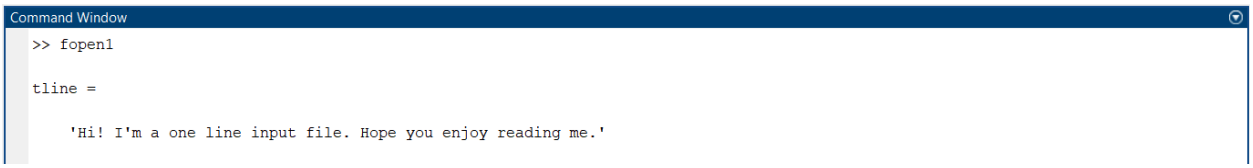
		5) This syntax returns a row vector containing the file identifiers of all open files. The identifiers reserved for standard input, output, and error are not included. 6) This syntax returns the file name that a previous call to fopen used when it opened the file specified by fileID. The output filename is resolved to the full path.

		7) This syntax additionally returns the permission, machine format, and encoding that a previous call to fopen used when it opened the specified file. If the file was opened in binary mode, permission includes the letter 'b'. The encodingOut output is a standard encoding scheme name. fopen does not read information from the file to determine these output values. An invalid fileID returns empty character vectors for all output arguments.

Examples:

Fopen1:

```
fileID = fopen('one_line_input.txt');
% open the file one_line_input.txt
tline = fgetl(fileID)
% pass the fileID to the fgetl function to read one line from the file
fclose(fileID);
% close the file
```



Fopen2:

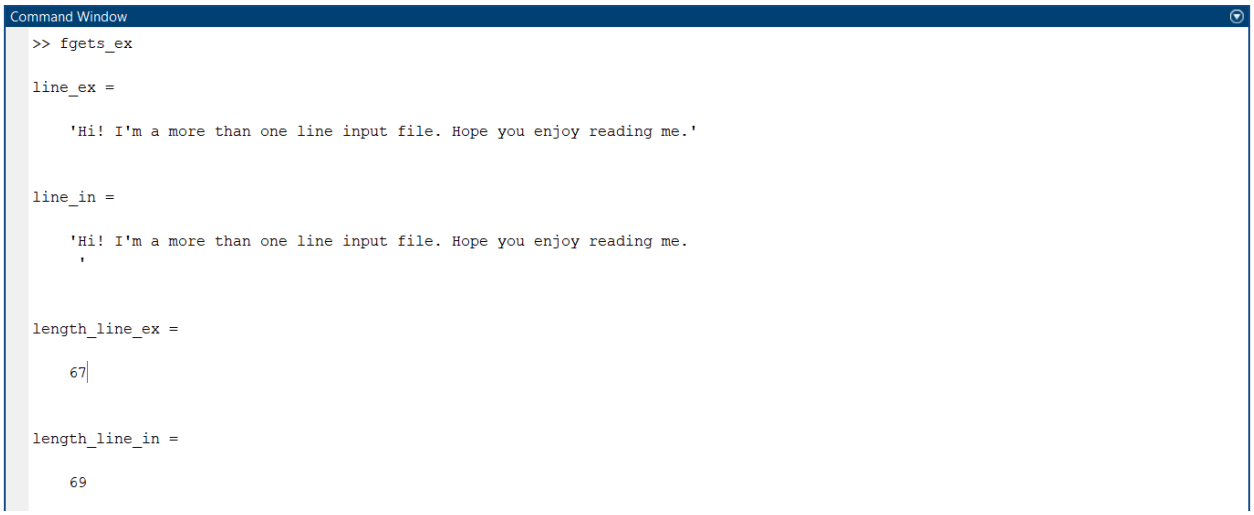
```
fileID = fopen('one_line_input.txt');
% open the file one_line_input.txt
fIDs = fopen('all')
% get the file identifiers of all open files
[filename,~,~,encoding] = fopen(fileID)
% get the file name and character encoding for the open file
```

Syntax	Meaning	Description
1) tline = fgets(fileID) 2) tline = fgets(fileID,nchar) 3) [tline,ltout] = fgets(___)	Read line from file, keeping newline characters	1) This syntax reads the next line of the specified file, including the newline characters. 2) This syntax returns up to nchar characters of the next line. 3) This syntax also returns the line terminators, if any, in ltout.

Example:

Fgets_ex:

```
fileID = fopen('more_than_one_line_input.txt');
% open the file more_than_one_line_input.txt
line_ex = fgetl(fileID)
% read line excluding newline character
frewind(fileID);
% reset the read position indicator back to the beginning of the file
line_in = fgets(fileID)
% read line including newline character
length_line_ex = length(line_ex)
% length of line_ex
length_line_in = length(line_in)
% length of line_in
```



Syntax	Meaning	Description
1) fclose(fileID) 2) fclose('all') 3) status = fclose(___)	Close one or all open files	1) This syntax closes an open file. 2) This syntax closes all open files. 3) This syntax returns a status of 0 when the close operation is successful. Otherwise, it returns -1.

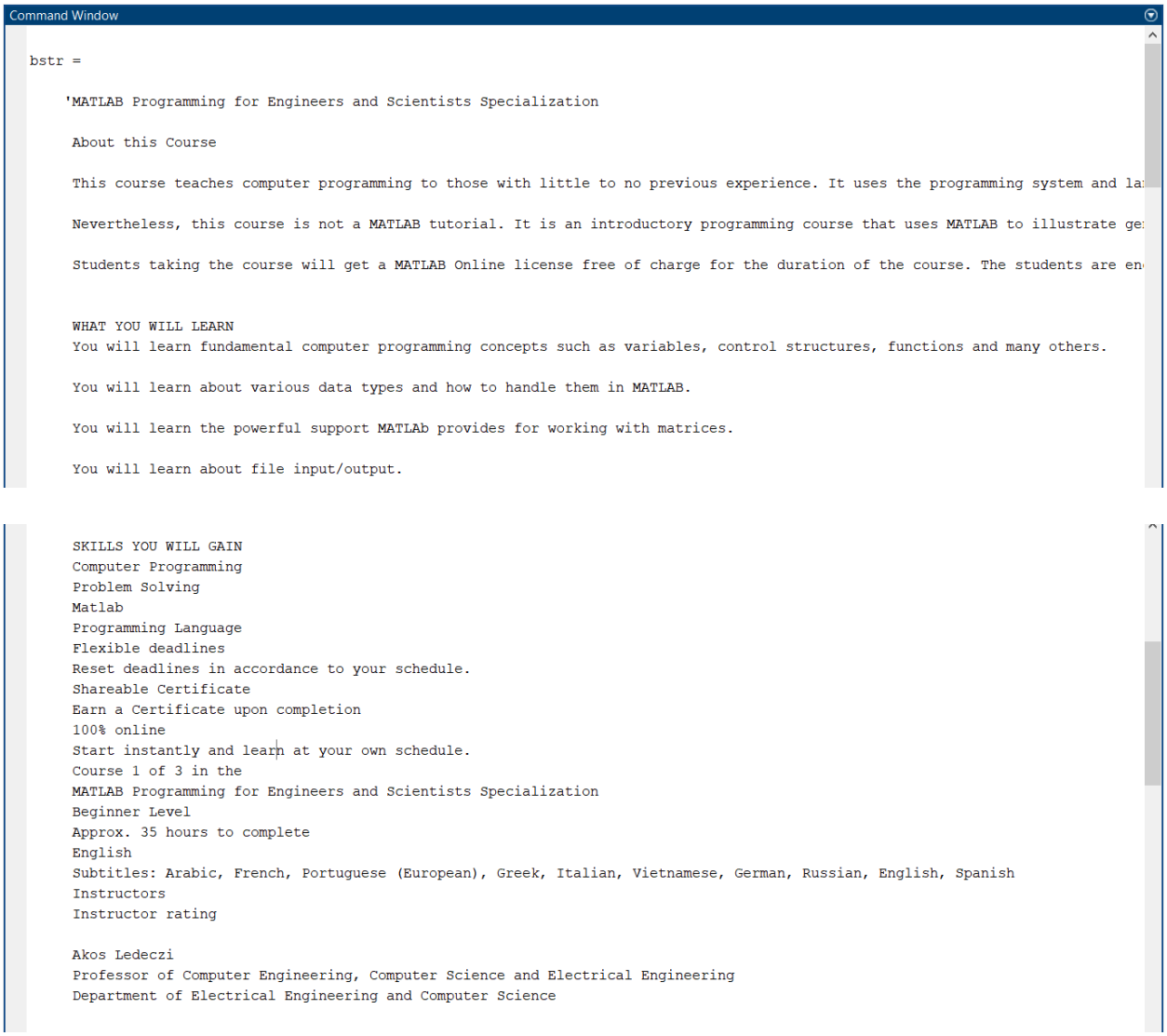
Example:

Fclose_ex:

```
fileID = fopen('hello.txt');
% open the file hello.txt
fgetl(fileID)
% read the first line of hello.txt
fclose(fileID);
% close the file
```

Read_the_bench_file_given:

```
clear; close all; clc;
fid = fopen('bench.txt','r');
bstr=[];
tline = fgets(fid);
while ischar(tline)
 bstr=[bstr,tline];
 tline = fgets(fid);
end
fclose(fid);
bstr,
```



2 Courses
Mike Fitzpatrick
Professor Emeritus of Computer Science, Computer Engineering, Electrical Engineering, Neurosurgery, and Radiology
Electrical Engineering & Computer Science, Neurological Surgery, Radiology & Radiological Sciences

2 Courses
Offered by
Vanderbilt University logo
Vanderbilt University
Vanderbilt University, located in Nashville, Tenn., is a private research university and medical center offering a full-range

Other courses in this Specialization
Syllabus - What you will learn from this course

WEEK 1
1 hour to complete
Course Pages
1 video (Total 2 min), 3 readings
WEEK 2
3 hours to complete
The MATLAB Environment
We will learn how to start MATLAB and will familiarize ourselves with its user interface. We will learn how to use MATLAB as a

7 videos (Total 132 min), 1 reading, 2 quizzes
WEEK 3
3 hours to complete
Matrices and Operators
The basic unit with which we work in MATLAB is the matrix. We solve problems by manipulating matrices, and operators are the p

6 videos (Total 82 min), 1 reading, 4 quizzes
WEEK 4
2 hours to complete
Functions
Functions let us break up complex problems into smaller, more manageable parts. We will learn how functions let us create reus

7 videos (Total 50 min), 1 reading, 3 quizzes
WEEK 5
3 hours to complete
Programmer's Toolbox
MATLAB has useful built-in functions and we will explore many of them in this section. We will learn about polymorphism and ho

5 videos (Total 83 min), 1 reading, 2 quizzes
WEEK 6
5 hours to complete
Selection
Selection is the means by which MATLAB makes decisions about the order in which it executes its statements. We will learn how

```
Programmer's Toolbox
MATLAB has useful built-in functions and we will explore many of them in this section. We will learn about polymorphism and ho

5 videos (Total 83 min), 1 reading, 2 quizzes
WEEK 6
5 hours to complete
Selection
Selection is the means by which MATLAB makes decisions about the order in which it executes its statements. We will learn how

7 videos (Total 80 min), 2 readings, 4 quizzes
WEEK 7
7 hours to complete
Loops
Loops give computers their power. We will learn how to use both of MATLAB's loop constructs: the for-loop and the while-loop.

5 videos (Total 133 min), 2 readings, 4 quizzes
WEEK 8
6 hours to complete
Data Types
Computers operate on bits, but humans think in terms of numbers, words, and other types of data. Like any good language, MATLA

6 videos (Total 194 min), 3 readings, 2 quizzes
WEEK 9
7 hours to complete
File Input/Output
Files are named areas in permanent memory for storing data that can be used as input or output to MATLAB and to other programs

4 videos (Total 75 min), 5 readings, 5 quizzes
'
```

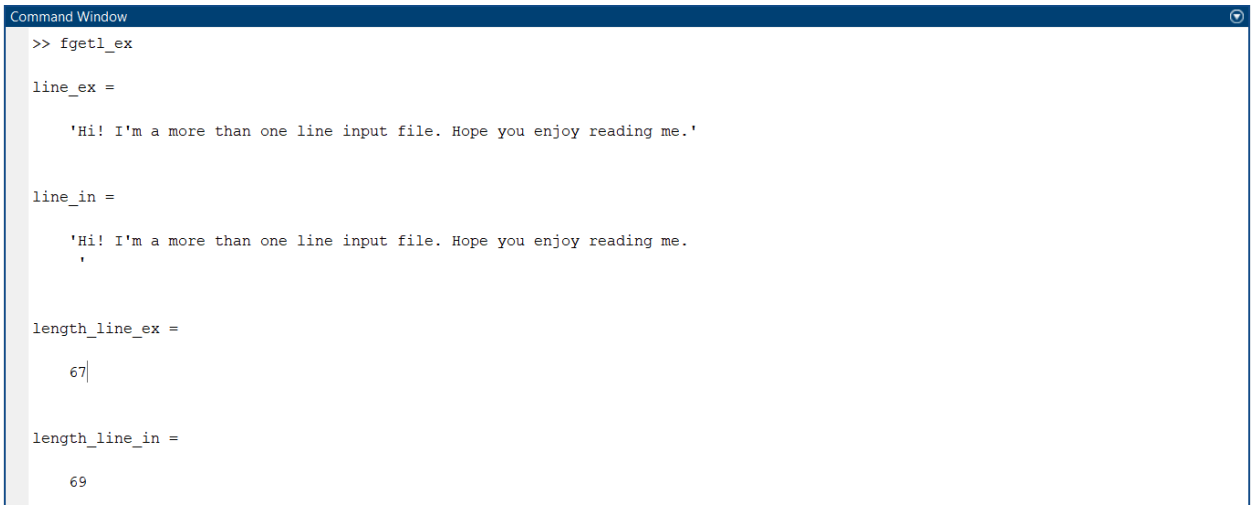
Q2

Syntax	Meaning	Description
1) tline = fgetl(fileID)	Read line from file, removing newline characters	1) This function returns the next line of the specified file, removing the newline characters. If the file is nonempty, then fgetl returns tline as a character vector. If the file is empty and contains only the end-of-file marker, then fgetl returns tline as a numeric value -1.

Example:

Fgetl_ex:

```
fileID = fopen('more_than_one_line_input.txt');
% open the file more_than_one_line_input.txt
line_ex = fgetl(fileID)
% read line excluding newline character
frewind(fileID);
% reset the read position indicator back to the beginning of the file
line_in = fgets(fileID)
% read line including newline character
length_line_ex = length(line_ex)
% length of line_ex
length_line_in = length(line_in)
% length of line_in
```



Syntax	Meaning	Description
1) c = newline	Create newline character	1) This function creates a newline character. newline is equivalent to char(10) or sprintf('\n').

Example:

Newline_ex:

```
chr = 'I am the first line';
chr = [chr newline 'And I guess I am the second line']
```

```
Command Window
>> newline_ex

chr =

    'I am the first line
     And I guess I am the second line'
```

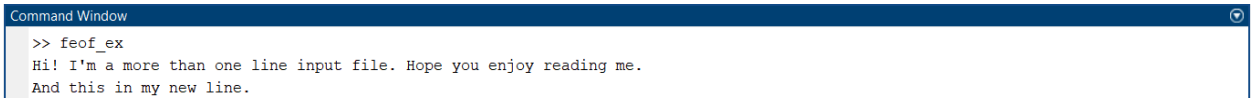
Q3

Syntax	Meaning	Description
1) status = feof(fileID)	Test for end of file	1) This function returns the status of the end-of-file indicator. The feof function returns a 1 if a previous operation set the end-of-file indicator for the specified file. Otherwise, feof returns a 0.

Example:

Feof_ex:

```
fileID = fopen('more_than_one_line_input.txt');
% open the file more_than_one_line_input.txt
while ~feof(fileID)
    tline = fgetl(fileID);
    disp(tline)
end
% read and display one line at a time until you reach the end of the file
fclose(fileID);
% close the file
```

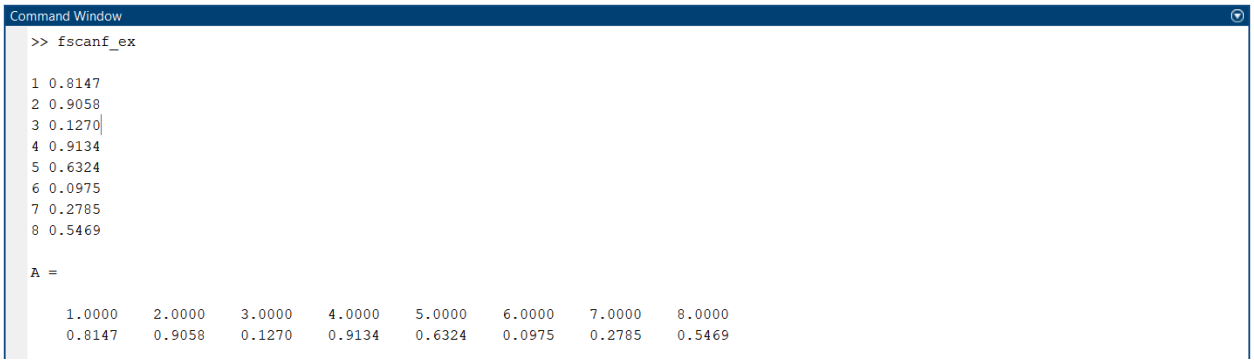


Syntax	Meaning	Description
1) A = fscanf(fileID,formatSpec) 2) A = fscanf(fileID,formatSpec,sizeA) 3) [A,count] = fscanf(___)	Read data from text file	1) This syntax reads data from an open text file into column vector A and interprets values in the file according to the format specified by formatSpec. 2) This syntax reads file data into an array, A, with dimensions, sizeA, and positions the file pointer after the last value read. 3) This syntax additionally returns the number of fields that fscanf reads into A.

Example:

Fscanf_ex:

```
x = 1:1:8;
% set the x range
y = [x;rand(1,8)];
% create 8 random numbers and store them into an array
fileID = fopen('nums2.txt','w');
% create the file nums2.txt and open it
fprintf(fileID,'%d %4.4f\n',y);
% print the contents of nums2.txt
type nums2.txt
% show the contents of the file
fileID = fopen('nums2.txt','r');
% open the file for reading, and obtain the file identifier, fileID
formatSpec = '%d %f';
% define the format of the data to read
sizeA = [2 Inf];
% define the shape of the output array
A = fscanf(fileID,formatSpec,sizeA)
% Read the file data, filling output array, A, in column order. fscanf reuses the
format, formatSpec, throughout the file
fclose(fileID);
% close the file
```



Q4

Syntax	Meaning	Description
1) A = sscanf(str,formatSpec) 2) A = sscanf(str,formatSpec,sizeA) 3) [A,n] = sscanf(___) 4) [A,n,errmsg] = sscanf(___) 5) [A,n,errmsg,nextindex] = sscanf(___)	Read formatted data from strings	1) This syntax reads data from str, converts it according to the format specified by formatSpec, and returns the results in an array. str is either a character array or a string scalar. 2) This syntax sets the size of the output array to be sizeA and then reads data from str into the output array. 3) This syntax also returns the number of elements that sscanf successfully reads into A. 4) This syntax also returns a character vector containing an error message when sscanf fails to read all the data into A.

		5) This syntax also returns the index of the position in str that immediately follows the last character scanned by sscanf.

Examples:

Sscanf1:

```
chr = '5.5  3.5  7.5'
% create a character vector that represents several numbers separated by whitespace
characters
A = sscanf(chr,'%f')
% convert the character vector to a column vector of numbers
```

Sscanf2:

```
str = "12 17 13 68 23 87 35"
% create a string that contains numbers separated by whitespace characters
[A,n] = sscanf(str,'%d')
% convert the numbers in the string using %d
% Count the elements that sscanf puts into the output array when it converts the
string to numbers
```

The difference between sscanf and fscanf:

fscanf reads the data from a chosen text file but sscanf only reads formatted data from strings.

Sscanf_fscanf_difference:

```
fid = fopen('bench.txt','r');
k = 0;
while ~feof(fid)
 curr = fscanf(fid,'%c',1);
 % Read data from text file
 if ~isempty(curr)
 k = k+1;
 bstr(k) = curr;
 end
end
fclose(fid);bstr,
[A,n,errmsg] = sscanf(bstr,'%f')

info='0.0735    0.1026    0.1964    0.2728    0.3955    0.3876';
info = sscanf(info,'%f', [2 3])
```

```
Command Window
  >> sscanf_fscanf_difference

  bstr =

      'MATLAB Programming for Engineers and Scientists Specialization

       About this Course

       This course teaches computer programming to those with little to no previous experience. It uses the programming system and language

       Nevertheless, this course is not a MATLAB tutorial. It is an introductory programming course that uses MATLAB to illustrate general

       Students taking the course will get a MATLAB Online license free of charge for the duration of the course. The students are encourage


       WHAT YOU WILL LEARN
       You will learn fundamental computer programming concepts such as variables, control structures, functions and many others.

       You will learn about various data types and how to handle them in MATLAB.

       You will learn the powerful support MATLAb provides for working with matrices.

       You will learn about file input/output.
```

SKILLS YOU WILL GAIN
Computer Programming
Problem Solving
Matlab
Programming Language
Flexible deadlines
Reset deadlines in accordance to your schedule.
Shareable Certificate
Earn a Certificate upon completion
100% online
Start instantly and learn at your own schedule.
Course 1 of 3 in the
MATLAB Programming for Engineers and Scientists Specialization
Beginner Level
Approx. 35 hours to complete
English
Subtitles: Arabic, French, Portuguese (European), Greek, Italian, Vietnamese, German, Russian, English, Spanish
Instructors
Instructor rating

Akos Ledeczi
Professor of Computer Engineering, Computer Science and Electrical Engineering
Department of Electrical Engineering and Computer Science

2 Courses
Mike Fitzpatrick
Professor Emeritus of Computer Science, Computer Engineering, Electrical Engineering, Neurosurgery, and Radiology
Electrical Engineering & Computer Science, Neurological Surgery, Radiology & Radiological Sciences

2 Courses
Offered by
Vanderbilt University logo
Vanderbilt University
Vanderbilt University, located in Nashville, Tenn., is a private research university and medical center offering a full-range

Other courses in this Specialization
Syllabus - What you will learn from this course

WEEK 1
1 hour to complete
Course Pages
1 video (Total 2 min), 3 readings
WEEK 2
3 hours to complete
The MATLAB Environment
We will learn how to start MATLAB and will familiarize ourselves with its user interface. We will learn how to use MATLAB as a

```
7 videos (Total 132 min), 1 reading, 2 quizzes
WEEK 3
3 hours to complete
Matrices and Operators
The basic unit with which we work in MATLAB is the matrix. We solve problems by manipulating matrices, and operators are the p

6 videos (Total 82 min), 1 reading, 4 quizzes
WEEK 4
2 hours to complete
Functions
Functions let us break up complex problems into smaller, more manageable parts. We will learn how functions let us create reus

7 videos (Total 50 min), 1 reading, 3 quizzes
WEEK 5
3 hours to complete
Programmer's Toolbox
MATLAB has useful built-in functions and we will explore many of them in this section. We will learn about polymorphism and ho

5 videos (Total 83 min), 1 reading, 2 quizzes
WEEK 6
5 hours to complete
Selection
Selection is the means by which MATLAB makes decisions about the order in which it executes its statements. We will learn how
```

```
Programmer's Toolbox
MATLAB has useful built-in functions and we will explore many of them in this section. We will learn about polymorphism and ho

5 videos (Total 83 min), 1 reading, 2 quizzes
WEEK 6
5 hours to complete
Selection
Selection is the means by which MATLAB makes decisions about the order in which it executes its statements. We will learn how

7 videos (Total 80 min), 2 readings, 4 quizzes
WEEK 7
7 hours to complete
Loops
Loops give computers their power. We will learn how to use both of MATLAB's loop constructs: the for-loop and the while-loop.

5 videos (Total 133 min), 2 readings, 4 quizzes
WEEK 8
6 hours to complete
Data Types
Computers operate on bits, but humans think in terms of numbers, words, and other types of data. Like any good language, MATLA

6 videos (Total 194 min), 3 readings, 2 quizzes
WEEK 9
7 hours to complete
File Input/Output
Files are named areas in permanent memory for storing data that can be used as input or output to MATLAB and to other programs

4 videos (Total 75 min), 5 readings, 5 quizzes
'
```

```
A =

     []


n =

     0


errmsg =

    'Matching failure in format.'


info =

    0.0735    0.1964    0.3955
    0.1026    0.2728    0.3876
```

Q5

fscanf, fgetl and fgets already discussed. Here are some other ways to read input files in Matlab:

- text = fileread(filename) returns contents of the file filename as a character vector.

- S = readlines(filename) creates an N-by-1 string array by reading an N-line file. Options could also be added like 'EmptyLineRule','skip' to skips empty lines.

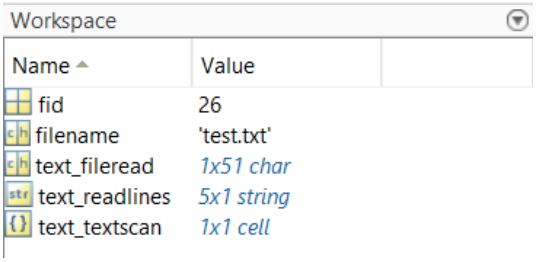
- C = textscan(fileID,formatSpec) reads data from an open text file into a cell array, C. The text file is indicated by the file identifier, fileID. textscan attempts to match the data in the file to the conversion specifier in formatSpec. The textscan function reapplies formatSpec throughout the entire file and stops when it cannot match formatSpec to the data. Textscan also has other syntaxes that we did not bring here because we thought the above syntax is enough to show how it works.

- Textread is also another option that is not recommended by Mathworks. We can use texscan instead.

- readtable, readtimetable. readmatrix and readvars also can read text files but they are used in more special cases that their names indicate.

```
clc;
clear;

filename = 'test.txt';
fid = fopen('test.txt', 'r');
text_fileread = fileread(filename)
text_readlines = readlines(filename)
text_textscan = textscan(fid,'%s')
```

```
Command Window
  text_fileread =

      '    x   x^2
        0.00 0.0000
        0.10 0.0100
        1.00 1.0000
       '


  text_readlines =

    5×1 string array

      "    x   x^2"
      " 0.00 0.0000"
      " 0.10 0.0100"
      " 1.00 1.0000"
      ""


  text_textscan =

    1×1 cell array

      {8×1 cell}
```


Workspace
Name
Value
fid
26
filename
'test.txt'
text_fileread
1x51 char
text_readlines
5x1 string
text_textscan
1x1 cell

Q6

Syntax	Meaning	Description
1) fprintf(fileID,formatSpec ,A1,...,An) 2) fprintf(formatSpec,A1,..., An) 3) nbytes = fprintf(___)	Write data to text file	1) applies the formatSpec to all elements of arrays A1,...An in column order, and writes the data to a text file. fprintf uses the encoding scheme specified in the call to fopen. 2) formats data and displays the results on the screen. 3) returns the number of bytes that fprintf writes, using any of the input arguments in the preceding syntaxes.

Input/Output Arguments:

fileID — File identifier, specified as one of the following: A file identifier obtained from fopen, 1 for standard output (the screen), 2 for standard error.
Data Types: double

formatSpec — Format of the output fields, specified using formatting operators. formatSpec also can include ordinary text and special characters. It can be a character vector in single quotes or a string scalar.

Formatting Operator

A formatting operator starts with %, and ends with a conversion character. Optionally, you can specify identifier, flags, field width, precision, and subtype operators between % and the conversion character.

Conversion Character: %i for signed int, %u for unsigned int, %f for floats, %s for strings, etc.
Optional Operators:
Identifier $: Order for processing the function input arguments.
Flags: _ for left-justify, + for sign character, 0 for zero padding, etc.
Field width: %12d for example to show min number of characters to print is 12.
There are also other operators that could make this report boring, so we did not bring them here.

A1,...,An — Numeric or character arrays, specified as a scalar, vector, matrix, or multidimensional array.
Data Types: single | double | int8 | int16 | int32 | int64 | uint8 | uint16 | uint32 | uint64 | logical | char

nbytes — Number of bytes that fprintf writes, returned as a scalar. When writing to a file, is determined by the encoding. When printing to the screen, is the number of characters displayed.

Examples:

```
clc;
clear;

A1 = [66.9, 90];
A2 = [0.8,  5 ;
      700, 0];
fprintf('var1: %5.2f var2: %6.1f\n',A1,A2)

fprintf('%i\n',[1 3 5 7]);

x = 0:.1:1;
A = [x; x.*x];
fileID = fopen('test.txt','w');
nbytes = fprintf(fileID,'%5s %5s\n','x','x^2')
nbytes = fprintf(fileID,'%5.2f %5.4f\n',A)
fclose(fileID);
type test.txt % writes the text file in the console
```

```
Command Window
var1: 66.90 var2:   90.0
var1:  0.80 var2:  700.0
var1:  5.00 var2:    0.0
1
3
5
7

nbytes =

    12


nbytes =

   143


    x   x^2
 0.00 0.0000
 0.10 0.0100
 0.20 0.0400
 0.30 0.0900
 0.40 0.1600
 0.50 0.2500
 0.60 0.3600
 0.70 0.4900
 0.80 0.6400
 0.90 0.8100
 1.00 1.0000
```

Syntax	Meaning	Description
1) type filename	Display contents of file	1) displays the contents of the specified file in the Command Window.

Input Arguments
filename — File name to display, specified as a character vector or a string. filename can be an absolute or relative path and can include a path and a file extension. type supports file names with these extensions.
.mlx Matlab live script , .mlapp for Matlab App File and .m for Matlab code
If you do not specify a file extension and a file without an extension does not exist, then the extension is .mlx, .mlapp, or .m.

* type leverages automatic character set detection to determine the file encoding for .m and other text files.

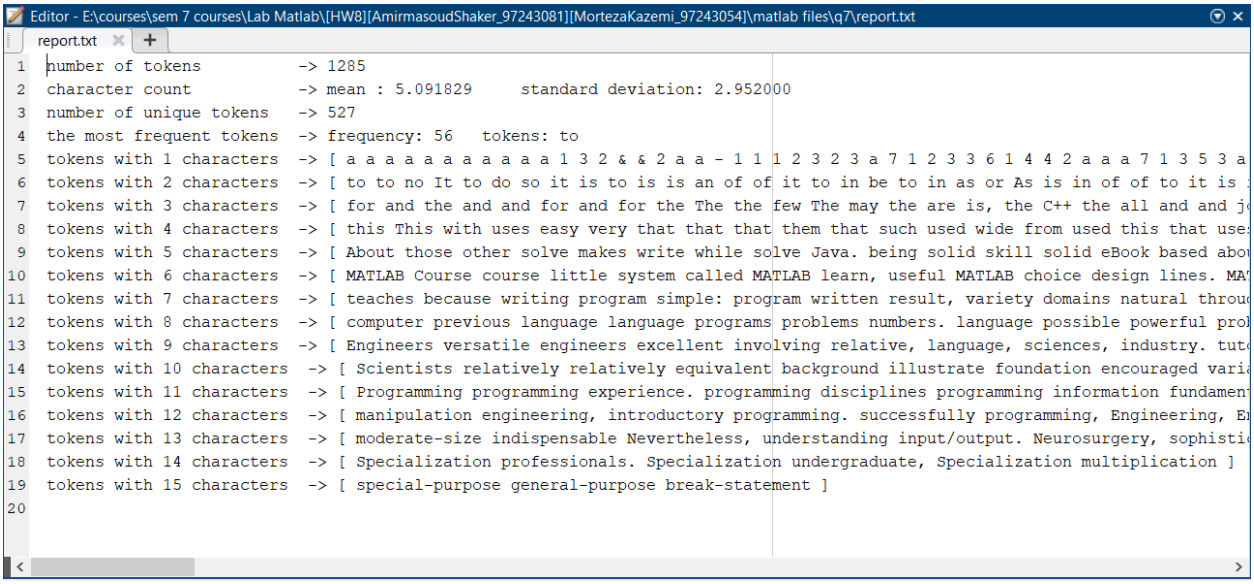
Examples:

```
clc;
clear;

type test.txt

type q6_example_2
```

Q7

```
clc;
clear;

f_adr = input('enter file adress: ', 's');
% f_adr = 'bench.txt';
f_id_r = fopen(f_adr); % file pointer for reading
f_id = fopen('report.txt','w'); % file pointer for writing

str_raw = textscan(f_id_r,'%s'); % reads the file into a 1*1 cell of cell array
text = str_raw{1}; % get the cell array
text = string(text); % convert the cell array to a string array

fprintf(f_id, 'number of tokens          -> %i \n', numel(text)); % numel returns num
of elements in the array text

char_count = strlength(text); % store the length of all the tokens in an array
size(char_count);
% calculating mean and standard deviation:
fprintf(f_id, 'character count           -> mean : %f\t standard deviation: %f \n',
mean(char_count), std(char_count));

unique_text = unique(text); % returns the same data as in text, but with no
repetitions.
fprintf(f_id, 'number of unique tokens   -> %i \n', numel(unique_text));

[s,i1,i2]=unique(text); % i1 and i2 are indexes to access text and s respectively. s
contains unique strings sorted
[M,F,C] = mode(i2); %  computes the mode over all elements of i2. C has all the modes
and F is the mode frequency.
modes = cell2mat(C); % convert cell array to array
fprintf(f_id, 'the most frequent tokens  -> frequency: %i \t tokens: ', F);
% a loop to print all the modes:
for n = 1 : length(modes)
    fprintf(f_id, '%s \t', s(modes(n)));
end
fprintf(f_id, '\n');

% printing lists based on string length:
for i = 1 : max(char_count)
    fprintf(f_id, 'tokens with %i characters  -> [ ', i);
    for j = 1 : length(text)
       if(char_count(j) == i)
           fprintf(f_id, '%s ', text(j));
       end
    end
    fprintf(f_id, '] \n');
end

fclose(f_id_r);
fclose(f_id);
```

```
Command Window
  enter file adress: bench.txt
```


Editor - E:\courses\sem 7 courses\Lab Matlab\[HW8][AmirmasoudShaker_97243081][MortezaKazemi_97243054]\matlab files\q7\report.txt
report.txt
number of tokens -> 1285
character count -> mean : 5.091829 standard deviation: 2.952000
number of unique tokens -> 527
the most frequent tokens -> frequency: 56 tokens: to
tokens with 1 characters -> [a a a a a a a a a a a 1 3 2 & & 2 a a - 1 1 1 2 3 2 3 a 7 1 2 3 3 6 1 4 4 2 a a a 7 1 3 5 3 a
tokens with 2 characters -> [to to no It to do so it is to is is an of of it to in be to in as or As is in of of to it is
tokens with 3 characters -> [for and the and and for and for the The the few The may the are is, the C++ the all and and j
tokens with 4 characters -> [this This with uses easy very that that that them that such used wide from used this that use
tokens with 5 characters -> [About those other solve makes write while solve Java. being solid skill solid eBook based abo
tokens with 6 characters -> [MATLAB Course course little system called MATLAB learn, useful MATLAB choice design lines. MA
tokens with 7 characters -> [teaches because writing program simple: program written result, variety domains natural throu
tokens with 8 characters -> [computer previous language language programs problems numbers. language possible powerful pro
tokens with 9 characters -> [Engineers versatile engineers excellent involving relative, language, sciences, industry. tut
tokens with 10 characters -> [Scientists relatively relatively equivalent background illustrate foundation encouraged vari
tokens with 11 characters -> [Programming programming experience. programming disciplines programming information fundamen
tokens with 12 characters -> [manipulation engineering, introductory programming. successfully programming, Engineering, E
tokens with 13 characters -> [moderate-size indispensable Nevertheless, understanding input/output. Neurosurgery, sophisti
tokens with 14 characters -> [Specialization professionals. Specialization undergraduate, Specialization multiplication]
tokens with 15 characters -> [special-purpose general-purpose break-statement]