VG101: Introduction to Computer and Programming

Recitation Class Notes

Chenhao YE, Luotian Xu

VG101 TA Group, UM-SJTU Joint Institute

2018 Fall

Source code available at
`https://github.com/Ye-Chenhao/VG101-Recitation-Class-Notes`

Outline

How to learn this course

How to learn this course?

- Follow the lecture
- Finish the homework on time

How to learn this course well?

- Don't do something stupid (including **violation of honor code**, late submission)
- Don't be afraid of asking and **exploring**
- Do as many labs as you can

Tips

- This course mainly focuses on the correctness of your code; you are not expected to improve the efficiency of code.
- Every **error** (mostly in red) means something is wrong. Fix it even when the outputs are sometimes right.
- Please be aware that a program which is right sometimes and wrong the other time, is **wrong**.

About honor code

First, please remember that you are **strongly encouraged** to discuss! Being brave to discuss with DALAO is an essential step to become a DALAO.

What you should not do?

- Don't show any code!
- Don't copy others' code!

Most of time, when you are violating honor code, you yourself know that you are doing something bad.
Please be aware that everytime you trying to violate the honor code but thinking that you can escape from the concequence, you are essentially challenging your instructor and TA group. All the code will be examed by specific tools, and almost all the tricks you can come up with are pale and useless.

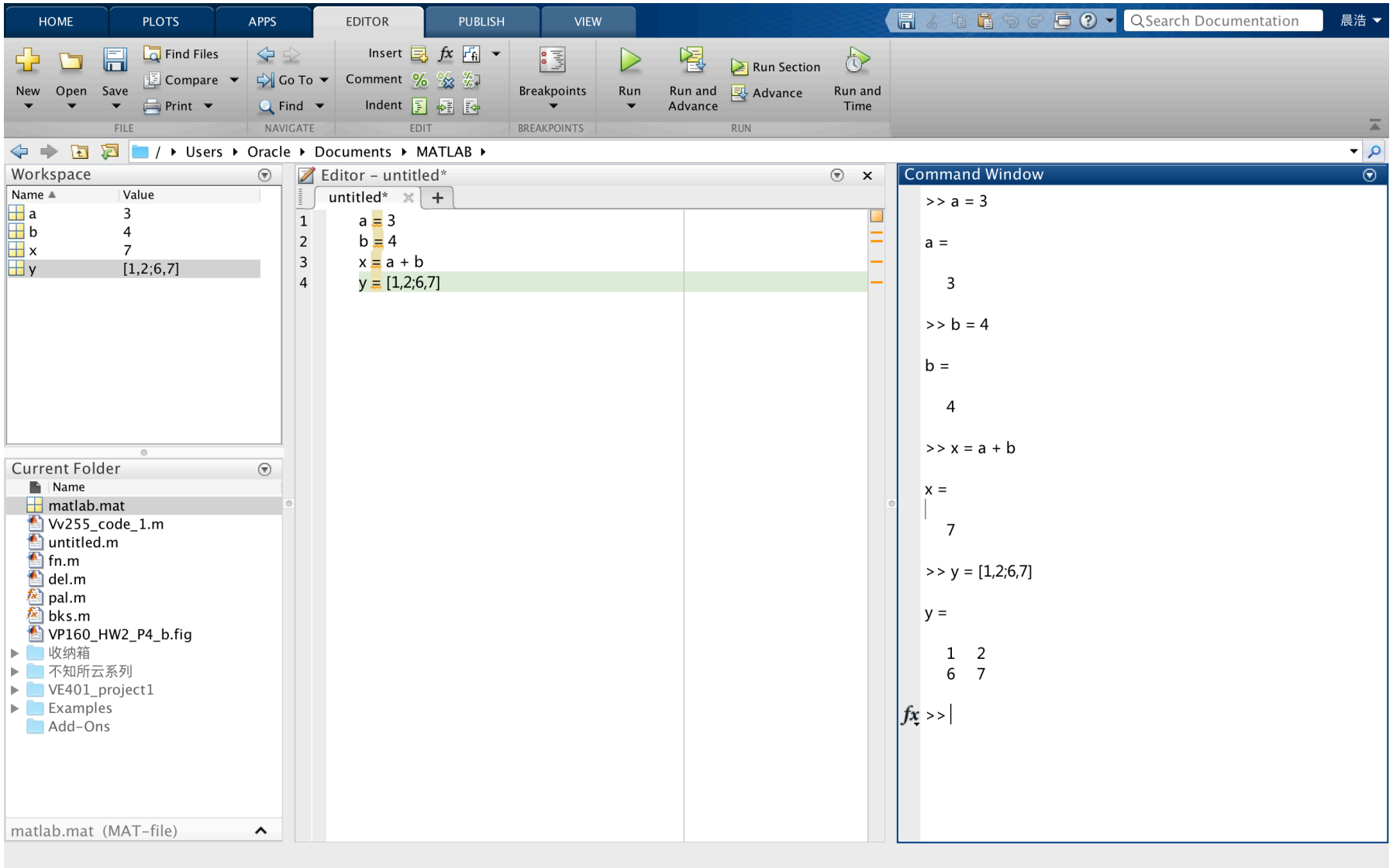
Outline

About MATLAB

- Commercial and experience (<-> Octave)
- Not for general purpose (no one will write an operating system using MATLAB)
- Excel at numerical calculations and plot
- Many useful toolboxes (though we will not cover these in this course, you will use some of them in advanced courses)
- Widely used in academia

Outline

Interface


HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
/ ▸ Users ▸ Oracle ▸ Documents ▸ MATLAB ▸
Workspace
Name
Value
a 3
b 4
x 7
y [1,2;6,7]
Current Folder
Name
matlab.mat
Vv255_code_1.m
untitled.m
fn.m
del.m
pal.m
bks.m
VP160_HW2_P4_b.fig
收纳箱
不知所云系列
VE401_project1
Examples
Add-Ons
matlab.mat (MAT-file)
Editor - untitled*
untitled*
1 a = 3
2 b = 4
3 x = a + b
4 y = [1,2;6,7]
Command Window
>> a = 3
a =
3
>> b = 4
b =
4
>> x = a + b
x =
7
>> y = [1,2;6,7]
y =
1 2
6 7
>>

Interface

Workspace

Store and show what variables are available and their value. You can double-click a variable to see its details (useful for matrix). You can also save and load workspace.

Command Window

Input command direct into the command line. Variables can be seen in the workspace.

Editor

Editor is actually where you "buffer" your commands. **Run** the code in the editor is equivalent to type the command line by line into command window.

Interface

Current Folder (& Environment)

It indicates which folder you are currently. When you operate on file (e.g. call a function from another file), MATLAB will search the file in current directory or search path. Personally, I suggest managing your file in the current directory, which will make your life easier.
For more details about search path, you can refer to
`https://ww2.mathworks.cn/help/matlab/matlab_env/what-is-the-matlab-search-path.html?lang=en`.

Outline

Operators & Keys

- "+", "-", "*", "/", ".*", "./", " ^", " .^": matrix or elementwise arithmetic
- "=": assignment
- "==": equal
- "&&", "||": logicial and / or
- "%": comment. Will be ignored by MATLAB. Make the code easier to understand
- ";": suppress output. Please add it to every line that you do not expect output! Extra output may result in dection
- "↑, ↓": view history
- Tab: indent
- "clc": clear command window
- "clear / clear all": clear workspace

Some Special Variables & Constants

Tips

All variables in MATLAB are matrix. Scalar is essential a 1×1 matix, but it allows some operations that may be invalid for a 1×1 matrix.

- "ans": default output variable; do not use "ans" as your own variable name in your script
- "i", "j": virtual number
- "Inf": infinite
- "NaN": not a number; mostly appear when the result exceed some limit or some errors occur
- "pi"

save & load

- save xxx: save all current variables into .mat file
- load xxx: load .mat file; will not clear variables alread in workspace; will cover the variable in workspace if two variables share same name

Command

- "exist": test the existence of a varibale or a file (exist xxx)
- "global": declare global variable
- "help", "doc": show document (help xxx; doc xxx)
- "clc": clear command window
- "clear / clear all": clear workspace

It's good habit to add "clc" and "clear" at the first of your program, **especially in your homework**.

Control Flow

- if, while, for
- while 1

```
1  for i = 1:5
2      disp(i);
3      if i == 4
4          disp('look, it is a
               four!')
5      end
6  end
7  j = 100;
8  while j > 95
9      disp(j);
10     j = j - 1;
11 end
```

Outputs:

```
1 ↩
2 ↩
3 ↩
4 ↩
look, it is a four! ↩
5 ↩
100 ↩
99 ↩
98 ↩
97 ↩
96 ↩
```

Control Flow

- break: break the loop
- continue: ignore the later code in this iteration; skip directly to the next iteration

```
1  i = 10;
2  while 1
3      i = i - 1;
4      if i == 8 || i == 7
5          continue;
6      end
7      if i < 3
8          break;
9      end
10     disp(i);
11 end
```

Outputs:

9 ↩

6 ↩

5 ↩

4 ↩

3 ↩

M-file

Source code of MATLAB code is stored in .m files. It is essential text file, which means actually you can modify it as for .txt file. There are two kinds of .m file: script and function.

script

Scripts are essentially putting the code line by line to the command window. A script has no specific input and output. It operates on the value of workspace. Note that, if you do not clear the workspace before running a scipt, all variables in the workspace will be used by script, so of which are not intended, and may cause what you call "XUANXUE" (dark magic). [1]

function

Functions take specific input and produce output. A function only working on its own varivales, which you can take it as a separate box. It promotes **code reuse** and **decoupling**.

[1] So you see why I mention adding "clear" in the front of your code is a good habit.

M-file

Aside: Code Reuse & Decoupling[2][3]

Code reuse not only means less code you need to produce; more importantly, it means less code you need to take care (debug, modify, or performance improvement). Decoupling means we would like the code be separated into modules, and limit the coupling only to the interfaces, i.e. where modules connect.

Low coupling:

High coupling:

[2] "Aside" means the content below will be discussed in more advanced courses, but it can be quite helpful if you know it now.

[3] Reference and thanks: some of content is from `https://github.com/tripack45/VE280-Notes`.