Logicals and Control Flow I

PNI Summer Internship 2020

Mai Nguyen

Logicals

- Special data type with two values: True (1) or False (0)
- Used for selection and control flow
- Typically called “booleans” in other languages

Logicals

- keywords: **true**, **false**

```
>> true

ans =

  logical

   1
```

```
>> false

ans =

  logical

   0
```

Evaluate statements: true or false?

- **Equality**: val1 == val2

Assign value

```
>> val = 3;
```

Test if val is 3

```
>> val == 3

ans =

  logical

   1
```

Test if val is 0

```
>> val == 0

ans =

  logical

   0
```

Evaluate statements: true or false?

- **Greater than, less than:** <, >, <=, =>

Assign value

```
>> val = 3;
```

Test if val is greater than 3

```
>> val > 3

ans =

  logical

   0
```

Test if val greater than or equal to 3

```
>> val >= 3

ans =

  logical

   1
```

Evaluate statements: true or false?

- **Not operator: ~**

Assign value

```
>> val = 3;
```

Test if val is 3

```
>> val == 3

ans =

  logical

   1
```

Test if val is NOT 3

```
>> val ~= 3

ans =

  logical

   0
```

Evaluate statements: true or false?

- Evaluate multiple statements: AND &&
- Both statements must be true to return true

Evaluates true

```
>> true && true

ans =

  logical

   1
```

Evaluates false

```
>> true && false

ans =

  logical

   0
```

```
>> false && false

ans =

  logical

   0
```

```
>> ~true && true

ans =

  logical

   0
```

Evaluate statements: true or false?

- Evaluate multiple statements: OR ||
- Either or both statements must be true to evaluate true

Evaluates true

```
>> true || true

ans =

  logical

   1
```

```
>> true || false

ans =

  logical

   1
```

Evaluates false

```
>> false || false

ans =

  logical

   0
```

```
>> false || ~true

ans =

  logical

   0
```

Evaluate statements: true or false?

- Compare strings: strcmp()

Create string

```
>> str = 'foo';
```

== compares every char

```
>> str == 'foo'

ans =

  1×3 logical array

   1   1   1
```

strcmp() compares entire string

```
>> strcmp(str, 'foo')

ans =

  logical

   1
```

Evaluate statements: true or false?

- Many built-in functions that return logicals
    - isnan
    - exist
    - isempty
    - ischar, isintenger, ismatrix, is(insertDataType)
    - isinf, isfinite

Practice

Logical Operators

- ==, <, >, <=, =>
- Not: ~
- And: &&
- Or: ||

- True or false? Write down your answer and then test in MATLAB

1. 2+2 == 4 TRUE
2. isempty([ ]) TRUE
3. isempty(5) FALSE
4. isempty([ ]) && (2+2 == 4) TRUE && TRUE -> TRUE
5. ~(isempty([ ])) && (2+2 == 4) ~TRUE && TRUE -> FALSE
6. ~isempty([ ]) || (2+2 == 4) ~TRUE || TRUE —> TRUE

Using logicals

- **Logical selection**
- Conditional control flow (if/else statements)

Logical selection

- Find values in matrices that meet a certain condition
- Example 1: find values > 5

```
>> mat = randi(10, 2, 4)

mat =

     9     2     7     3
    10    10     1     6
```

```
>> mat > 5

ans =

  2×4 logical array

   1   0   1   0
   1   1   0   1
```

Logical selection

- Find values in matrices that meet a certain condition
- Example 2: find values > 5 AND < 10

```
>> mat = randi(10, 2, 4)

mat =

     9     2     7     3
    10    10     1     6
```

```
>> (mat > 5) & (mat < 10)

ans =

  2×4 logical array

   1   0   1   0
   0   0   0   1
```

Logical selection

- Find values in matrices that meet a certain condition
- Example 3: find values that are NaNs

```
>> mat = [.23 .19 .64 NaN; NaN .7 .55 .04]

mat =

    0.2300    0.1900    0.6400       NaN
       NaN    0.7000    0.5500    0.0400
```

```
>> isnan(mat)

ans =

  2×4 logical array

   0   0   0   1
   1   0   0   0
```

Logical selection

- Use logicals to get values meeting certain conditions
- Example 1: find values > 6

```
>> mat = randi(10, 2, 4)

mat =

     7     8     7     8
     8     4     2     1
```

```
>> inds = mat > 6

inds =

  2×4 logical array

   1   1   1   1
   1   0   0   0
```

```
>> mat(inds)

ans =

     7
     8
     8
     7
     8
```

Logical selection

- Use logicals to get values meeting certain conditions
- Example 2: find values > 6 OR < 3

```
>> mat = randi(10, 2, 4)

mat =

     7     8     7     8
     8     4     2     1
```

```
>> mat(mat > 6 | mat < 2)
ans =

     7
     8
     8
     7
     8
     1
```

Logical selection

- Use logicals and **find** function to get index of values meeting certain conds
- Example 1: find indices of non-NaN values

```
>> mat = [NaN 2 3 0; NaN NaN NaN 5]

mat =

   NaN     2     3     0
   NaN   NaN   NaN     5
```

```
>> find(~isnan(mat))

ans =

     3
     5
     7
     8
```

Logical selection

- Use logicals and **find** function to get index of values meeting certain conds
- Example 2: find indices of zero values

```
>> mat = randi(10, 4, 4) - 5

mat =

     0    -3     5    -1
    -4    -2     5     5
    -2     0     0    -1
    -3    -4     0    -3
```

```
>> find(mat == 0)

ans =

     1
     7
    11
    12
```

Logical selection

- Use logicals to replace values meeting certain conds
- Example 1: replace values < 0.5 with NaN

```
>> mat = rand(4,4)

mat =

    0.7803    0.0965    0.5752    0.8212
    0.3897    0.1320    0.0598    0.0154
    0.2417    0.9421    0.2348    0.0430
    0.4039    0.9561    0.3532    0.1690
```

```
>> mat(mat < .4) = nan

mat =

    0.7803       NaN    0.5752    0.8212
       NaN       NaN       NaN       NaN
       NaN    0.9421       NaN       NaN
    0.4039    0.9561       NaN       NaN
```

Logical selection

- Use logicals to replace values meeting certain conds
- Example 2: replace values NaNs with different numbers

```
>> mat = [0 0 nan; nan nan 0; 0 0 nan]

mat =

     0     0   NaN
   NaN   NaN     0
     0     0   NaN
```

```
>> mat(isnan(mat)) = 1:4

mat =

     0     0     3
     1     2     0
     0     0     4
```

Logical selection summary

- Find values in an array that meet a certain condition(s) and return a matrix of logicals
- Return values in an array that meet a certain condition(s)
- Return indices of values in an array that meet a certain condition(s)
- Replaces values in an array that meet certain condition(s) with other values

Practice

1. Using the randi function, create a 3x4 matrix of values from 1-10

2. Return a matrix of logicals indicating values that are greater than or equal to 8

3. Find the values that are greater than or equal to 8

4. Find the indices of values that are greater than or equal to 8

5. Find the values that are greater than or equal to 8 OR less than 3

6. Replace all the values less than 5 with nans

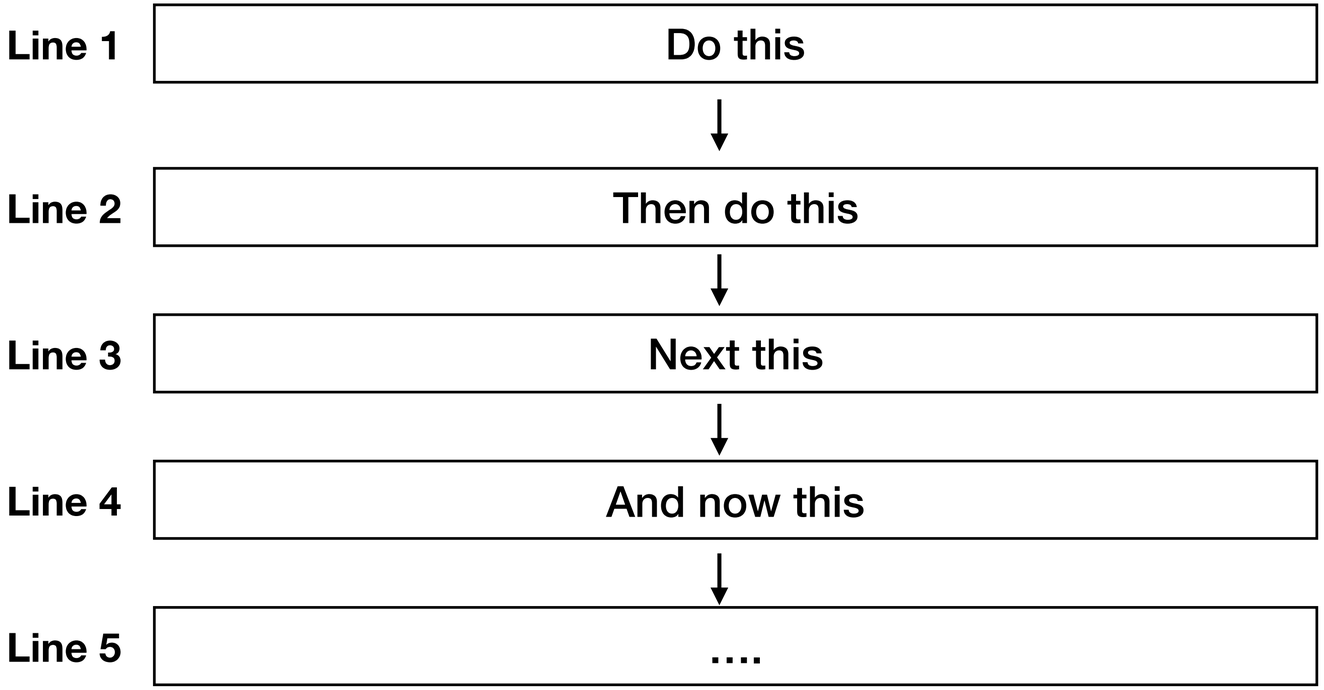
7. Find the indices of the nans

Using logicals

- Logical selection
- **Conditional control flow (if/else statements)**

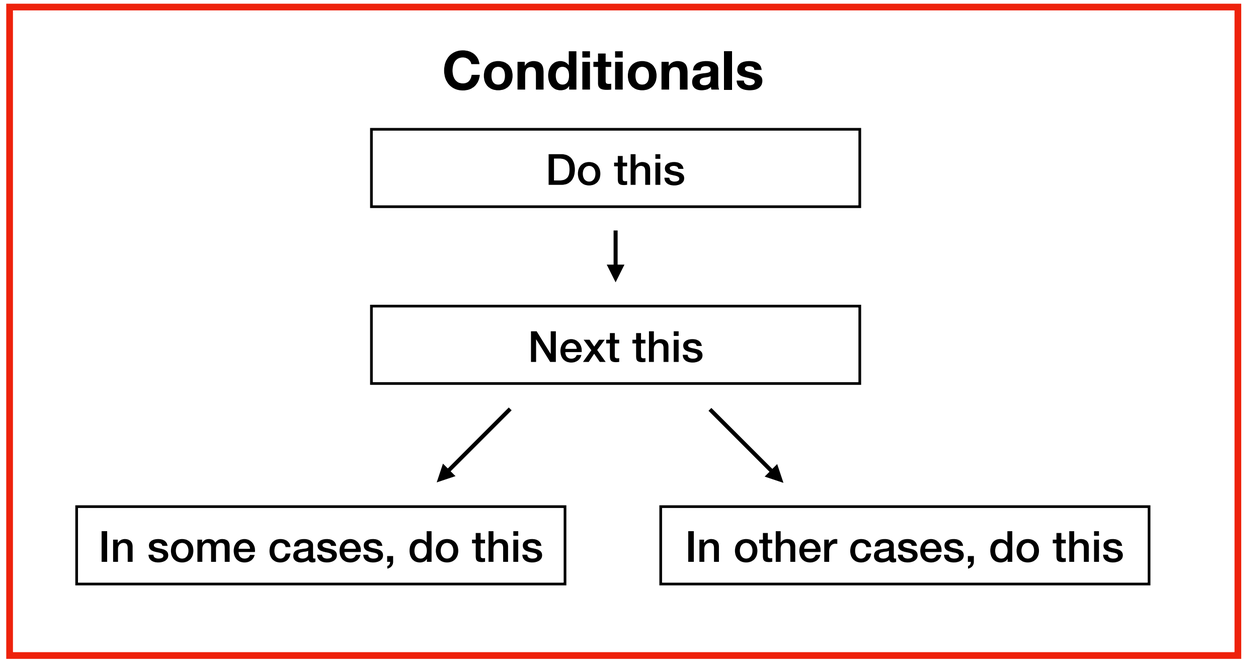
Control flow

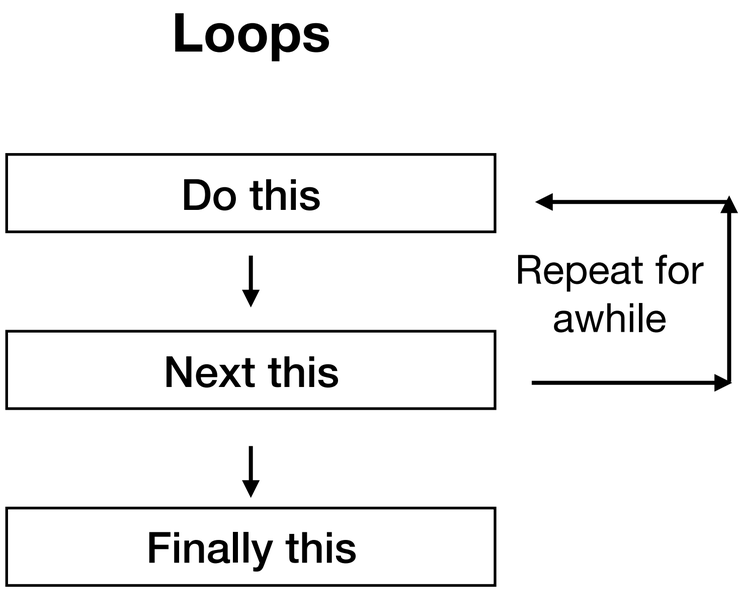
- So far: MATLAB runs every line in a script and/or function



Control flow

- Control flow allows us to control which code is run (conditionals) and how many times (loops)





Conditionals: if/else statements

```
1  if some_cond_is_true
2
3      % do something
4
5  end
```

- Start with keyword **if**
- **Conditional statement** that evals to True or False
- Close with keyword **end**
- **Code body**

Conditionals: if/else statements

is_odd.m | is_3.m

```
1
2   function is_3(val)
3
4   if val == 3
5       fprintf('This number is 3!\n')
6   end
```

COMMAND WINDOW

```
>> is_3(10)
>> is_3(2.99999)
>> is_3(3)
This number is 3!
```

Doesn’t do anything if statement is false

Conditionals: if/else statements

```
if some_cond_is_true
    % do this

else
    % otherwise do this

end
```

- Start with keyword **if**
- Close with keyword **end**
- **Code body**
- Optional: keyword **else**

Conditionals: if/else statements

is_odd.m | is_3.m | untitled2 *

```
1
2   function is_3(val)
3
4   if val == 3
5       fprintf('This number is 3!\n')
6   else
7       fprintf('This is not 3.')
8   end
```

COMMAND WINDOW

```
>> is_3(99)
This is not 3.
>> is_3(0)
This is not 3.
>> is_3(3)
This number is 3!
```

Maybe we should print something else if val is specifically -3?

Conditionals: if/else statements

```
if some_cond_is_true
    % do this

elseif some_other_cond_is_true
    % do this

else
    % otherwise do this

end
```

- Start with keyword **if**
- Close with keyword **end**
- **Code body**
- Optional: keyword **else**
- Optional: keyword **elseif**

Note that elseif statements are evaluated in order!!

Conditionals: if/else statements

is_odd.m | is_3.m | untitled2 *

```
function is_3(val)

if val == 3
    fprintf('This number is 3!\n')

elseif val == -3
    fprintf('This number is -3!\n')

else
    fprintf('This is not 3.\n')
end
```

COMMAND WINDOW

```
>> is_3(0)
This is not 3.
>> is_3(-3)
This number is -3!
>> is_3(3)
This number is 3!
```

What if some idiot passes something other than a number as an argument?

Conditionals: if/else statements

```
if this_cond_is_true
    % do this

else
    % otherwise do one of these things

    if some_cond_is_true
        % do this
    elseif some_other_cond_is_true
        % do this
    else
        % otherwise do this
    end
end
```

- Nested if/else statements
- **Outer** if/else
- **Inner** if/else

Indentation is important for readability! Use ctrl/cmd + i for auto-indent

Conditionals: if/else statements

is_3.m | untitled *

```
function is_3(val)

if ischar(val)
    fprintf('That''s not even a number!\n')
else
    if val == 3
        fprintf('This number is 3!\n')

    elseif val == -3
        fprintf('This number is -3!\n')

    else
        fprintf('This is not 3.\n')
    end
end

end
```

COMMAND WINDOW

```
>> is_3(3)
This number is 3!
>> is_3(-3)
This number is -3!
>> is_3(3.33)
This is not 3.
```

Practice

- Write a function pass_test(score) that checks if the input value is at least a passing score of 0.65. The function has one argument, score
    - If the score is between 0.65 and 1, the student passes! Print a message saying so.
    - If the score is between 0 and 0.64, the student fails. Print a message saying so.
    - If the score is not between 0 and 1 inclusive, it is invalid. Print a message saying so
- Test your function
    - pass_test(.96)
    - pass_test(.65)
    - pass_test(.5)
    - pass_test(-.1)
    - pass_test(1.5)

```
function function_name(arg)
    if some_cond
        %do stuff
    elseif some_other_cond
        %do stuff
    else
        %do stuff
end
```

Write your function in a separate file!

Conditionals: switch statements

```
if condition1
    % do this
elseif condition2
    % do this
elseif condition3
    % do this
elseif condition4
    % do this
elseif condition5
    % do this
elseif condition6
    % do this
elseif condition7
    % do this
else
    % do this
end
```

in general don't do this

- Use switch statements instead of if/else if you have a lot of cases
- Cannot use relational operators < and >

Conditionals: switch statements

```
switch value
    case option1
        % do this if value == option1
    case option2
        % do this if value == option2
    case option3
        % do this if value == option3
    otherwise
        % otherwise do this
end
```

- Start with keyword **switch**
- **switch_expression** or value used in conditional
- keyword **case** specifies an option
- defines **conditional**, compare value to option1

Conditionals: switch statements

start_exp.m | is_odd.m | is_3.m | untitled2 * | define_gene

```
function start_exp(cond)

    switch cond
        case 1
            fprintf('Run cond 1')
        case 2
            fprintf('Run cond 2')
        case 3
            fprintf('Run cond 3')
        otherwise
            fprintf('Invalid condition')
    end

end
```

COMMAND WINDOW

```
>> start_exp(1)
Run cond 1
>> start_exp(2)
Run cond 2
>> start_exp(3)
Run cond 3
>> start_exp(0)
Invalid condition
```

Practice

- Write a function that gives feedback on grades, grade_feedback(letterGrade).
    - If the grade is an A, print “outstanding”
    - If the grade is a B, print “very good”
    - If the grade is a C, print “average”
    - If the grade is a D, print “passed”
    - If the grade is an F, print “failed”
    - If the input value is not one of the above, print “invalid input”
- Test your function
    - grade_feedback(‘A’) %repeat for all letter grades
    - grade_feedback(100)

Review

Logicals
True/false
==, ~=
>, <, =>, <=
strcmp()
isnan()
exist()
isempty()
ischar(), ismatrix()
isinf() isfinite()

Logical selection
Return logical
Return values
find() index
Replacing values

If/else statements
if
elseif
else
nested

Switch statements
switch
case
otherwise