MA2252 Introduction to Computing

Lecture 6: Functions (contd.)

Sharad Kumar Keshari

School of Computing and Mathematical Sciences

University of Leicester

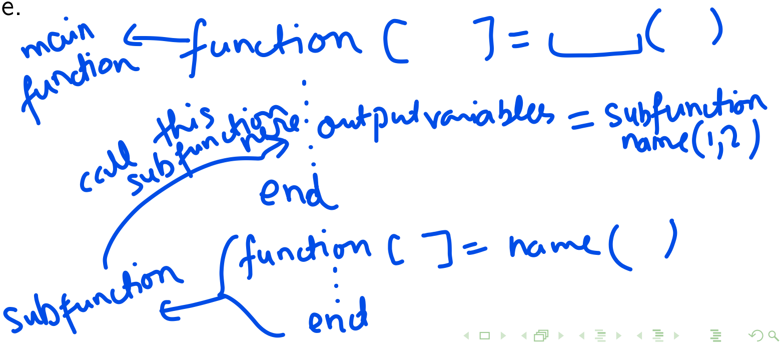
Learning outcomes

At the end of lecture, students will be able to understand and create

- Subfunctions
- Function handles
- Script files

Subfunctions

A subfunction is a function defined under a parent/main function in the same .m file.



```
function [   ] = ____(   )
    :
    : outputvariables = subfunction name(1,2)
    :
end
function [ ] = name(   )
    :
    :
end
```

Subfunctions (contd.)

$ax^2 + bx + c = 0$

$\downarrow$

x1 , x2

Example:

```
function [x1,x2] = myrootsfun(a,b,c)

%This function calculates roots of quadratic equation with
%coefficients a, b and c.
%Sharad
%28-09-2022

D = mydiscriminant(a,b,c);     %call the mydiscriminant subfunction
x1=(-b+sqrt(D))/(2*a);
x2=(-b-sqrt(D))/(2*a);

end
```

→ $D = b^2 - 4ac$

→ subfunction

main function variables → (D, a, b, c, x_1, x_2)

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$x = \frac{-b \pm \sqrt{D}}{2a}$$

D → discriminant

Subfunctions (contd.)

```
function [disc] = mydiscriminant(A,B,C)
```

(disc, A, B, C)

```
%This subfunction calculates the discriminant D.

disc=Bˆ2-4*A*C;

end
```

Subfunctions (contd.)

Note:

only parent function is allowed to call its subfunction

- Only parent function can call its subfunction.
- Subfunction retains a separate workspace from its parent function.

Subfunctions (contd.)

Demo

Activity

Consider again the subfunction mydiscriminant(A,B,C). What happens when you type

DISC=mydiscriminant(1,-7,10) in command window and hit ‘Enter’ ?

$$DISC = (-7)^2 - 4 \times 1 \times 10$$
$$= 49 - 40 = 9$$

Let’s do a mentimeter poll!

Activity (contd.)

Please go to the link `https://www.menti.com/al9p1z6skq4f` provided in chat

or

visit `https://www.menti.com` and enter the code **39701413**

Function handles

x, y, z

A function handle is a variable which stores some function.

Construction:

1. Using built-in functions: function_handle=@function_name

   assignment operator

   function's name you want to store in your function handle

   name of your function handle

   Example: F=@sin

   function handle

   Here, variable F is function handle which stores sine function.

Function handles (contd.)

→ user-defined function

2. Using anonymous function:

function_handle=@(input variables) function_definition

→ definiti

↘ variables which your function contains

Examples:

- reciprocal=@(x) 1/x

  $reciprocal = \frac{1}{x}$ $reciprocal(10) = \frac{1}{10} = 0.1$

- myimplicit=@(x,y) x+y+x*y

  $myplicit \doteq x+y+xy$

- combinations=@(n,r) factorial(n)/(factorial(r)*factorial(n-r))

  $myimplicit(1,2) = 1+2+2 = 5$

no. of ways to choose r objects from n objects ← $^nC_r = \frac{n!}{r!(n-r)!}$

$combinations(5,2) = 5!/2!3! = \frac{5\times\overset{2}{\cancel{4}}\times\cancel{3!}}{\cancel{2!}\cdot\cancel{3!}} = 10$

Function handles (contd.)

F = @sin

F(pi/2)

sin(x)

sin~~(x,y~~)

Demo

Function handles (contd.)

f = ~~combinations~~ is combinations.

Using function handle to pass a function to other functions

Example: f=@sin; g=@cos; → sumfun is a function which calculates $f(x)+g(x)$

→ f,g,h

$f = \sin x$
$g = \cos x$

function handles

$x = \pi/2$

$\sin \pi/2 + \cos \pi/2$
$= 1 + 0$
$= 1$

```
function [y] = sumfun(f,g,x)

%This function calculates the sum f(x)+g(x) for any two given
%functions f and g.

y=f(x)+g(x);

end
```

$f(x) * g(x)$

$f \circ g(x) = f(g(x))$

Y = sumfun(@sin,@cos,x)

Y = sumfun(f,g,Pi/2)

Function handles (contd.)

Demo

Function handles (contd.)

sin, sin

F

Using Function Handles

Pros

1. To use function as a variable whenever needed.
2. No need to write a .m file to define your function.
3. To pass function as input to other functions.

~~fa~~
header
body
end

Function handles (contd.)

$\underbrace{[x1, x2]}_{2 \text{ outputs}} = \text{my roots fun}(a,b,c)$

Cons

1. Only works with functions with one output.
2. Only useful when your function has a simple definition.

   $\hookrightarrow$ better make a .m file

Script files

Script file is a .m file where you write your code.

- To open a script file, click ‘New Script’ in HOME menu.
- To save, click ‘Save’ in EDITOR menu.
- To run the script file, click ‘Run’ in EDITOR menu.

Script files (contd.)

Example:

```
%This script file calculates the roots of a quadratic equation with
%coefficients a,b and c.
clc
clear all
a=1;
b=-7;
c=10;
D = b^2-4*a*c;     %calculate the discriminant D
x1=(-b+sqrt(D))/(2*a);     %calculate first root x_1
x2=(-b-sqrt(D))/(2*a);     %calculate second root x_2
```

→ good idea to start with clc & clean all

clc: clean the screen (command window)

clean all: to clean previously stored variables

a=1

b=-7

c=10

D=9   x1=5

x2= 2

Script files (contd.)

Demo

Script files (contd.)

Script files vs Functions

- Script files share their workspace with command window workspace. Functions have their own workspace.
- Script files are used for specific task. Functions are useful when the same task has to be done for different inputs.

→ goal → which doesn't need changing

your inputs

with functions you can change inputs

End of Lecture 6

Please provide your feedback ▸ here