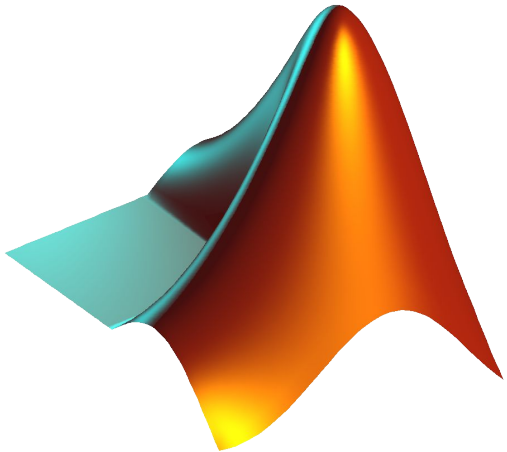
CS 1112 Introduction to Computing Using MATLAB



Instructor: Dominic Diaz

Website: https://www.cs.cornell.edu/courses/cs1112/2022fa/

Today: Functions

Agenda and announcements

- Last time
  - Loops, nested loops, and graphics
- This time
  - Functions
- Announcements
  - Project 2 due 9/19!
    - If you work with someone else: one person forms the group on CMS, the other person accepts the invite, then submit your files.
    - Many office hours between now and due date (my OH 2:30 on Monday in Gates 417)

Things to watch out for project 2

- Always start scripts with a comment describing what the code does
- Comment your code (but not excessively)
- Indent code nested inside if statements, for loops, and while loops
- Only use the input function when the documentation indicates that there is a user-input.
  - For example, in project 2 problem 1: There should be 2 user-inputs
- Do not put semicolon at the end of `if`, `elseif`, `else`, `for`, `while`, `end`, and `function` lines.
  - In other words, any lines of code that start with any of these keywords should not end with a semicolon

```
k = 1; n = 10;
while k <= n
    disp(k);
    k = k + 1;
end
```

Functions

max(endPt, cricitalPt)

rand

min(4, pi)

plot(x, y, 'bo')

sin(x)

fprintf('MATLAB is cool')

rem(10, 6)

DrawDisk(1, 2, 2, 'y')

input('Input a number:')

disp(numCats)

DrawRect(2, 3, 1, 'c')

DrawStar(7, 2, -.5, 'y')

title('Trajectory of a golf ball')

Functions

- Many built-in functions in MATLAB
  - General math: min, max, abs, rem, …
  - Trigonometry: sin, cos, tan, asin, …
  - Integer computation: floor, ceil, round, …
  - Plotting: plot, title, xlabel, ylabel, …
  - Input/output: fprintf, sprintf, disp, …
  - …
- We can add our own user-defined functions!
  - Goals for user-defined functions:
    - Should be able to specify input
    - Should have output or do something useful
    - Should be simple to use
    - Should make your scripts more manageable

Function: a group of statements that together perform a task.

- We can write our own functions (user-defined functions) to perform a specific task
  - Example: Draw a rectangle with specified coordinates, length, width, and color. (DrawRect.m from lecture 7.)
  - Example: Generate a random number in a specified interval (recall back to problem 2 on project 1–randomly placing a the light source)
  - Example: Convert polar coordinates to x-y (Cartesian) coordinates

```
function DrawRect(a,b,L,W,c)
% Adds a rectangle to the current window. Assumes hold is on.
% The rectangle has vertices (a,b), (a+L,b), (a+L,b+W), and (a,b+W) and
color c
% where c is either an rgb vector or one of the built-in colors 'r', 'g',
% 'y', 'b', 'w', 'k', 'c', or 'm'.

x = [a a+L a+L a  ];
y = [b b   b+W b+W];
fill(x,y,c)
```

How to write your own user-defined function

```
function [outputs] = FunctName(inputs)
```

Keyword (tells the computer that the we are defining a function now)

Output parameter list
- Single output does not require [ ]
- Multiple parameters are comma-separated and enclosed in [ ]

Name of the function. Should be concise but give you an idea of what the function does.
- If you are creating a function file, the file name should be the same as the function name.

Input parameter list enclosed in ( ).
- Multiple parameters are comma-separated
- ( ) can be empty, i.e. no inputs

Example of a useful user-defined function

The function *definition*:

```
function [x, y] = Polar2xy(r, theta)
% Convert polar coordinates (r, theta) to cartesian coordinates (x,y).
% theta is in degrees.
rads = theta*pi/180;
x = r*cos(rads);
y = r*sin(rads);
```

Using the function (in a script, the command window, or another function):

```
% Convert polar (r1, theta1) to Cartesian (x1, y1)
r1 = 1; theta1 = 30;
[x1, x2] = Polar2xy(r1, theta1);
plot(x1, x2, 'b*');
```

The function *call*

Function *header* defines how the function is called

The function *definition*:

```
function [x, y] = Polar2xy(r, theta)
% Convert polar coordinates (r, theta) to cartesian coordinates (x,y).
% theta is in degrees.
rads = theta*pi/180;
x = r*cos(rads);
y = r*sin(rads);
```

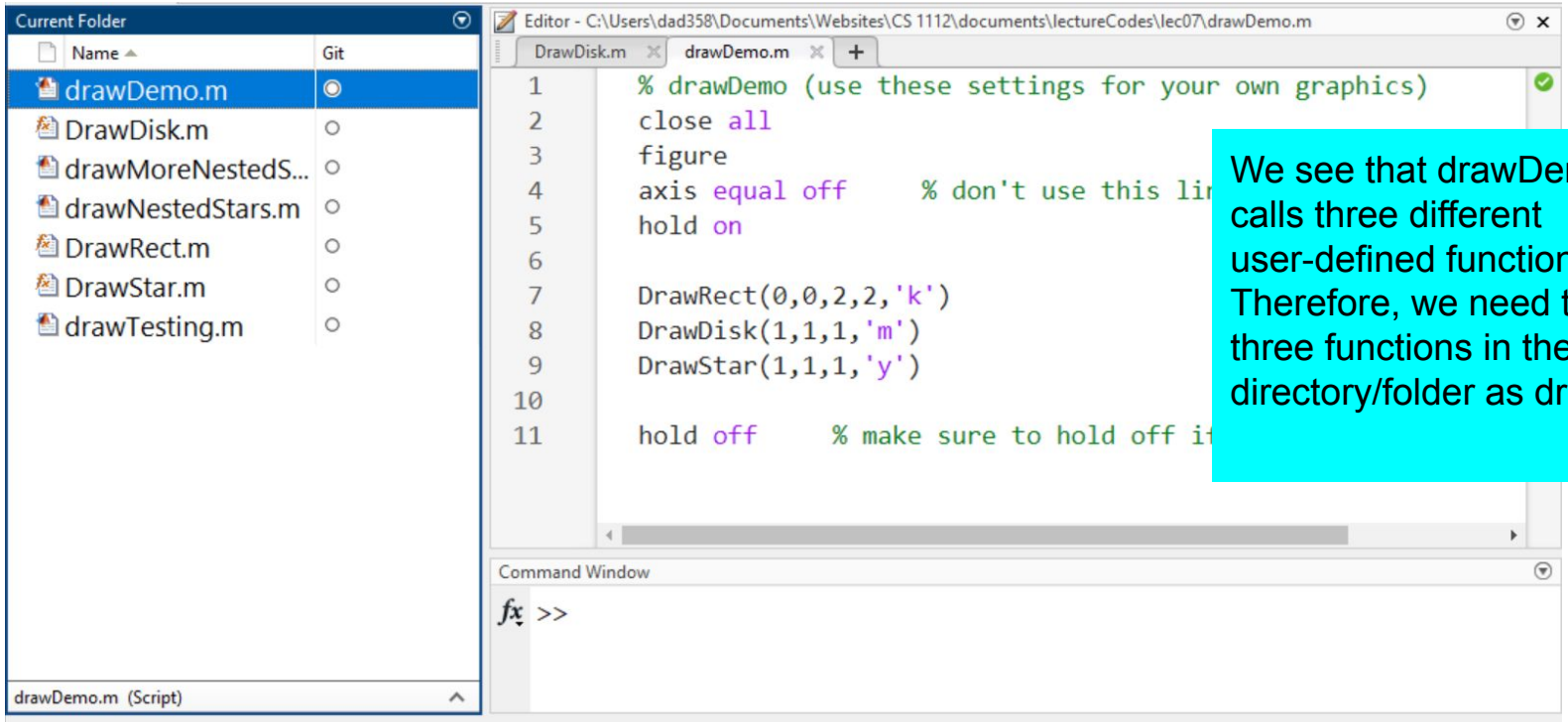
Inputs and outputs can have same/different names as in function header.

Using the function:

```
% Convert polar (r1, theta1) to Cartesian (x1, y1)
r1 = 1; theta1 = 30;
[x1, x2] = Polar2xy(r1, theta1);
plot(x1, x2, 'b*');
```

Accessing functions

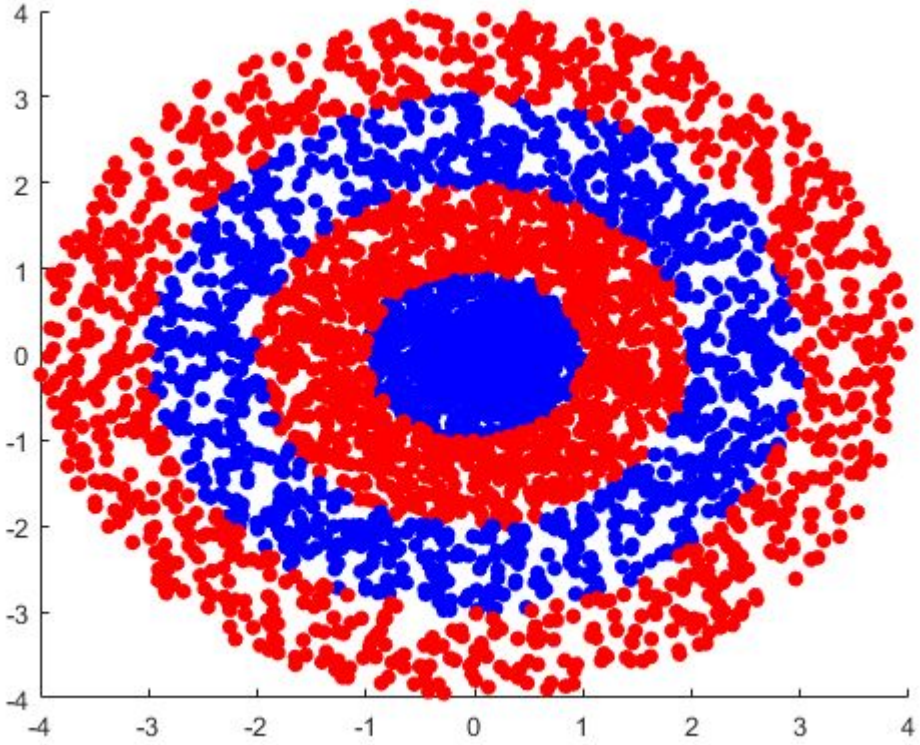
For now*, if your script calls a function, make sure they are in the same directory.



We see that drawDemo.m calls three different user-defined functions. Therefore, we need those three functions in the same directory/folder as drawDemo.

*you can get around this by using the MATLAB path (but you won't need to know this for CS 1112).

Ex: Draw a bullseye figure with randomly placed dots



Task: draw d random dots in each of c concentric rings (let d and c both be user inputs). Each ring should have “radius” 1.

Example, in the left image

d = 1000

c = 4

Helpful user-defined functions

Before we jump into solving this problem, let's look at some user-defined functions.

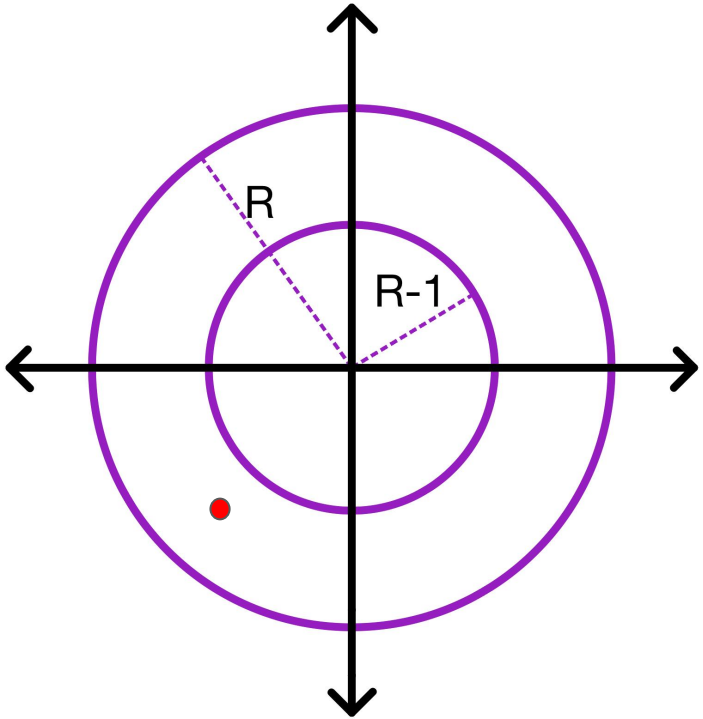
```
function [x, y] = Polar2xy(r, theta)
% Convert polar (r, theta) to cartesian (x,y).
% theta is in degrees.

rads = theta*pi/180;
x = r*cos(rads);
y = r*sin(rads);



function DrawColorDot(x, y, color)
% Draw a dot on at position(x,y).
% In red if color=0, otherwise blue.
if (color==0)
   plot(x,y,'r.','markersize',20)
else
   plot(x,y,'b.','markersize',20)
end
```

Notes on functions:

- After the function header but before the code, provide comments describing what the function does, the inputs, and the outputs.
- Some functions have no outputs, like the second function.

Building the algorithm

First let's figure out how to draw one dot between radius R and R-1.



Pseudocode to plot a dot randomly within radii R and R-1:

Choose a random r between R and R-1
Choose a random angle between 0 and 360
Convert the polar coordinates to cartesian coordinates
Plot a circle

Pseudocode to plot a dot randomly within radii R and R-1 and use color c:

Choose a random r between R and R-1
Choose a random angle between 0 and 360
Convert the polar coordinates to cartesian coordinates
Plot a circle of color c

Convert the pseudocode to actual code!

```
radius = rand + R-1;
theta = rand*(360);
[x, y] = Polar2xy(radius, theta);
DrawColorDot(x, y, _________)
```

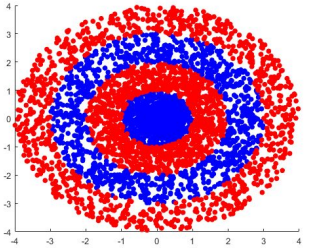
Say we have the following functions:

```
function [x, y] = Polar2xy(r, theta)
% Convert polar to cartesian.
% theta is in degrees.
rads = theta*pi/180;
x = r*cos(rads);
y = r*sin(rads);
```

```
function DrawColorDot(x, y, color)
% Draw a dot on at position(x,y). In
% red if color=0, otherwise blue.
if (color==0)
   plot(x,y,'r.','markersize',20)
else
   plot(x,y,'b.','markersize',20)
end
```

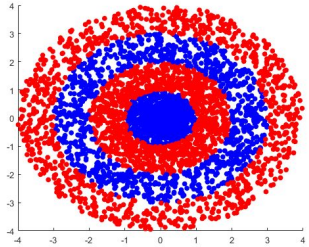
```
% Draw d random dots in each of c concentric rings
_____________________________________________
_____________________________________________


_______________
_______________
_______________
_______________

% Put dots in the area between circles with radii R and (R-1)
_____________
    % Draw d dots
    _____________
        r = rand + R-1;
        theta = rand*(2*pi);
        [x, y] = Polar2xy(r, theta);
        DrawColorDot(x, y, ________);
    end
end
_________
```

Task: draw d random dots in each of c concentric rings (let d and c both be user inputs). Each circle should have radius 1. The innermost circle should have blue dots then alternate colors between blue and red.



```
% Draw d random dots in each of c concentric rings
c = input('How many concentric rings? ');
d = input('How many dots in each ring? ');

close all
figure
axis equal off
hold on

% Put dots in the area between circles with radii R and (R-1)
for R = 1:c
    % Draw d dots
    for dotNum = 1:d
        r = rand + R-1;
        theta = rand*(2*pi);
        [x, y] = Polar2xy(r, theta);
        DrawColorDot(x, y, rem(R,2));
    end
end
hold off
```

```
% Draw d random dots in each of c concentric rings
c = input('How many concentric rings? ');
d = input('How many dots in each ring? ');

close all
figure
axis equal off
hold on

% Put dots in the area between circles with radii R and (R-1)
for R = 1:c
    % Draw d dots
    for dotNum = 1:d
        r = rand + R;
        theta = rand*(2*pi);
        [x, y] = Polar2xy(r, theta);
        DrawColorDot(x, y, rem(R,2));
    end
end
hold off
```



Why `rem(R, 2)`?

If R = 1, `rem(R,2) = 1` =>color blue
If R = 2, `rem(R,2) = 0` =>color red
If R = 3, `rem(R,2) = 1` =>color blue
If R = 4, `rem(R,2) = 0` =>color red

```
function DrawColorDot(x, y, color)
% Draw a dot on at position(x,y). In
% red if color=0, otherwise blue.
if (color==0)
    plot(x,y,'r.','markersize',20)
else
    plot(x,y,'b.','markersize',20)
end
```