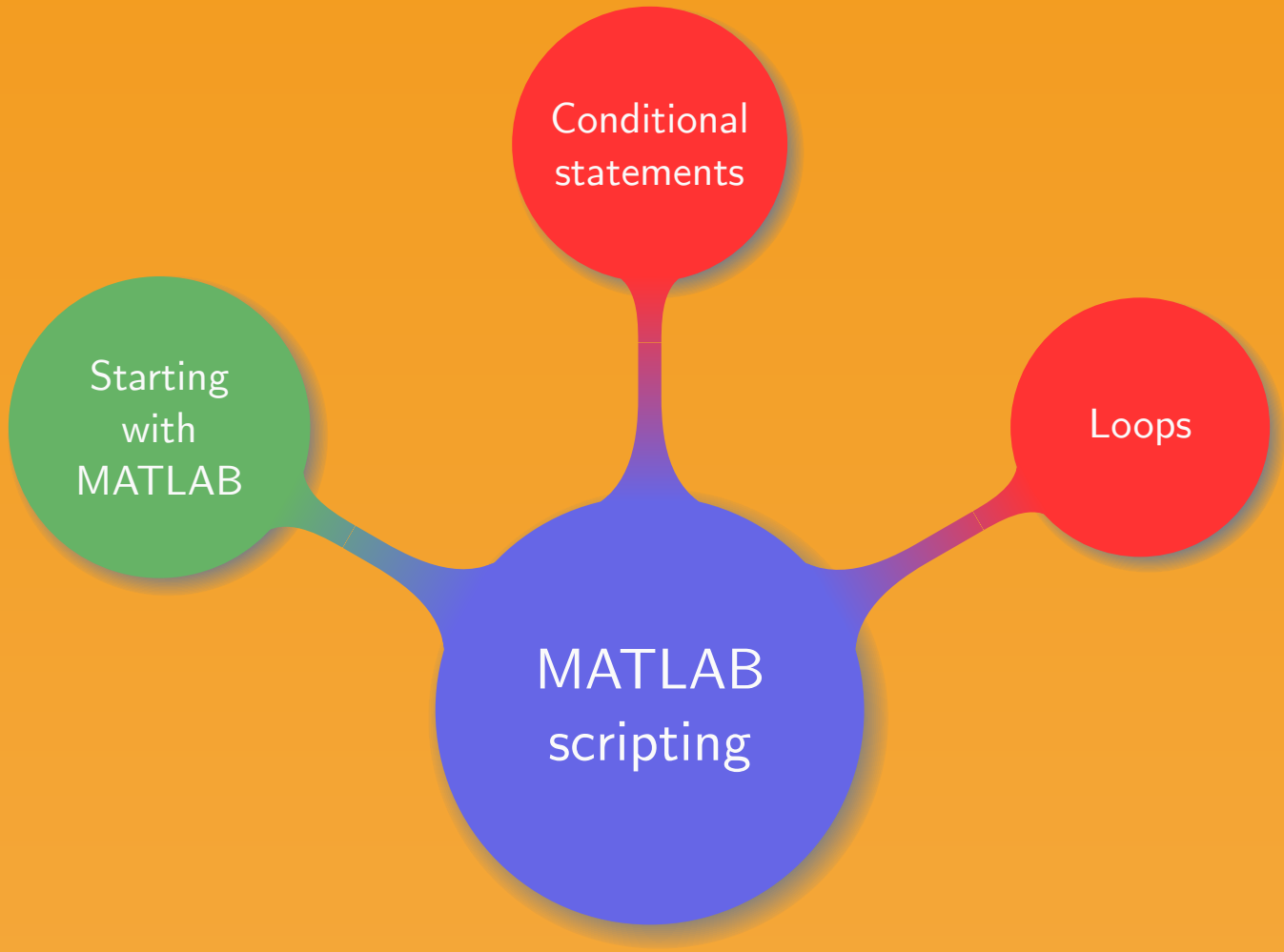
Introduction to Computer and Programming

2. MATLAB scripting

Manuel – Summer 2019

Chapter organisation



Running MATLAB

Two modes to start MATLAB:

- Desktop: graphical user interface
- Terminal: allows remote access, no mouse support

View in desktop mode:

1. Command history
2. Workspace
3. Command window
4. Help

Running MATLAB

Two modes to start MATLAB:

- Desktop: graphical user interface
- Terminal: allows remote access, no mouse support

View in desktop mode:

1. Command history
2. Workspace
3. Command window
4. Help

Files must be in the current directory or a directory listed in the path

MATLAB as a calculator

MATLAB as a simple calculator:

- Addition: `+`
- Subtraction: `-`
- Multiplication: `*`
- Power: `^`
- Right division: /
- Left division: \
- Order: `()`
- π: `pi`
- $\sqrt{-1}$: `i` or `j`
- Infinity: `Inf`

MATLAB as an advanced calculator:

- Hide the result: end the line with ";"
- Variables: must start with a letter, e.g. `a=1+2; A=3+2; a1_=4+5;`
- Comments: ignore everything after "%"
- Write two commands on a same line: separate them with a ","
- Split a line over several lines: end a partial line with "...", e.g.

```
very long line easier ...
  to read over two lines
```

Density of the Sun

MATLAB code to input in the workspace window:

```
r=1.496*10^11; c=4.379*10^9; G=6.674*10^-11;
T=365*24*3600;
V=4*pi/3*(c/(2*pi))^3;
M=4*pi^2*r^3/(G*T^2);
M/V
```

Density of the Sun

MATLAB code to input in the workspace window:

```
r=1.496*10^11; c=4.379*10^9; G=6.674*10^-11;
T=365*24*3600;
V=4*pi/3*(c/(2*pi))^3;
M=4*pi^2*r^3/(G*T^2);
M/V
```

Understanding the code:

- How are variables named and used?
- Could the code be shorter?

M-File

MATLAB script:

- Write the code in a file and load it
- Variables are added to the workspace
- To avoid variable conflicts use: `clear`, `clear all`, `clc`
- Add *cell breaks* to debug the code

M-File

MATLAB script:

- Write the code in a file and load it
- Variables are added to the workspace
- To avoid variable conflicts use: `clear`, `clear all`, `clc`
- Add *cell breaks* to debug the code

Exercise.
Write a script which prompts the user for two numbers, stores their sum in a variable, and displays the result.

M-File

MATLAB script:

- Write the code in a file and load it
- Variables are added to the workspace
- To avoid variable conflicts use: `clear`, `clear all`, `clc`
- Add *cell breaks* to debug the code

Exercise.
Write a script which prompts the user for two numbers, stores their sum in a variable, and displays the result.

```
1 clear all, clc;
2 number1=input('Input a number: ');
3 number2=input('Input a number: ');
4 numbers=number1+number2;
5 disp(numbers);
```

Generating arrays and matrices

Arrays are of a major importance in MATLAB

Generating arrays and matrices:

- Obtain a sequence of numbers: `a:b` or `a:b:c`
- Concatenate (join) elements: `[ ]`
- Define a 1-dimensional array: `[a:b]` or `[a:b:c]`
- Define a 2-dimensional array: `[a b c; d e f;]`
- Get n equidistant elements in $[a, b]$: `linspace(a, b, n)`
- Get an $n \times m$ array of 0: `zeros(n,m)`
- Get an $n \times m$ array of 1: `ones(n,m)`

Array and matrices

Explain each of the following commands:

```
clear all
a=magic(5)
a=[a;a+2], pause
a(:,3)=[]
a(:,3)=5
a(7,3), pause
```

```
a=reshape(a,5,8)
a', pause
sum(a)
sum(a(:,1))
sum(a(1,:))
```

Array and matrices

Explain each of the following commands:

```
clear all
a=magic(5)
a=[a;a+2], pause
a(:,3)=[]
a(:,3)=5
a(7,3), pause
```

```
a=reshape(a,5,8)
a', pause
sum(a)
sum(a(:,1))
sum(a(1,:))
```

Difference between arrays and matrices:

- Arrays:
  - Processed element by element
  - Add a "." in front of each operation, e.g. `.*`
- Matrices:
  - Default operations
  - Conjugate transpose: `'`
  - Determinant: `det`
  - Inverse: `inv`
  - Eigenvalues: `eig`

Basic operations

Explain each of the following commands:

```
A = [2 7 9 7 ; 3 1 5 6 ; 8 1 2 5]
A(:,[1 4]), pause
A([2 3],[3 1]), pause
reshape(A,2,6), pause
A(:), pause
flipud(A), pause
fliplr(A), pause
[A A(:,end)], pause
A(1:3,:), pause
[A ; A(1:2,:)], pause
sum(A),pause
sum(A'), pause
sum(A,2), pause
[ [ A ; sum(A) ] [ sum(A,2) ; sum(A(:)) ] ], pause
A.'
```

Accessing elements in a matrix

Given a matrix, elements can be accessed by:

- Coordinates: use the `(row,column)` position
- Indices:
  - Use a single number representing a position
  - The top left element has index 1
  - The bottom right "number of elements"

Accessing elements in a matrix

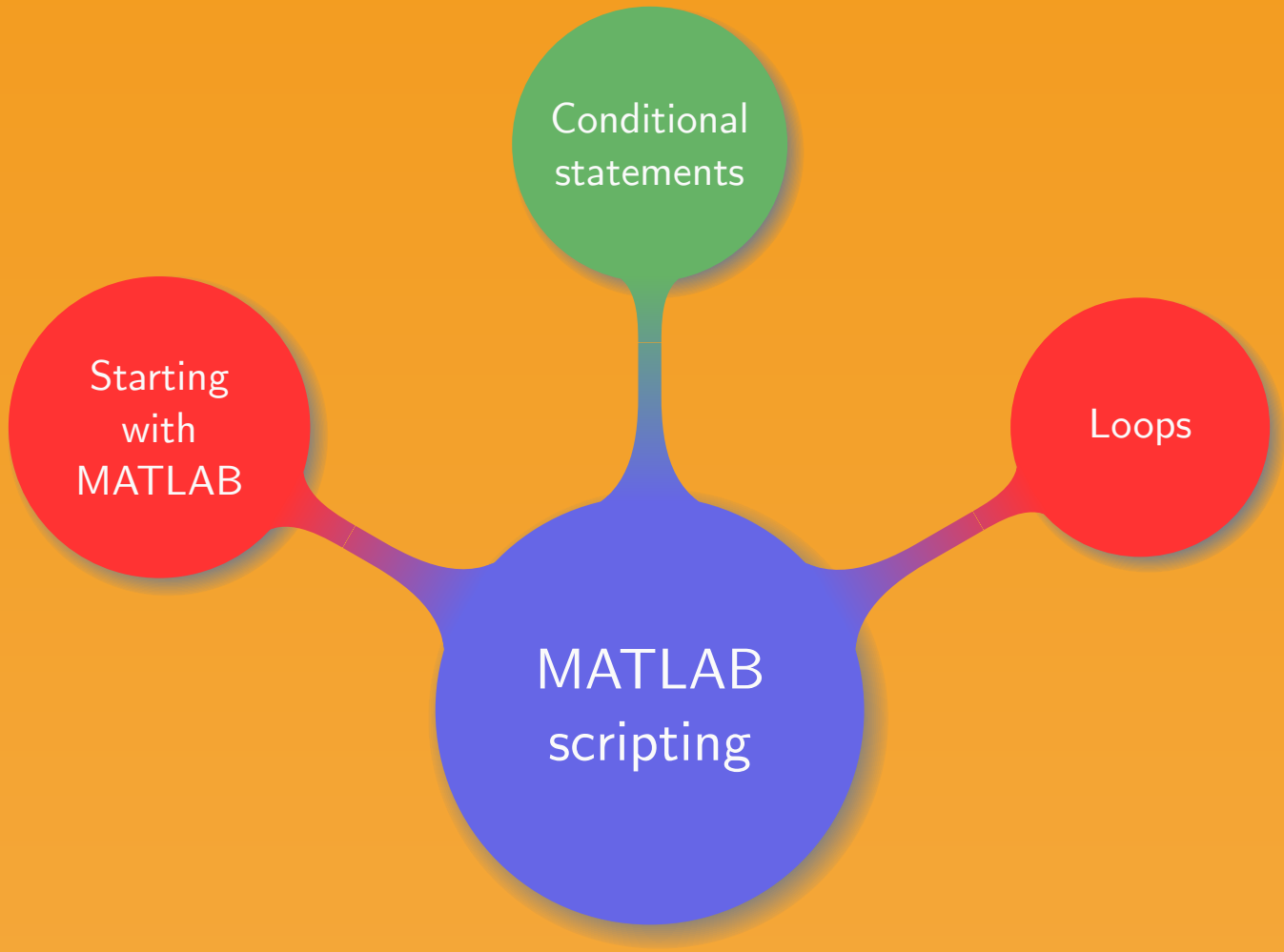
Given a matrix, elements can be accessed by:

- Coordinates: use the `(row,column)` position
- Indices:
  - Use a single number representing a position
  - The top left element has index 1
  - The bottom right "number of elements"

Example.
Explain each of the following commands:

```
A=magic(5)
A(3,2)
A(6)
numel(A)
```

Chapter organisation



Boolean logic

Run instructions based on the truth value of a given expression

Truth table for the three common operations:

A	B	$A \wedge B$	$A \vee B$	$A \oplus B$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Relational operators in MATLAB

Comparative operators:

- Less than: <
- Less or equal: <=
- Greater than: >
- Greater or equal: >=
- Equal to: ==
- Not equal to: ~=

Relational operators in MATLAB

Comparative operators:

- Less than: `<`
- Less or equal: `<=`
- Greater than: `>`
- Greater or equal: `>=`
- Equal to: `==`
- Not equal to: `~=`

Logical operators:

- And: `&`
- Or: `|`
- Not: `~`
- Xor: `xor(·,·)`

Relational operators in MATLAB

Comparative operators:

- Less than: `<`
- Less or equal: `<=`
- Greater than: `>`
- Greater or equal: `>=`
- Equal to: `==`
- Not equal to: `~=`

Logical operators:

- And: `&`
- Or: `|`
- Not: `~`
- Xor: `xor(·,·)`

Short-circuit operators:

- Evaluate expression `B` if and only if `A` is true: `A && B`
- Evaluates expression `B` only if `A` is false: `A || B`

The `if` and `switch` statements

If it rains, then I take my umbrella

```
if expression1
  statements1
elseif expression2
  statements2
  else
  statements
end
```

The `if` and `switch` statements

If it rains, then I take my umbrella

```
if expression1
  statements1
elseif expression2
  statements2
  else
  statements
end
```

```
switch variable
  case value1
    statements1
  case value2
    statements2
  otherwise
    statements
end
```

When it rains, I take my umbrella, and my hat when it's sunny

Simple application scripts

Example.

```
1  exist('./file') & load('./file')
2  exist('./file') && load('./file')
3  k=input('Press a key: ','s');
4  if k>='0' && k<='9'
5    disp('Digit')
6  else
7    disp('Not a digit')
8  end
```

Understanding the code:

- Explain this script
- How to request a user input?
- What is `'s'` on line 3?
- What is a digit?

Simple application scripts

Example.

```
1  exist('./file') & load('./file')
2  exist('./file') && load('./file')
3  k=input('Press a key: ','s');
4  if k>='0' && k<='9'
5    disp('Digit')
6  else
7    disp('Not a digit')
8  end
```

Understanding the code:

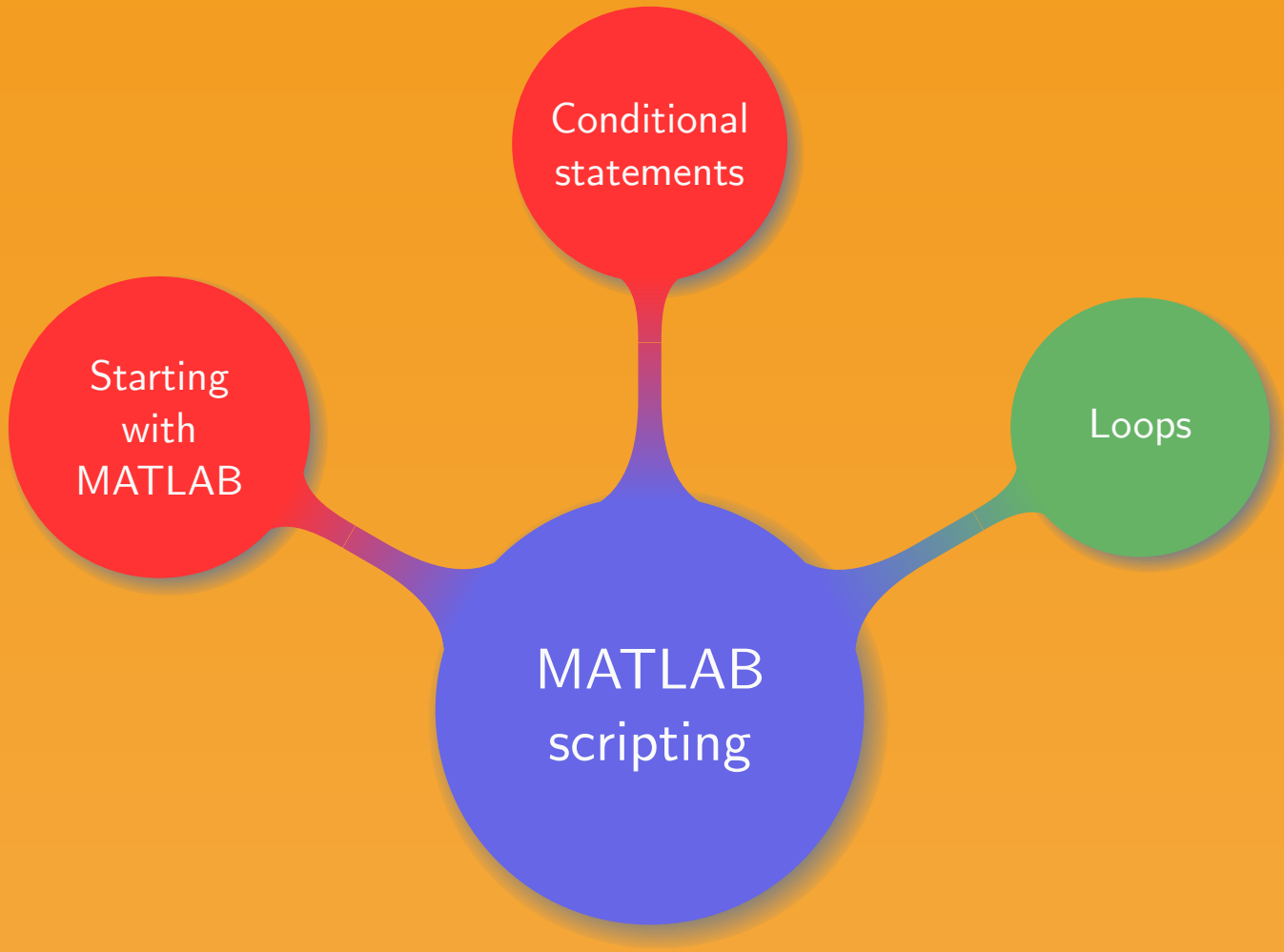
- Explain this script
- How to request a user input?
- What is `'s'` on line 3?
- What is a digit?

```
1  i=input('Input a digit: ');
2  switch i
3    case 0
4      disp('0')
5    case {1,2,3,4}
6      disp('<5')
7    otherwise
8      disp('>=5')
9  end
```

Understanding the code:

- Explain this script
- How is the code aligned?
- Why is `input` used without the parameter `'s'`?

Chapter organisation



Basics on loops

Loops in MATLAB:

- Definition: group of statements repeatedly executed as long as a given conditional expression remains true
- Types: `while`, `for`, and vectorizing
- Vectorizing: generate a vector containing all elements
- `For` loop: clear steps and predefined end
- `While` loop: end based on a boolean expression
- Order of preference: vectorizing, `for`, and `while`

The `while` loop

```
while expression
  statements
end
```

```
i=0
while true
  i=i+1
end
```

The `while` loop

```
while expression
  statements
end
```

```
i=0
while true
  i=i+1
end
```

Example.

```
i=1; o=input('Input a basic arithmetic operation: ','s');
while (o(i) >= '0' && o(i) <= '9') i = i+1; end
n1=str2num(o(1:i-1)); n=o(i); n2=str2num(o(i+1:end));
switch n
  case '+', n1+n2
  case '-',  n1-n2
  case '*',  n1*n2
  case '/',  n1/n2
  otherwise, disp('Not a basic arithmetic operation')
end
```

The `while` loop

Understanding the code:

- How well is the code formatted?
- Reformat the code with more spacing
- What is the user expected to input?
- What is the purpose of the `while` loop?
- How is switch used?
- What is happening if something else that an integer is input?

The `for` loop

```
for i=start:increment:end
  statements
end
```

```
a=[]
for i=0:2:100
  a=[a i]
end
```

Understanding the code:

- How is the code indented?
- What is the role of the increment?
- What is this code doing?
- Can you think of a faster way to obtain the same result?

Vectorizing loop

Use MATLAB optimizations for vectors and array to construct lists

Example.

```
1 a=zeros(1,100000000); i=1;
2 tic; while i<=100000000; a(i)=2*(i-1); i=i+1; end; toc;
3 a=zeros(1,100000000);
4 tic; for i=1:100000000; a(i)=2*(i-1); end; toc;
5 tic; [0:2:199999999]; toc;
```

Understanding the code:

- Reformat and indent the code with one instruction per line
- What is this code doing?

The `continue` and `break` commands

More advanced loop commands:

- Directly jump to the next iteration: `continue`
- Exit the loop early: `break`

The `continue` and `break` commands

More advanced loop commands:

- Directly jump to the next iteration: `continue`
- Exit the loop early: `break`

Example.

```
d={'1','2','3','4','5','6','7','8','9','0'}; cnt=0;
w=input('Input a word: ','s');
for i=1:length(w);
  switch w(i);
    case d;
      continue;
    case ' ';
      break;
    otherwise
      cnt=cnt+1;
  end,
end
cnt
```

The `continue` and `break` commands

Understanding the code:

- What is this code doing?
- How is the code indented?
- What is the variable `d`?
- How are `continue` and `break` used?

Efficiency

Arrays are stored *linearly* inside memory:

- Row first: elements are read by row
- Column first: elements are read by column
- MATLAB uses the *column-major order*
- When using MATLAB the column should be in the outer loop

Efficiency

Arrays are stored *linearly* inside memory:

- Row first: elements are read by row
- Column first: elements are read by column
- MATLAB uses the *column-major order*
- When using MATLAB the column should be in the outer loop

Exercise.

Does MATLAB store $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ as 1,2,3,4,5,6 or 1,4,2,5,3,6?

Efficiency

Example.

```
N = 10000; a = zeros(N);
tic;
  for j = 1:N
    for i=1:N
      a(j,i) = 1;
    end
  end
toc;
```

Efficiency

Example.

```
1 N = 10000; a = zeros(N);
2 tic;
3   for j = 1:N
4     for i=1:N
5       a(j,i) = 1;
6     end
7   end
8 toc;
```

Understanding the code:

- What is this code doing?
- Is `j` representing the rows or the columns, what about `i`?
- What is happening if `i` and `j` are switched on line 5?

Accessing specific elements in a matrix

Access elements depending on a *logical mask:*

1. Generate an logical array depending on some condition
2. Apply a transformation only on a 1 in the logical array

Accessing specific elements in a matrix

Access elements depending on a *logical mask:*

1. Generate an logical array depending on some condition
2. Apply a transformation only on a 1 in the logical array

Example.

- For a matrix A set all its elements larger than 10 to 0
- Given a vector square all its even values and cube the others

```
A=magic(5); B=A >10;A(B)=0
a=input('Vector: ')
b=(mod(a,2)==0);
c=a.^2;
c(~b)=a(~b).^3
```

Accessing specific elements in a matrix

Understanding the code:

- What is the result of `whos B`?
- What does `B=A > 10` mean?
- What is the goal of line 3?
- After line 4 what is in `c`?
- Why is `~b` used?

Key points

- How to write simple scripts in MATLAB?
- What is the difference between an array and a matrix?
- What is a conditional statements?
- What loop types exist in MATLAB, which one is best used?
- What is a logical mask?

Thank you!