10/22/14 Programming II: User-Defined Functions

↳ A fcn carries out an operation

fcn: $y = f(x)$

output, fcn, input

↳ fs: sin(x), sqrt(x), length(x), exp(x)

↳ All built into Matlab

↳ Can define our own (UDFs)

↳ New tab → function fx

```
function [out] = quartic(in)
% Quartic raises input to ^4.
    out = in.^4;
end
```

tests: a = 2; x = [1:3], y = reshape([1:6], 3, 2)'

quartic(a), then x, then y.

All variables in a fcn are defined "locally"
↳ only "in" & "out" communicate externally.

↳ "global" variables can pass back & forth
↳ Must be "declared" in all fcns, etc.

Example 2:

```
function [out] = example2 (in)
% example2 calcs value of polynomial below:
out = in.^3./(in.^2+9);
end
```

tests:

a) example2(1) → 1/10

b) x = [-3:0.25:3];
plot(x, example2(x))

↑ fcn

↑ can "pass" a fcn into a fcn...

Scripts vs. Fcns!

- both end in .m; only fcns start w/ "function [ ]= ..."
- fcn variable are local
- fcns accept input arguments; output output arguments

Major!: ## Anonymous Fcns

name = @ (arg1, arg2, ...) expression

ex. 1:

quintic = @ (x) x.^5

↳ one, single matlab expression

↳ multiline ⇒ fcn

↳ @ ⇒ defines anon. fcn & assigns the name as a fcn handle (like a pointer) that can be passed to other fcns.

ex. 2,

```
cube_2d = @(x,y) x.^3 + y.^3
```

ex. 3

```
a = 1; b = 2; c = 3;
parabola = @(x) a*x.^2 + b*x + c
```

↳ definition statement captures & fixes a, b, c values

Anon-fcns = great; very handy.

Minor: Subfunctions

Inside Primary (called) fcns, can define sub-fcns to break up complex tasks, etc.

Also nested fcns, but not used here.

TH

bad name; big idea. → # Function functions

↳ fcns that can call other fcns.

ex. `x = [0: 0.01: 4*pi^2];`

`plot(x, sin(sqrt(x)))`

fcn fcn — passed fcn

`plot(x, sqrt(sin(x)))`

HW 7.2 → great fcn passing example.

Where do we put our UDFs?

↳ In a folder that is permanently added to Matlab's 'path'

↳ Can then Access UDF's from anywhere

↳ >> addpath 'FolderName'

>> savepath

↳ >> addpath(genpath('folder name'))

↳ also includes subfolders...

Other UDFs

1) "xylabel"

fcn xylabel(xstr, ystr, fs)
% writes x & y labels to a plot...
xlabel(xstr, 'fontsize', fs);
ylabel(ystr, 'fontsize', fs);
end

↳ now use in a plotting script

1b) "setup_UDF"

```
fcn setup_UDF
%Sets up start of scripts
%JMA 11/9/18
    clc;
    clear;
    close all;
end
```

2) "sine2line"

```
fcn [y] = sine2line(k, x, xbreak)
% sine then switches to line
% k = "wavenumber" of sine wave
% x = x-array
% xbreak = break point to flat line

    lv_sine = (x <= xbreak);

    lv_line = (x > xbreak); % not lv_sine

y = sine(k*x) * lv_sine + ...
    sin(k*xbreak) * lv_line;

plot(x, y, '-k', 'linewidth', 1.5)
xylabel('X', 'Y', 18)
end
```