Module 10: File I/O

Chul Min Yeum

Assistant Professor

Civil and Environmental Engineering

University of Waterloo, Canada

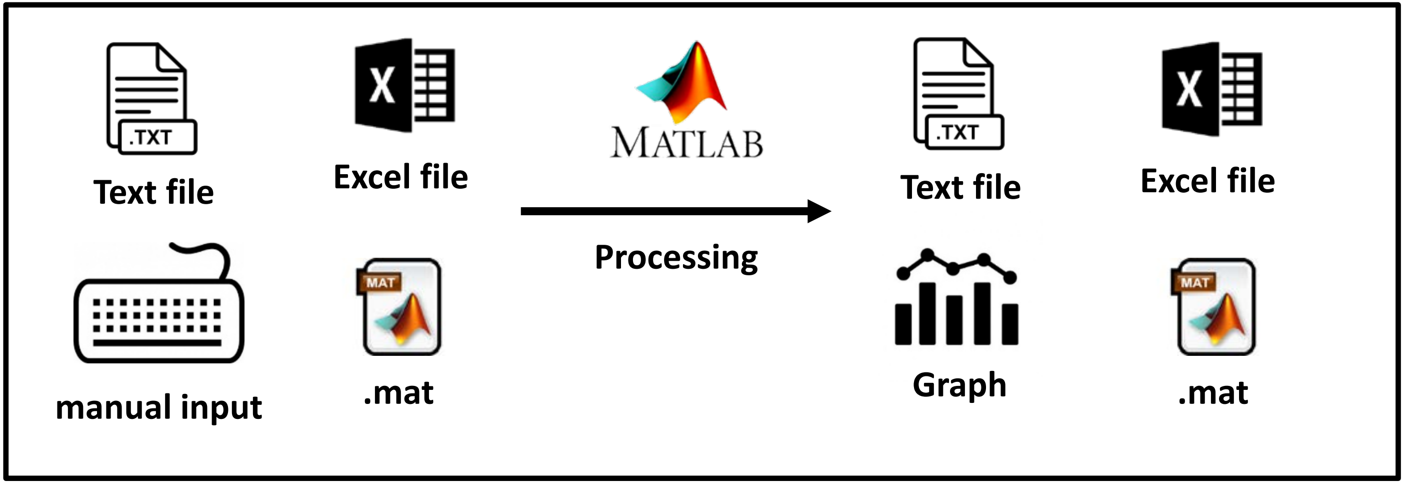
UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING

Module 10: Learning Outcomes

- Store variables in MATLAB Workspace to a MAT-file
- Import text files to MATLAB Workspace
- Explain the difference between text and numeric data when they are read from the file
- Read and Write MS Excel files

File Input & Output (I/O)

- MATLAB has functions to read from and write to many different file types, for example, spreadsheets.
- MATLAB has a special binary file type that can be used to store variables and their contents in MAT-files.



Text file

Excel file

MATLAB

Processing

Text file

Excel file

manual input

.mat

Graph

.mat

Using MAT-files for Variables

- MATLAB has functions that allow reading and saving variables from files.
- These files are called MAT-files (because the extension is `.mat`).
- Variables can be written to MAT-files, appended to them, and read from them.
- Rather than just storing data, MAT-files **store variable names and their values**

- To save all workspace variables in a file, the command is:

```
    save filename
```

- To save just one variable to a file, the format is:

```
    save filename variablename
```

- To read variables from a MAT-file into the base workspace:

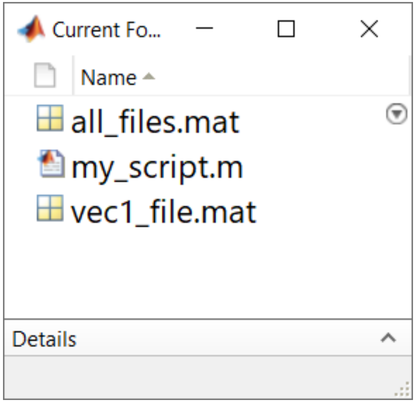
```
    load filename variablelist
```

Example Save and Load Variables Using MAT files

my_script. m

```
vec1 = zeros(1, 3);
mat1 = ones(2, 2);
char1 = ‘a’;

save all_files
save vec1_file vec1
```

Current folder



all_files.mat
my_script.m
vec1_file.mat

```
load all_files
```

Name	Value
vec1	[0 0 0]
mat1	[1 1; 1 1]
char1	‘a’

```
load vec1_file
```

Name	Value
vec1	[0 0 0]

```
load all_files mat1
```

Name	Value
mat1	[1 1; 1 1]

⚠ : When you are working on a large-scale project, saving all files causes long saving and loading time. Selectively saving and loading is recommended.

Importing Data from a .txt File

- The text file must be saved in the current folder that you are working in on MATLAB

- Delimiter: sequence of one or more characters used to specify boundaries between separate regions (e.g., comma (,), semicolon (;), space( ))

- Three importing scenarios:
  - Importing numeric data
  - Importing character (string) data
  - Importing numeric and character data

Print Formatting Text

- The print formatting texts specify in what layout and what data type a column of data will be imported/exported as – for multiple columns of data, use multiple print format operators

- Common print formatting texts:
  - `%d` –For integer numbers
  - `%f` –For floating point (decimal) numbers
  - `%s` –For entire character vectors or strings

- Note that you can only use 1 format text per column (you cannot specify `'%s'` for the first value in a column and `'%f'` for all other values

- Example: If you wanted your first column of data to be integer data, the second column to be stored as character data and the third to be floating point numbers, the correct format text would be: `'%d %s %f'`

textscan Function

- The `textscan` function will create a **cell** array from the data read on the `.txt` file

```
C = textscan(fileID,formatSpec)
```

- This function can be used to import **numeric data, text data, and both types of data** from the same file easily
- When importing using `textscan`, the data is imported column-by-column.
- You can specify using print formatting texts what type of data you want each column to be.
- Data will be stored in a cell, **where each column is a different cell array element**
- You have the option of specifying a delimiter. The default delimiter is white-space.

fopen and fclose function

- Used to open a file for a specific purpose
- You can specify your purpose using a permission specifier(the second input)
- The file does not have to already exist (although for reading data it should)

- Read only access (default):

```
    fid = fopen(‘sample_data.txt’, ‘r’)
```

- Write only access (discard original contents of a file):

```
    fid = fopen(‘sample_data.txt’, ‘w’)
```

- Read and write access (discard original contents of file if writing):

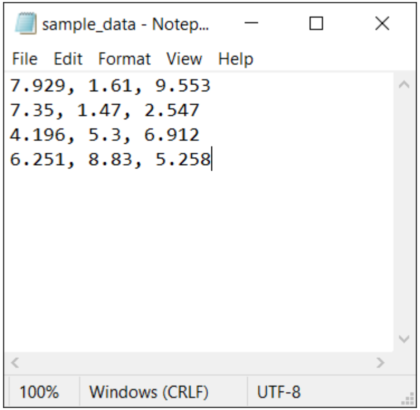
```
    fid = fopen(‘sample_data.txt’, ‘w+’)
```

- Once you finish read/write operation on the file, you need to close the file

```
    fclose(fid)
```

General Procedure for Importing Data using textscan

```
1   fid = fopen(‘sample_data.txt’);
2
3   test_data = textscan(fid, ‘%f %f %f’, ‘delimieter’, ‘,’);
4
5   fclose(fid);
```



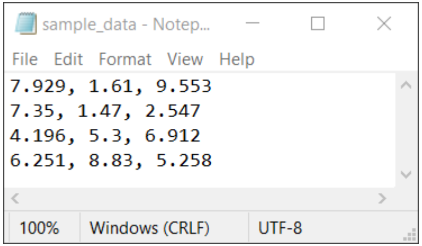
```
7.929, 1.61, 9.553
7.35, 1.47, 2.547
4.196, 5.3, 6.912
6.251, 8.83, 5.258
```

- Line 1: Open the file to be read by creating a ‘fid’. This is creating a numeric ID that represents the file.

- Line 3: Use the textscan function to import the data using the proper format string and delimiter. If the delimiter is whitespace, you do not need to specify a delimiter.

- Line 5: Close the file you have just read.

Example: Read Numeric Data



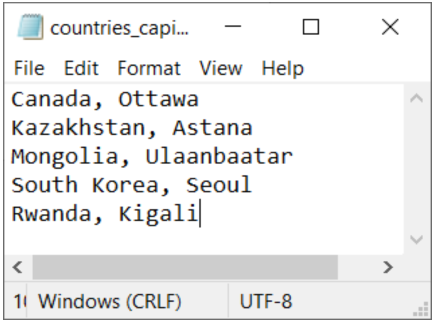
```
7.929, 1.61, 9.553
7.35, 1.47, 2.547
4.196, 5.3, 6.912
6.251, 8.83, 5.258
```

```
fid = fopen('sample_data.txt’);

samp_data = textscan(fid, '%f %f %f', 'delimiter', ',’);

fclose(fid);
```

```
>> samp_data

samp_data =

  1×3 cell array

    {4×1 double}    {4×1 double}    {4×1 double}

>> samp_data{1}

ans =

    7.9290
    7.3500
    4.1960
    6.2510
```

Three columns of numeric data – when importing, you will need to specify three print formatting texts.

Example: Read Text Data



```
Canada, Ottawa
Kazakhstan, Astana
Mongolia, Ulaanbaatar
South Korea, Seoul
Rwanda, Kigali
```

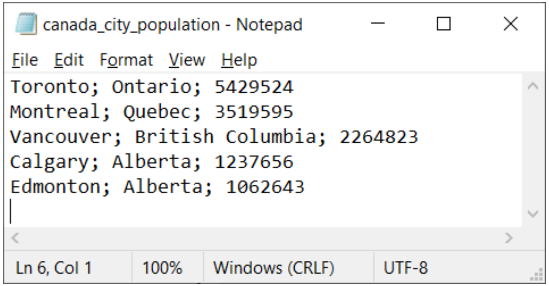
⚠: When you read or write text information in a text file, you should not use the white-space delimiter because texts likely include the white space.

```
fid = fopen('countries_capitals.txt’);

text_data = textscan(fid, '%s %s', 'delimiter', ',’);

fclose(fid);
```

```
>> text_data

text_data =

  1×2 cell array

    {5×1 cell}    {5×1 cell}

>> text_data{1}

ans =

  5×1 cell array

    {'Canada'     }
    {'Kazakhstan' }
    {'Mongolia'   }
    {'South Korea'}
    {'Rwanda'     }
```

Example: Read Numeric and Text Data

```
fid = fopen('canada_city_population.txt’);

city_data = textscan(fid, '%s %s %f', 'delimiter', ';’);

fclose(fid);
```

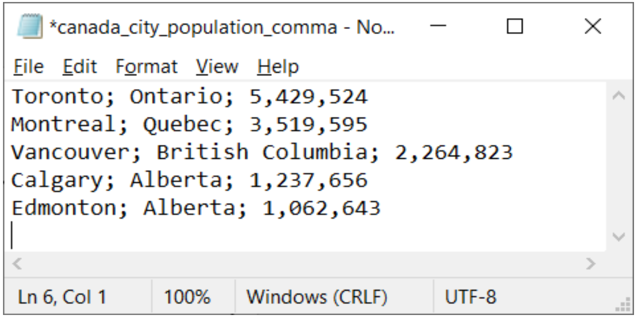


```
Toronto; Ontario; 5429524
Montreal; Quebec; 3519595
Vancouver; British Columbia; 2264823
Calgary; Alberta; 1237656
Edmonton; Alberta; 1062643
```

```
>> city_data

city_data =

  1×3 cell array

    {5×1 cell}    {5×1 cell}    {5×1 double}

>> city_data{3}

ans =

     5429524
     3519595
     2264823
     1237656
     1062643
```

Example: Read Number (with Comma) and Text Data

Optional


*canada_city_population_comma - No...
File Edit Format View Help
Ln 6, Col 1
100%
Windows (CRLF)
UTF-8

```
Toronto; Ontario; 5,429,524
Montreal; Quebec; 3,519,595
Vancouver; British Columbia; 2,264,823
Calgary; Alberta; 1,237,656
Edmonton; Alberta; 1,062,643
```

```
fid = fopen( ...
'canada_city_population_comma.txt’);

text_data = textscan(fid, ...
'%s %s %f',  'delimiter', ';’);

fclose(fid);
```

Not working

⚠: 5,429, 524 is not a numeric type because of commas. Thus, the corresponding column can’t be read using ‘%f’.

```
fid = ...
fopen('canada_city_population_comma.txt’);

text_data = textscan(fid, '%s %s %s’,...
'delimiter', ';’);
fclose(fid);

pop_data_cell = text_data{3};

n_data = numel(pop_data_cell);
pop_data = zeros(n_data,1);
for ii=1:n_data
   text_text = pop_data_cell{ii};
   num_text = text_text(text_text ~= ',');
   pop_data(ii) = str2double(num_text);
end
```

```
>> pop_data

pop_data =

     5429524
     3519595
     2264823
     1237656
     1062643
```

Read and Write Excel Data

- The `xlsread` function can be used for importing MS Excel data into MATLAB. However, the use of `xlsread` function is not recommend starting in R2019a.
- MATLAB introduce a **new data type** called **`table`**. It is a suitable for column-oriented or tabular data that is often stored as columns in a text file or a spreadsheet. However, we will not cover this new data type in this course.
- Instead, we will import the data as a cell array using `readcell` or `readmatrix`

readmatrix and readcell Function

- `readmatrix` creates an array by reading column-oriented data from a file.

```
M = readmatrix(filename)
M = readmatrix(filename,'Sheet',sheet,'Range',range)
```

- `readcell` creates a cell array by reading column-oriented data from a file.

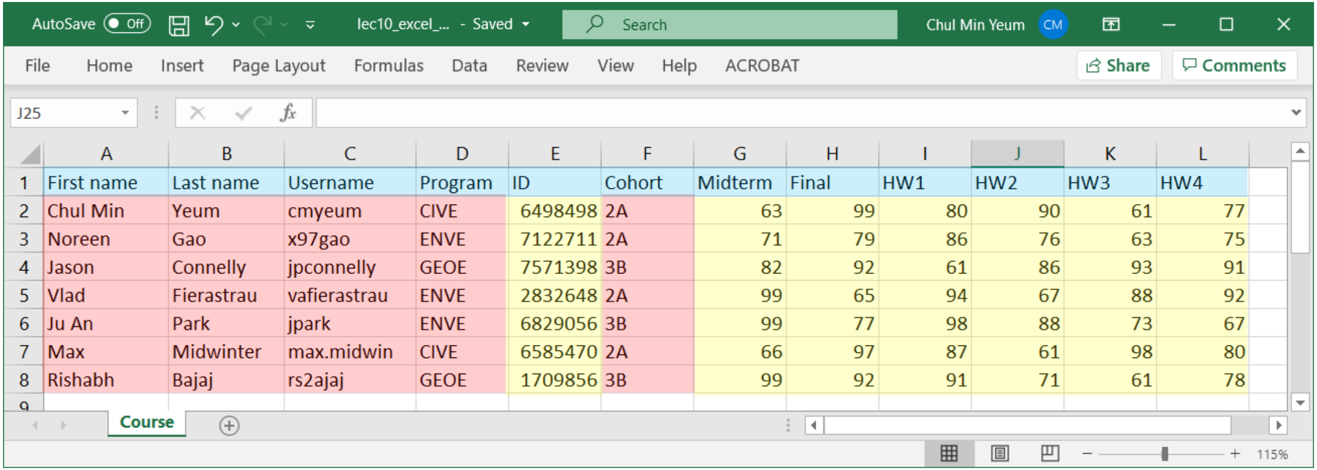
```
C = readcell(filename)
C = readcell(filename,'Sheet',sheet,'Range',range)
```

😊: `readmatrix` or `readcell` determines the file format from the file extension:

- .txt, .dat, or .csv for delimited text files
- .xls, .xlsb, .xlsm, .xlsx, .xltm, .xltx, or .ods for spreadsheet files

Example: Read Data from an Excel File (File)

lec10_excel_file.xlsx



	A	B	C	D	E	F	G	H	I	J	K	L
1	First name	Last name	Username	Program	ID	Cohort	Midterm	Final	HW1	HW2	HW3	HW4
2	Chul Min	Yeum	cmyeum	CIVE	6498498	2A	63	99	80	90	61	77
3	Noreen	Gao	x97gao	ENVE	7122711	2A	71	79	86	76	63	75
4	Jason	Connelly	jpconnelly	GEOE	7571398	3B	82	92	61	86	93	91
5	Vlad	Fierastrau	vafierastrau	ENVE	2832648	2A	99	65	94	67	88	92
6	Ju An	Park	jpark	ENVE	6829056	3B	99	77	98	88	73	67
7	Max	Midwinter	max.midwin	CIVE	6585470	2A	66	97	87	61	98	80
8	Rishabh	Bajaj	rs2ajaj	GEOE	1709856	3B	99	92	91	71	61	78
9												

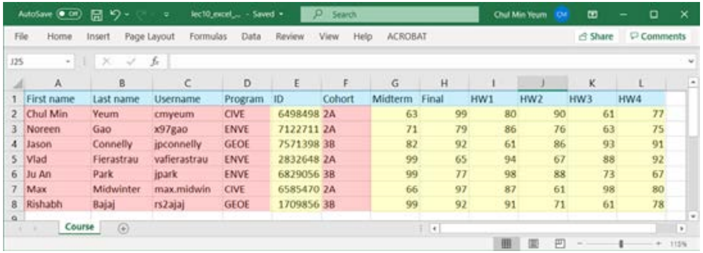
Header

Text values

Numeric values

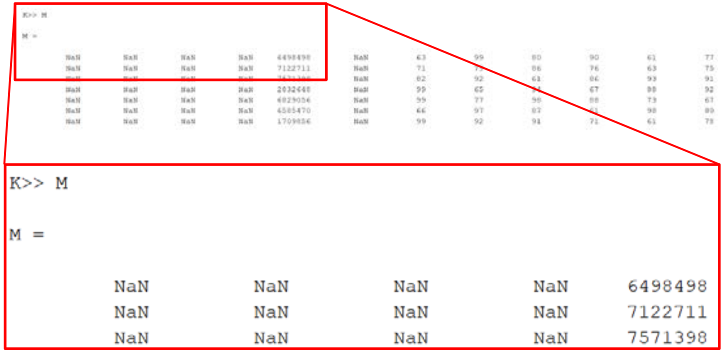
⚠ : We need to use a cell array to contain all these data in one variable.

Example: Read Data from an Excel File (readmatrix)



	A	B	C	D	E	F	G	H	I	J	K	L
1	First name	Last name	Username	Program	ID	Cohort	Midterm	Final	HW1	HW2	HW3	HW4
2	Chul Min	Yeum	cmyeum	CIVE	6498498	2A	63	99	80	90	61	77
3	Noreen	Gao	x97gao	ENVE	7122711	2A	71	79	86	76	63	75
4	Jason	Connelly	jpconnelly	GEOE	7571398	3B	82	92	61	86	93	91
5	Vlad	Fierastrau	vafierastrau	ENVE	2832648	2A	99	65	94	67	88	92
6	Ju An	Park	jpark	ENVE	6829056	3B	99	77	98	88	73	67
7	Max	Midwinter	max.midwin	CIVE	6585470	2A	66	97	87	61	98	80
8	Rishabh	Bajaj	rs2ajaj	GEOE	1709856	3B	99	92	91	71	61	78

```
filename = 'lec10_excel_file.xlsx';
M = readmatrix(filename);
```



```
K>> M

M =

       NaN        NaN        NaN        NaN    6498498        NaN        63        99        80        90        61        77
       NaN        NaN        NaN        NaN    7122711        NaN        71        79        86        76        63        75
       NaN        NaN        NaN        NaN    7571398        NaN        82        92        61        86        93        91
       NaN        NaN        NaN        NaN    2832648        NaN        99        65        94        67        88        92
       NaN        NaN        NaN        NaN    6829056        NaN        99        77        98        88        73        67
       NaN        NaN        NaN        NaN    6585470        NaN        66        97        87        61        98        80
       NaN        NaN        NaN        NaN    1709856        NaN        99        92        91        71        61        78
```

```
K>> M

M =

       NaN        NaN        NaN        NaN    6498498
       NaN        NaN        NaN        NaN    7122711
       NaN        NaN        NaN        NaN    7571398
```

```
filename = 'lec10_excel_file.xlsx';
M = readmatrix(filename);

M_num = M;
lg_mat = isnan(M_num);
idx = logical(sum(lg_mat));
M_num(:,idx) = [];
```

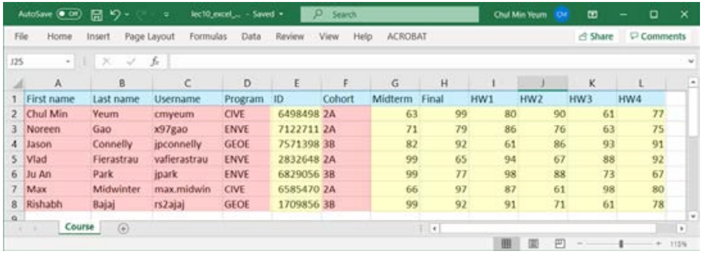
option1

📖: isnan returns a logical array containing 1 where the elements are NaN. Here NaN means it is not a numeric (number) value. We extract numeric array using a vectorized code.

```
filename = 'lec10_excel_file.xlsx';
M = readmatrix(filename);

M1 = readmatrix(filename, 'Range', ...
'E2:E8'); % set range
M2 = readmatrix(filename, 'Range', ...
'G2:L8'); % set range

M_num = [M1 M2]; % joining cell arrays
```

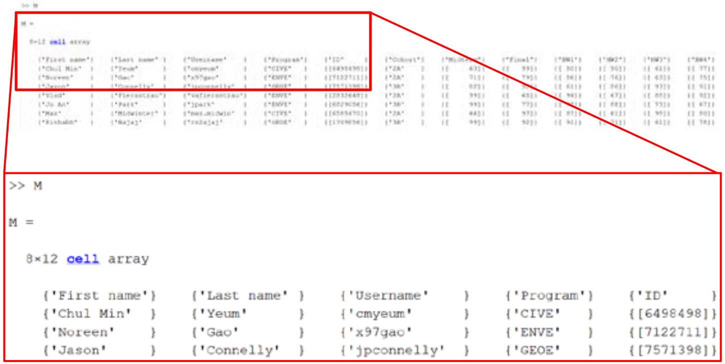
option2

Example: Read Data from an Excel File (readcell)



	A	B	C	D	E	F	G	H	I	J	K	L
1	First name	Last name	Username	Program	ID	Cohort	Midterm	Final	HW1	HW2	HW3	HW4
2	Chul Min	Yeum	cmyeum	CIVE	6498498	2A	63	99	80	90	61	77
3	Noreen	Gao	x97gao	ENVE	7122711	2A	71	79	86	76	63	75
4	Jason	Connelly	jpconnelly	GEOE	7571398	3B	82	92	61	86	93	91
5	Vlad	Fierastrau	vafierastrau	ENVE	2832648	2A	99	65	94	67	88	92
6	Ju An	Park	jpark	ENVE	6829056	3B	99	77	98	88	73	67
7	Max	Midwinter	max.midwin	CIVE	6585470	2A	66	97	87	61	98	80
8	Rishabh	Bajaj	rs2ajaj	GEOE	1709856	3B	99	92	91	71	61	78

```
filename = 'lec10_excel_file.xlsx';
M = readcell(filename);
```



```
>> M

M =

  8×12 cell array

    {'First name'}    {'Last name' }    {'Username'    }    {'Program'}    {'ID'      }
    {'Chul Min'  }    {'Yeum'      }    {'cmyeum'      }    {'CIVE'   }    {[6498498]}
    {'Noreen'    }    {'Gao'       }    {'x97gao'      }    {'ENVE'   }    {[7122711]}
    {'Jason'     }    {'Connelly'  }    {'jpconnelly'  }    {'GEOE'   }    {[7571398]}
```

```
filename = 'lec10_excel_file.xlsx';

M1 = readcell(filename, 'Range’, ...
'E2:E8');
M2 = readcell(filename, 'Range’, ...
'G2:L8’);

M_num_cell = [M1 M2];
M_num = cell2mat(M_num_cell);
```

🕮: All data will be read using cell types. We can access both text and numeric data. However, when you process only numeric data, and spreadsheet contains large volume of text data, the size of M will get larger causing taking up large memory and slow processing speed. Thus, in such case, I recommend using `reamatrix` rather than `readcell`.

writematrix and writecell Function

- `writematrix` writes numeric array A to a file with the name and extension specified by `filename`

```
writematrix(A, filename)
writematrix(A, filename,'Sheet',sheet,'Range',range)
```

- `writecell` write cell array C to a file with the name and extension specified by `filename`

```
writecell(C, filename)
writecell(C, filename,'Sheet',sheet,'Range',range)
```

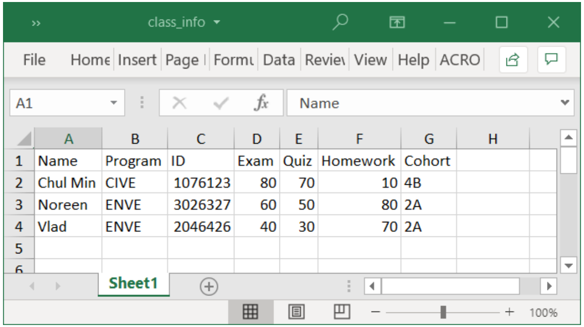
😌: `writematrix` or `writecell` determines the file format based on the specified extension:

- .txt, .dat, or .csv for delimited text files
- .xls, .xlsb, .xlsm, .xlsx, .xltm, .xltx, or .ods for spreadsheet files

Example: Write Data to an Excel File (writecell)

```
cl_info = cell(3,7);

cl_info{1,1} = 'Chul Min';
cl_info{1,2} = 'CIVE';
cl_info{1,3} = 1076123;
cl_info{1,4} = [80 90];
cl_info{1,5} = [70 30 50];
cl_info{1,6} = [4 5 1 2];
cl_info{1,7} = '4B';

cl_info{2,1} = 'Noreen';
cl_info{2,2} = 'ENVE’;
cl_info{2,3} = 3026327;
cl_info{2,4} = [100 70];
cl_info{2,5} = [10 20 70];
cl_info{2,6} = [2 7 8 9];
cl_info{2,7} = '2A';

cl_info{3,1} = 'Vlad';
cl_info{3,2} = 'ENVE';
cl_info{3,3} = 2046426;
cl_info{3,4} = [50 90];
cl_info{3,5} = [90 60 80];
cl_info{3,6} = [1 2 6 2];
cl_info{3,7} = '2A';
```

```
cl_header = {'Name', 'Program', 'ID', 'Exam', 'Quiz’,...
'Homework', 'Cohort’};

cl_all = cat(1, cl_header, cl_info);
% cl_all = [cl_header; cl_info];

writecell(cl_all, ‘cl_info.xlsx’);
```



	A	B	C	D	E	F	G	H
1	Name	Program	ID	Exam	Quiz	Homework	Cohort	
2	Chul Min	CIVE	1076123	80	70	10	4B	
3	Noreen	ENVE	3026327	60	50	80	2A	
4	Vlad	ENVE	2046426	40	30	70	2A	
5								
6								

class_info.xlsx

🕮: In the original cell, there is no header. Thus, before you store the cell array data, you need to append the header to the cell data.

Tip for Quick Testing

Optional

Sometimes, you want to do quick testing on a part of data in an excel file, but you do not want to write a script to read/extract the values from the file.

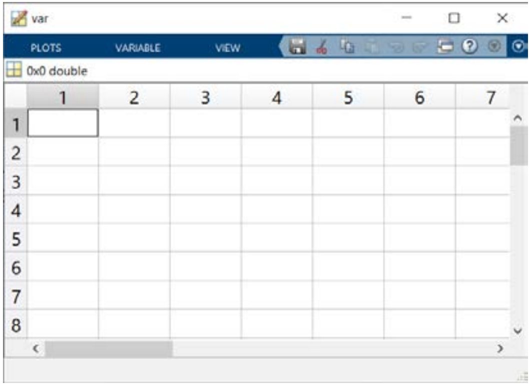
Step 1. Make an empty variable

Step 2. Double click the variable in Workspace to edit the values of the variable.

Step 3. Copy the data from the excel file or text file and paste them in the values of the variable.

Step 4. Save the variable in .mat file.

Step 5. Use the variable for your computation



😆: You can copy excel data and paste them in the variable through this variable window. You can open it by double-clicking the variable name in Workspace. The format of the variable window is similar to the ones in Excel.