MATLAB

Input/Output

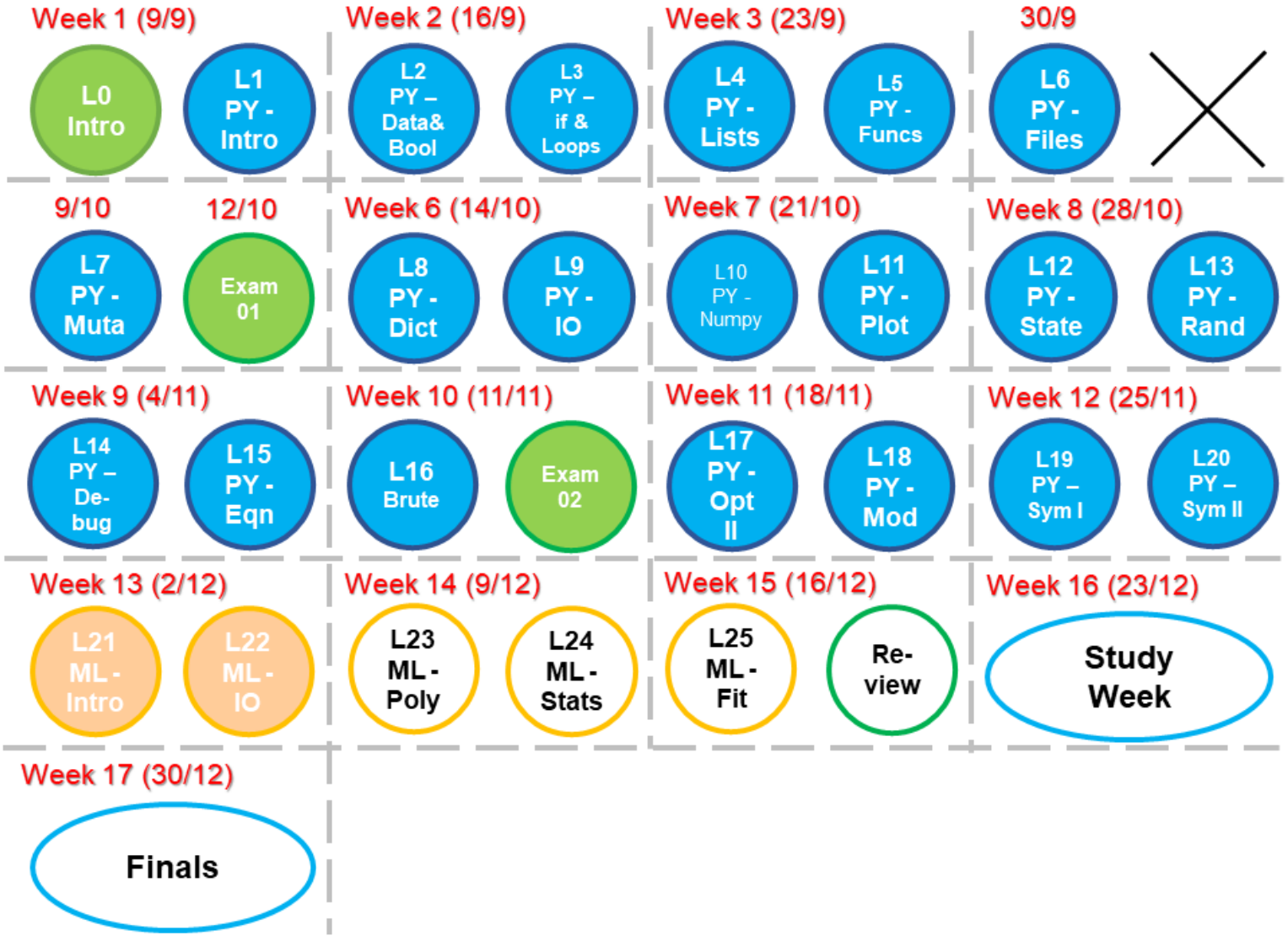
Announcements

quiz: `quiz22` due on Thurs 05/12

lab: `lab` on Fri 06/12

hw: `hw12` on matlab wbsite due Wed 11/12

Roadmap


Week 1 (9/9)
L0 Intro
L1 PY - Intro
Week 2 (16/9)
L2 PY – Data& Bool
L3 PY – if & Loops
Week 3 (23/9)
L4 PY - Lists
L5 PY - Funcs
30/9
L6 PY - Files
9/10
L7 PY - Muta
12/10
Exam 01
Week 6 (14/10)
L8 PY - Dict
L9 PY - IO
Week 7 (21/10)
L10 PY - Numpy
L11 PY - Plot
Week 8 (28/10)
L12 PY - State
L13 PY - Rand
Week 9 (4/11)
L14 PY – De-bug
L15 PY - Eqn
Week 10 (11/11)
L16 Brute
Exam 02
Week 11 (18/11)
L17 PY - Opt II
L18 PY - Mod
Week 12 (25/11)
L19 PY – Sym I
L20 PY – Sym II
Week 13 (2/12)
L21 ML - Intro
L22 ML - IO
Week 14 (9/12)
L23 ML - Poly
L24 ML - Stats
Week 15 (16/12)
L25 ML - Fit
Re-view
Week 16 (23/12)
Study Week
Week 17 (30/12)
Finals

Objectives

A. Understand multiple returns from a function.
B. Understand data sources in MATLAB, particularly `importdata`, `imread`, and `webread`.
C. Distinguish functions and scripts.
D. Plot

Basic Review

Question

$$\begin{pmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{pmatrix}$$

How can we produce this array?

A `ones(3,3) - 2*eye(3,3)`

B `ones(3,3) + 2*eye(3,3)`

C `2*ones(3,3) + eye(3,3)`

D `2*ones(3,3) - eye(3,3)`

Question

$$\begin{pmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{pmatrix}$$

How can we produce this array?

A `ones(3,3) - 2*eye(3,3)`

B `ones(3,3) + 2*eye(3,3)`

C `2*ones(3,3) + eye(3,3)`

D `2*ones(3,3) - eye(3,3)` ⋆

Other stuff

Most variables are created as a *double* (i.e., *long float*)

Can type cast; To integer `int8`(x) or `uint8`(x); or 16 or 32 or 64

A lot of functions are not covered as they are similar to Python

So you need to search online or use `doc` or `help`

Arrays Redux

Basics

```
a = [ 1 2 3 ]; %row vector
b = [ 1 2 3 ]'; %column vector
A = [ 1 2 3 ; 4 5 6 ]; %matrix
```

Indexing arrays

We can index arrays with arrays.

```
A = 0:10:100;
B = A( [ 5,9,2,2 ] );
```

Indexing arrays

We can index arrays with arrays.

```
A = 0:10:100;
B = A( [ 5,9,2,2 ] );
```

Ans:

A = 0 10 20 30 40 50 60 70 80 90 100

B = 40 80 10 10

Indexing arrays

We can index arrays with arrays.

```
A = 0:10:100;
B = A( [ 5,9,2,2 ] );
```

Ans:
A = 0 10 20 30 40 50 60 70 80 90 100
B = 40 80 10 10

We can also slice.

```
A = 0:10:100;
B = A( 4:7 );
```

Indexing arrays

In more dimensions:

```
A = [ 1,2,3 ; 4,5,6 ; 7,8,9 ];
B = A( 1:2,1:2 );
C = A( :,1:2 );
```

Indexing arrays

In more dimensions:

```
A = [ 1,2,3 ; 4,5,6 ; 7,8,9 ];
B = A( 1:2,1:2 );
C = A( :,1:2 );
```

ans =

```
B =
     1     2
     4     5
C =
     1     2
     4     5
     7     8
```

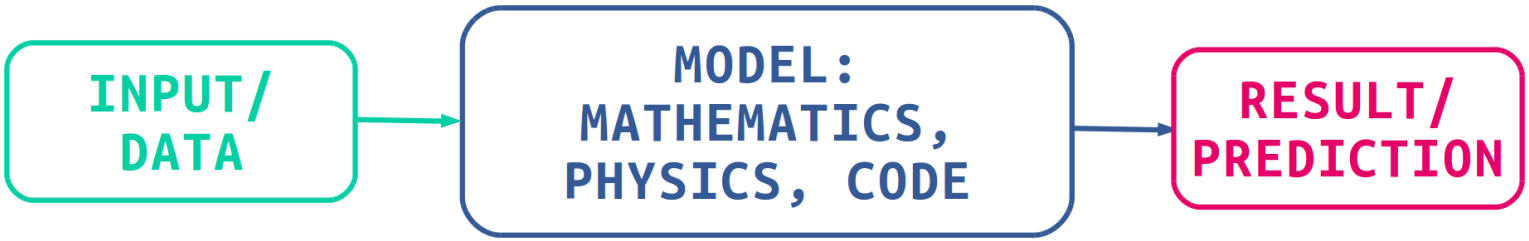
Indexing arrays

What are the differences in these? Why?

```
A = [ 1,2,3 ; 4,5,6 ; 7,8,9 ];
A( 2 )
A( 2,: )
A( :,2 )
```

Data Processing

Modeling


INPUT/
DATA
MODEL:
MATHEMATICS,
PHYSICS, CODE
RESULT/
PREDICTION

File Input

File I/O - Save

MATLAB encourages the storage of complicated variables, such as the results of numerical calculations, as ‘mat‘ files.

Saving data uses `save`:

File I/O - Save

MATLAB encourages the storage of complicated variables, such as the results of numerical calculations, as ‘mat‘ files.

Saving data uses `save`:

```
A = [ 1 2 3 ; 4 5 6 ];
B = [ 3 4 5 ; 1 2 8 ];
save( 'test', 'A' );  %save only A into test.mat
```

or

```
save( 'test');  %save everything in Workspace into
                %test.mat
```

File I/O - Save

MATLAB encourages the storage of complicated variables, such as the results of numerical calculations, as ‘mat‘ files.

Saving data uses `save`:

```
A = [ 1 2 3 ; 4 5 6 ];
B = [ 3 4 5 ; 1 2 8 ];
save( 'test', 'A' );  %save only A into test.mat
```

or

```
save( 'test');  %save everything in Workspace into
                %test.mat
```

or

Use `save test.txt A -ascii -append` to append the value of `A` into a file `test.txt`

File I/O - Save

MATLAB encourages the storage of complicated variables, such as the results of numerical calculations, as ‘mat‘ files.

Saving data uses `save`:

```
A = [ 1 2 3 ; 4 5 6 ];
B = [ 3 4 5 ; 1 2 8 ];
save( 'test', 'A' );  %save only A into test.mat
```

or

```
save( 'test');  %save everything in Workspace into
                %test.mat
```

or

Use `save test.txt A -ascii -append` to append the value of `A` into a file `test.txt`

There is a slight difference between these methods. Please test in MATLAB

File I/O - Load

Use `load` to open:

```
A = load( 'test', 'A' );
```

load from `text.mat` variable `A`

File I/O - Load

Use `load` to open:

```
A = load( 'test', 'A' );
```

load from `text.mat` variable `A`

Use `imread` to open images (.jpg, .png or others):

```
A = imread( 'myPicture.jpg' );
```

File I/O - Load

A more advanced tool: `importdata`

```
dataV = importdata( 'rainfall.txt' );
```

File I/O - Load

A more advanced tool: `importdata`

```
dataV = importdata( 'rainfall.txt' );
```

Import data in file into an array, here into `dataV`

Can also be used to process CSVs, image types, etc.

File I/O - Load

A more advanced tool: `importdata`

```
dataV = importdata( 'rainfall.txt' );
```

Import data in file into an array, here into `dataV`

Can also be used to process CSVs, image types, etc.

Old process using `fopen`, `fscanf`, `fclose`, `fprintf` also common.

Web Input

Web I/O

`webread` processes data gracefully.

```
url = 'http://zjui.intl.zju.edu.cn/sites/
       default/files/ueditor/1572/upload/
       image/20180917/1537152279766332.jpg'

data = webread( url );
image( data ); %display image from an array
```

Plotting

Plotting

`plot` works identically to `plt.plot`.

`figure` creates a new figure (window for plots).

```
x = 0:.1:2*pi
y = sin( x )

figure(100) %give the figure a number
plot( x,y,'o' )
title( 'sin(x)' )
xlabel( 'x values' )
ylabel( 'y values' )
```

Plotting

`plot` works identically to `plt.plot`.

`figure` creates a new figure (window for plots).

```
x = 0:.1:2*pi
y = sin( x )

figure(100) %give the figure a number
plot( x,y,'o' )
title( 'sin(x)' )
xlabel( 'x values' )
ylabel( 'y values' )
```

⋆MATLAB also has an good plot editor. ⋆

Plotting

Other plots to use:

A. `fplot` - plot an equation
B. `plot3` - 3D plot
C. `fcontour` - plot contour
D. `subplot` - small plots within a plot

Aside on functions

You can define a single-line function locally using the syntax:

```
f = @(t) cos( 3*t );
```

Plotting

```
x = @(t) cos( 3*t );
y = @(t) sin( 2*t );
fplot( x,y )

t = 0:pi/50:10*pi;
st = sin(t);
ct = cos(t);
plot3(st,ct,t)
```

Plotting

```
f = @(x,y) sin( x ) + cos( y );
fcontour( f )

subplot(2,1,1);
x = linspace(0,10);
y1 = sin(x);
plot(x,y1)

subplot(2,1,2);
y2 = sin(5*x);
plot(x,y2)
```

Images

Images

Images can also be opened as files.

```
A = importdata( 'rabbit-bw.jpg' );
image( A );
```

Images

Images can also be opened as files.

```
A = importdata( 'rabbit-bw.jpg' );
image( A );
```

Black and white images are arrays of `0`s and `1`s.

Greyscale images are values from `0` and `1`.

Color images are three-dimensional arrays. (Why?)

Variations exist depending on the underlying data.

Other stuff

Multiple returns

Functions can return several values.

Multiple returns

Functions can return several values.

```
function [ a,b ] = nonsense( x,y )
    a = x ^ 2;
    b = y ^ 3;
end

[ q r ] = nonsense( 3,4 );
```

Multi-dimension char arrray

But be careful—sizes cause surprises.

```
A = [ 'HELLO'; 'WORLD' ];
C = [ 'HELLO'; 'WORLD!' ];

A( 2,1 )
C( 2,1 )
```

What are A and C??

Multi-dimension string array

But be careful—sizes cause surprises.

```
A = [ "HELLO"; "WORLD" ];
C = [ "HELLO"; "WORLD!" ];
A( 2,1 )
C( 2,1 )
```

What are A and C??

Summary

A. Like Python, load and read files
B. Like Matplotlib, plot different types of graphs
C. Like Python, functions with many outputs
D. Unlike Python, there are differences between ’ and ”