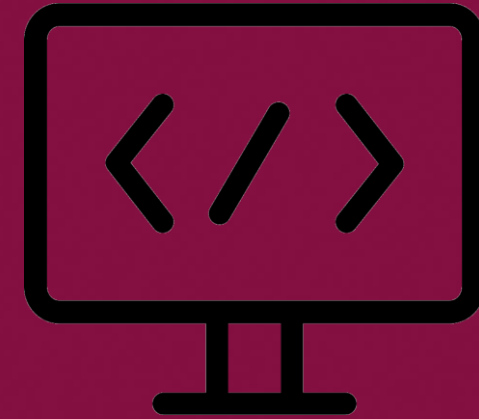


SEEE1022 INTRODUCTION TO SCIENTIFIC PROGRAMMING



UTM
UNIVERSITI TEKNOLOGI MALAYSIA

CH6 Controlled Input and Output



Dr. Mohd Saiful Azimi Mahmud (azimi@utm.my)
P19a-04-03-30, School of Electrical Engineering, UTM

www.utm.my

innovative • entrepreneurial • global



univteknologimalaysia



utm_my



utmofficial

1. Prompt the user for input to an M-file program.
2. Display output using the `disp` function.
3. Create formatted output using `fprintf` and `sprintf`.
4. Import and export ASCII and binary files using `load` and `save` functions.
5. Import and export spreadsheet files using `xlsread` and `xlswrite` functions.
6. Know how to perform debugging .

IMPORT & EXPORT DATA

- At some point, you will often need to store data on a disk.
- The process of moving data between MATLAB and disk are as follows:
 - **IMPORT** – from disk to MATLAB workspace.
 - **EXPORT** – from MATLAB workspace to disk.
- Data stored on the disk can be in two formats:
 1. **Text format** – data values are ASCII codes.
 2. **Binary format** – not ASCII, just binary number. Cannot be viewed in text editor.
- MATLAB use function `load` and `save` to import and export the data respectively.

save FUNCTION

- `save` function save (export) data to disk at MATLAB current folder.

EXAMPLE 8

```
>> a = [1 2 3 4];  
>> b = [2 4 6 8];  
>> save data1.mat  
>> save data2.dat -ascii'  
>> save data3.dat -mat  
>> save data4.dat a -ascii
```

- `data1.mat` is a binary file (default). The file contain all variable available from the workspace. In this example both `a` and `b`.
- `data2.dat` is an ASCII file contain both `a` and `b`
- `data3.dat` is a binary file
- `data4.dat` is an ASCII file contain only values from variable `a`.
- `save` function accept any extension filename.

Load FUNCTION: BINARY FILE

EXAMPLE 9

```
>> a = [1 2 3 4];  
>> b = [2 4 6 8];  
>> save data1.mat  
>> save data2.dat -mat
```

```
>> clear
```

```
>> load data1.mat
```

```
>> whos
```

Name	Size	Bytes	Class	Attributes
a	1x4	32	double	
b	1x4	32	double	

- Binary file save both the variable name and its values.
- Thus, loading the binary data into MATLAB will import the variables as it was saved.
- Loading data2.dat will results the same.

Load FUNCTION: ASCII FILE

EXAMPLE 10

```
>> a = [1 2 3 4];  
>> b = [2 4 6 8];  
>> save data1.dat -ascii
```

```
>> clear  
>> load data1.dat  
>> whos
```

Name	Size	Bytes	Class	Attributes
data1	2x4	64	double	

```
>> data1  
data1 =  
     1     2     3     4  
     2     4     6     8
```

- ASCII file only save the variables values as text.
- Thus, loading the ASCII data into MATLAB will import only the values.
- Imported variable name is the file name itself.

EXAMPLE : GREEK LETTER CONVERSION TABLE

EXAMPLE 11

- In this example, we are going to create a function that convert a string of Greek letter name (e.g. 'alpha', 'beta') to its hex value. This hex value will be very useful when we want to display the Greek letter using `fprintf` or `sprintf` function.
- To do this, we will:
 1. Create a .mat file containing the conversion table. The idea of creating the table in a file is so that it can be easily updated later. Thus, updating the table does not require us to update the program file since program file is normally available to the programmer only, not to the user.
 2. Create the conversion function according to the conversion table file.
- In this example, *cell array* data type will be used (not previously covered). Basically, cell array is just like a normal array but it can have a mix data type inside the array. Other than that, it use a curly bracket `{}` for accessing its array elements.

EXAMPLE : GREEK LETTER CONVERSION TABLE

EXAMPLE 11

1. Create the table as a cell array data and save it into a binary .mat file. To make the process easy, we can write below script file for this purpose.

```
GLtable = {'alpha' 'beta' 'gamma' 'delta' 'epsilon' ...  
          'zeta' 'eta' 'theta'};  
N = length(GLtable);  
startDecVal = hex2dec('3B1');  
  
for n = 1:N  
    dec = startDecVal + n-1;  
    hex = dec2hex(dec);  
    GLtable{2,n} = hex;  
end  
  
save 'GreekLetterHexTable.mat' GLtable
```

- Try to write the pseudo code for the above program to understand how it is done.

EXAMPLE : GREEK LETTER CONVERSION TABLE

EXAMPLE 11

2. Create the conversion function.

```
function GLhex = GL2hex(GLname)

load GreekLetterHexTable %import the conversion table
N = length(GLtable);

for n = 1:N
    currentGL = GLtable{1,n};
    compareMAT = char(GLname,currentGL);
    if compareMAT(1,:) == compareMAT(2,:)
        GLhex = GLtable{2,n};
        break
    end
end
```

```
>> beta = GL2hex('beta')
beta =
    '3B2'
```

OTHER FILE FORMAT

- Other than the binary and ASCII file format, there are many other file formats that can be imported and exported between the MATLAB workspace and disk.
- Spreadsheet, audio, image, video, scientific data and XML document are among those other file formats.
- Instead of `load` and `save`, MATLAB use `read` and `write` notation to import and export data from these files respectively.
- In this chapter, we will only discuss on how to read and write a spreadsheet file (*See MATLAB documentation for other type of files*).

READ AND WRITE SPREADSHEET

- **Syntax:**

```
xlsread(filename, sheet, xlRange)    %import data  
xlswrite(filename, A, sheet, xlRange) %export data
```

- **Description**

1. `filename` is the name of the spreadsheet file.
2. `sheet` is the worksheet number.
3. `xlRange` is a string describing rectangular region on the worksheet. For example 'A1:C3'.
4. `A` is the matrix to be written on the worksheet.
5. The only compulsory input to the functions is `filename`. In this case, default value for `sheet` is 1 and `xlRange` starts at cell A1.
6. See MATLAB documentation for full list of syntax and its detail description.

TEMPERATURE DATA

EXAMPLE 11

A temperature reading in °C at a room was recorded and saved in a tempData.xlsx file. Convert all the temperature reading into Kelvin (°F) and save the values on same file after the degree column.

	A	B	C	D	E	F	G	H
1	Time	8:00 AM	8:10 AM	8:20 AM	8:30 AM	8:40 AM	8:50 AM	9:00 AM
2	°C	23	23	23	23	23	23	
3	°F							
4								
5								
6								
7								
8								
9								

TEMPERATURE DATA

EXAMPLE 11

```
tempC = xlsread('tempData.xlsx', 'B2:Z2');  
tempK = tempC + 273.15;  
fprintf('%d\x2103 %10.2f\x2109\n', [tempC;tempK])  
  
xlswrite('tempData.xlsx', tempK, 'B3:Z3');
```

23°C	296.15°F
24°C	297.15°F
23°C	296.15°F
24°C	297.15°F
24°C	297.15°F
·	·
·	·
34°C	307.15°F

- Open the file tempData.xlsx to see the result.

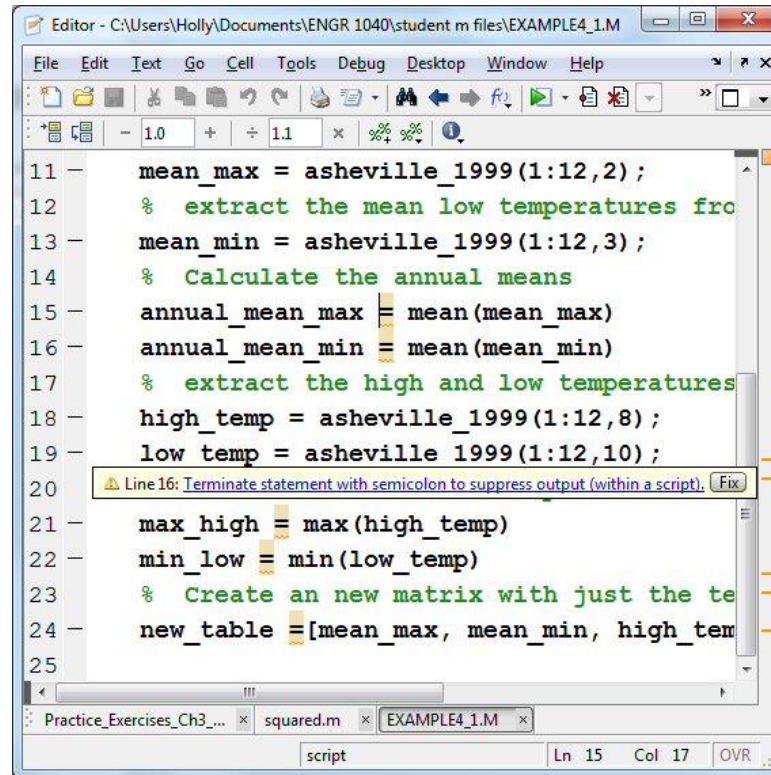
DEBUGGING

DEBUGGING YOUR CODE

- A software bug is a problem that exists in the code you have written.
- Three different types of errors are:
 1. Syntax errors: you simply have typed an illegal expression, independent of the values of the variables in the expression.
 2. Run-time errors: logic errors that result in an illegal expression for specific values of the data (harder to fix)
 3. Logic errors that result in the program executing completely, but the answer that they return is incorrect (hardest to fix).
- MATLAB[®] includes a number of tools to help you debug your code, including the error bar and more comprehensive tools that allow you to step through the code

ERROR BAR

- Whenever you use an M-file, notice that along the right-hand side of the figure window a vertical bar appears
- That marks locations where there are actual errors or where MATLAB® has issued warnings



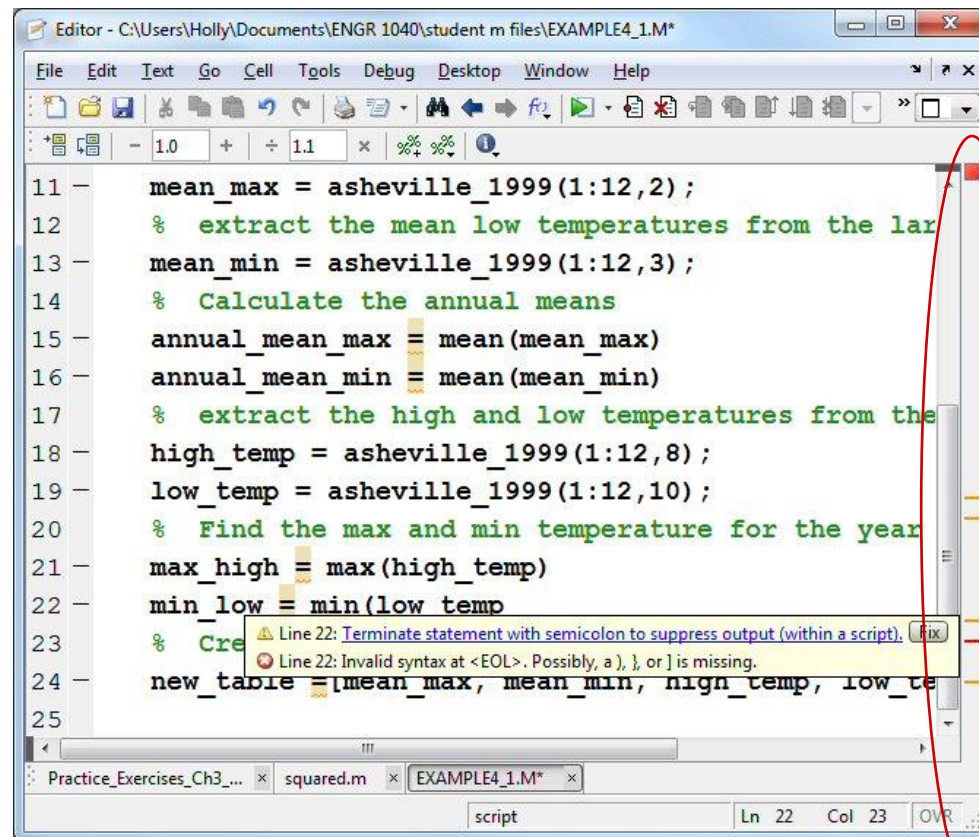
```

11 -   mean_max = asheville_1999(1:12,2);
12 -   % extract the mean low temperatures fro
13 -   mean_min = asheville_1999(1:12,3);
14 -   % Calculate the annual means
15 -   annual_mean_max = mean(mean_max)
16 -   annual_mean_min = mean(mean_min)
17 -   % extract the high and low temperatures
18 -   high_temp = asheville_1999(1:12,8);
19 -   low_temp = asheville_1999(1:12,10);
20 -   ⚠ Line 16: Terminate statement with semicolon to suppress output (within a script). (Fix)
21 -   max_high = max(high_temp)
22 -   min_low = min(low_temp)
23 -   % Create an new matrix with just the te
24 -   new_table = [mean_max, mean_min, high_tem
25 -

```

ERROR BAR

- If the errors shown on the error bar are marked in red, they will cause the M-file to stop executing



```

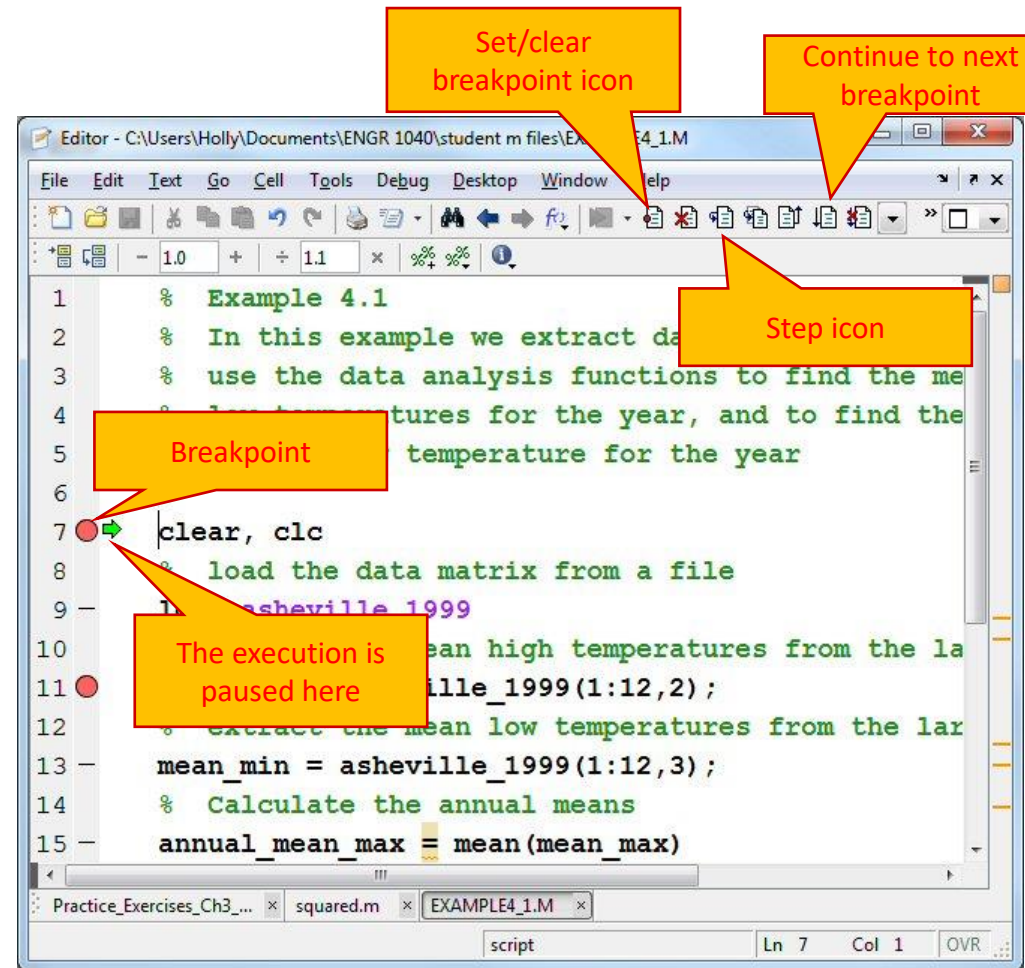
11 — mean_max = asheville_1999(1:12,2) ;
12 — % extract the mean low temperatures from the lar
13 — mean_min = asheville_1999(1:12,3) ;
14 — % Calculate the annual means
15 — annual_mean_max = mean(mean_max)
16 — annual_mean_min = mean(mean_min)
17 — % extract the high and low temperatures from the
18 — high_temp = asheville_1999(1:12,8) ;
19 — low_temp = asheville_1999(1:12,10) ;
20 — % Find the max and min temperature for the year
21 — max_high = max(high_temp)
22 — min_low = min(low_temp
23 — % Cre
24 — new_table = [mean_max, mean_min, high_temp, low_te
25 —

```

Line 22: Terminate statement with semicolon to suppress output (within a script).
 Line 22: Invalid syntax at <EOL>. Possibly, a), }, or] is missing.

BREAKPOINTS

- When trying to find logic errors in a piece of code, it is often useful to run sections of the program, then to stop, evaluate what has happened, and continue.
- Debugging toolbar allows you to set breakpoints (places in the code where the execution stops while you evaluate results) and to step through the code one line at a time. Breakpoints can't be enabled until all of the syntax errors have been resolved.





univteknologimalaysia



utm_my



utmofficial

Thank You

www.utm.my

innovative • entrepreneurial • global