Source: <https://www.pluralsight.com/courses/matlab-fundamentals>

# 3. Data Types

## 3.1 Numeric Type

>> x = 5

x =

5

>> class(x)

ans =

double

>> whos x

Name Size Bytes Class Attributes

x 1x1 8 double

>> class(x)

ans =

double

>> y = single(5)

y =

single

5

>> class(y)

ans =

single

>> whos x

Name Size Bytes Class Attributes

x 1x1 8 double

>> whos y

Name Size Bytes Class Attributes

y 1x1 4 single

>> realmin('single')

ans =

single

1.1755e-38

>> realmax('double')

ans =

1.7977e+308

>> z = uint64(5)

z =

uint64

5

>> whos z

Name Size Bytes Class Attributes

z 1x1 8 uint64

>>

## 3.2 Inf, Nan and Other Constants

>> 1/0

ans =

Inf

>> inf

ans =

Inf

>> isinf(ans)

ans =

logical

1

>> whos inf

>> inf

ans =

Inf

>> whos ans

Name Size Bytes Class Attributes

ans 1x1 8 double

>>

>> 0/0

ans =

NaN

>> inf/inf

ans =

NaN

>> isnan(ans)

ans =

logical

1

>> pi

ans =

3.1416

>> whos pi

>> pi

ans =

3.1416

>> whos ans

Name Size Bytes Class Attributes

ans 1x1 8 double

>> format rat

>> pi

ans =

355/113

>> format longg

>> i

ans =

0 + 1i

>> j

ans =

0 + 1i

>> sqrt(-5:5)

ans =

Columns 1 through 2

0 + 2.23606797749979i 0 + 2i

Columns 3 through 4

0 + 1.73205080756888i 0 + 1.4142135623731i

Columns 5 through 6

0 + 1i 0 + 0i

Columns 7 through 8

1 + 0i 1.4142135623731 + 0i

Columns 9 through 10

1.73205080756888 + 0i 2 + 0i

Column 11

2.23606797749979 + 0i

>> i

ans =

0 + 1i

>> whos ans

Name Size Bytes Class Attributes

ans 1x1 16 double complex

## 3.3 Numeric Output Formats

>> 10^10

ans =

1.0000e+10

>> format longg

>> ans

ans =

10000000000

>> format bank

>> pi

ans =

3.14

>> format hex

>> 123

ans =

405ec00000000000

>> format

>> format rat

>> 2\*3/7+1/12

ans =

79/84

>> pi/2

ans =

355/226

>> format

## 3.4 Character Strings

>> s = 'hello'

s =

hello

>> 'let''s go'

ans =

let's go

>> whos s

Name Size Bytes Class Attributes

s 1x5 10 char

>> s(1)

ans =

h

>> whos ans

Name Size Bytes Class Attributes

ans 1x1 2 char

>> a = 'hello'

a =

hello

>> b = 'world'

b =

world

>> [a ', ' b '!']

ans =

hello, world!

>> age = 30

age =

30

>> ['I am ' num2str(age) ' years old']

ans =

I am 30 years old

>> disp(ans)

I am 30 years old

>> dsolve('Dq = q\*r', 'q(0) = q0', 't') % pass to toolbox as string for evaluation

ans =

q0\*exp(r\*t)

>> strcmpi('ABC', 'abc') % compare two strings (case insensitive)

ans =

logical

1

## 3.5 Structures

>> person.name = 'john'

person =

struct with fields:

name: 'john'

>> person.age = 22

person =

struct with fields:

name: 'john'

age: 22

>> person.age

ans =

22

>> person.name

ans =

john

>> whos person

Name Size Bytes Class Attributes

person 1x1 368 struct

>> address.housename = 123

address =

struct with fields:

housename: 123

>> address.streetname = 'london road'

address =

struct with fields:

housename: 123

streetname: 'london road'

>> person.address = address

person =

struct with fields:

name: 'john'

age: 22

address: [1×1 struct]

## 3.6 Cell Arrays

>> x = 42

x =

42

>> whos x

Name Size Bytes Class Attributes

x 1x1 8 double

>> x(2) = 24

x =

42 24

>> whos x

Name Size Bytes Class Attributes

x 1x2 16 double

>> address = {123, 'london road'}

address =

1×2 cell array

[123] 'london road'

>> person = {'john', 22, address}

person =

1×3 cell array

'john' [22] {1×2 cell}

>> whos person

Name Size Bytes Class Attributes

person 1x3 606 cell

>> person(1)

ans =

cell

'john'

>> whos ans

Name Size Bytes Class Attributes

ans 1x1 120 cell

>> person{1}

ans =

john

>> whos ans

Name Size Bytes Class Attributes

ans 1x4 8 char

## 3.7 Function Handles

>> s = @sin

s =

function\_handle with value:

@sin

>> s(pi / 2)

ans =

1

>> sumOver3 = @(x, y) (x + y) / 3

sumOver3 =

function\_handle with value:

@(x,y)(x+y)/3

>> sumOver3(1, 2)

ans =

1

>> incApply = @(x, y, f) f(x + 1, y + 1)

incApply =

function\_handle with value:

@(x,y,f)f(x+1,y+1)

>> incApply(2, 3, sumOver3)

ans =

2.3333

# 8. Object Oriented Programming

## 8.1 Classes

%{

Value class

- Similar semantics to variables

- Copying a value class copies all its memory

Handle class

- Can be shared (i.e. referenced) by more than one object or function

- Copying simply copies a handle, not the contents

- Inherits from handle

%}

>> account.balance = 0 % This only creates a structure.

% This will not provide us any method for doing anything, e.g. withdrawing money.

% Create a class Account: shown in another script.

%{

classdef Account < handle

end

%}

>> a = Account

a =

Account with no properties.

>> whos a

Name Size Bytes Class Attributes

a 1x1 8 Account

## 8.2 Properties

% Add properties to original Account class

%{

classdef Account < handle

properties

Balance = 0

end

end

%}

>> a = Account

a =

Account with properties:

Balance: 0

>> a.Balance

ans =

0

>> a.Balance = 100

a =

Account with properties:

Balance: 100

## 8.3 Methods

%{

classdef Account < handle

properties

Balance = 0

end

methods

function Deposit(obj, amount)

obj.Balance = obj.Balance + amount;

end

function Withdraw(obj, amount)

if (amount <= obj.Balance)

obj.Balance = obj.Balance - amount;

else

disp('insufficient funds')

end

end

end

end

%}

>> a = Account

a =

Account with properties:

Balance: 0

>> a.Deposit(100)

>> a.Balance

ans =

100

>> a.Withdraw(60)

>> a.Balance

ans =

40

>> a.Withdraw(100)

insufficient funds