L09: Iteration

For and while loops

Lucas A. J. Bastien

E7 Spring 2017, University of California at Berkeley

February 06, 2017

Version: release

Announcements

Lab 03 is due on February 10 at 12 pm (noon)

Readings for this week

- Chapters 5 and 6

Today:

- Iteration (`for` and `while` loops)

Wednesday:

- Recursion (*i.e.* when functions call themselves)

Friday:

- Discussion and practice questions (array operations versus loops)

Introduction to iteration

1! equals 1

Introduction to iteration

1! equals 1

2! equals 2

Introduction to iteration

1! equals 1

2! equals 2

3! equals 6

Introduction to iteration

1! equals 1

2! equals 2

3! equals 6

4! equals 24

Introduction to iteration

1! equals 1

2! equals 2

3! equals 6

4! equals 24

5! equals 120

Introduction to iteration

1! equals 1

2! equals 2

3! equals 6

4! equals 24

5! equals 120

6! equals 720

Introduction to iteration

1! equals 1

2! equals 2

3! equals 6

4! equals 24

5! equals 120

6! equals 720

7! equals 5040

Introduction to iteration

1! equals 1

2! equals 2

3! equals 6

4! equals 24

5! equals 120

6! equals 720

7! equals 5040

8! equals 40320

Introduction to iteration

1! equals 1

2! equals 2

3! equals 6

4! equals 24

5! equals 120

6! equals 720

7! equals 5040

8! equals 40320

9! equals 362880

Introduction to iteration

1! equals 1

2! equals 2

3! equals 6

4! equals 24

5! equals 120

6! equals 720

7! equals 5040

8! equals 40320

9! equals 362880

10! equals 3628800

Introduction to iteration

1! equals 1

2! equals 2

3! equals 6

4! equals 24

5! equals 120

6! equals 720

7! equals 5040

8! equals 40320

9! equals 362880

10! equals 3628800

To notice:

- Parts of the line are repeated from one line to the next
- Parts of the line change from one line to the next

Introduction to iteration

1! equals 1

2! equals 2

3! equals 6

4! equals 24

5! equals 120

6! equals 720

7! equals 5040

8! equals 40320

9! equals 362880

10! equals 3628800

To notice:

- Parts of the line are repeated from one line to the next
- Parts of the line change from one line to the next

Iteration is used to repeat the execution of a section of code

- The code can produce different results at each step

Syntax of for loops

```
>> for iteration_variable = row_vector
>> % The code to be repeated goes here.
>> % The code will be repeated once for
>> % each value in row_vector. At each step,
>> % iteration_variable will have the corresponding
>> % value from the row vector
>> end
```

Syntax of for loops

```
>> for iteration_variable = row_vector
>> % The code to be repeated goes here.
>> % The code will be repeated once for
>> % each value in row_vector. At each step,
>> % iteration_variable will have the corresponding
>> % value from the row vector
>> end
```

Notes:

- The `for` and `end` keywords and the equal sign are mandatory

Syntax of for loops

```
>> for iteration_variable = row_vector
>> % The code to be repeated goes here.
>> % The code will be repeated once for
>> % each value in row_vector. At each step,
>> % iteration_variable will have the corresponding
>> % value from the row vector
>> end
```

Notes:

- The `for` and `end` keywords and the equal sign are mandatory
- row_vector can be of class `char`, `double`, or `logical`

Syntax of for loops

```
>> for iteration_variable = row_vector
>> % The code to be repeated goes here.
>> % The code will be repeated once for
>> % each value in row_vector. At each step,
>> % iteration_variable will have the corresponding
>> % value from the row vector
>> end
```

Notes:

- The `for` and `end` keywords and the equal sign are mandatory
- row_vector can be of class `char`, `double`, or `logical`
- You can use something other than a row vector of class `char`, `double`, or `logical`, but I recommend against it (more difficult to use)

Syntax of for loops

```
>> for iteration_variable = row_vector
>> % The code to be repeated goes here.
>> % The code will be repeated once for
>> % each value in row_vector. At each step,
>> % iteration_variable will have the corresponding
>> % value from the row vector
>> end
```

Notes:

- The `for` and `end` keywords and the equal sign are mandatory
- row_vector can be of class `char`, `double`, or `logical`
- You can use something other than a row vector of class `char`, `double`, or `logical`, but I recommend against it (more difficult to use)
- In E7 applications, the row vector will most often be of class `double`

Examples of for loops

```
>> % Show the values of factorial(n) from n=1 to n=10
>> for i = 1:10
>>     fprintf('%d! equals %d\n', i, factorial(i));
>> end



>> % Show the values of factorial(n) from n=1 to n=10,
>> % calculating the factorial "manually" at each step
>> value = 1;
>> for i = 1:10
>>     value = value * i;
>>     fprintf('%d! equals %d\n', i, value);
>> end
```

More complex example: `my_sum.m`

For loops: practice question

What will the value of the variable "x" be after executing the following code?

```
>> x = 1;
>> for v = [2, 4, 6, 10]
>>     if v > 5
>>         x = x + v;
>>     end
>>     x = x + 1;
>> end
```

(A) 1

(B) 5

(C) 21

(D) 2

(E) None of the above

For loops: practice question

What will the value of the variable “x” be after executing the following code?

```
>> x = 1;
>> for v = [2, 4, 6, 10]
>>     if v > 5
>>         x = x + v;
>>     end
>>     x = x + 1;
>> end
```

(A) 1

(B) 5

(C) [✓ 21]

(D) 2

(E) None of the above

For loops: practice question

Assuming we start with an empty Workspace, what will the value of the variable “x” be after executing the following code?

```
>> x = 1;
>> for i = 10:-2:4
>>     x = x^2;
>>     for j = 10:1
>>         x = x * (y+1);
>>     end
>> end
```

(A) 1

(B) `Inf`

(C) `NaN`

(D) Error: Undefined function or variable ’y’

For loops: practice question

Assuming we start with an empty Workspace, what will the value of the variable "x" be after executing the following code?

```
>> x = 1;
>> for i = 10:-2:4
>>     x = x^2;
>>     for j = 10:1
>>         x = x * (y+1);
>>     end
>> end
```

(A) [✓ 1]

(B) `Inf`

(C) `NaN`

(D) Error: Undefined function or variable 'y'

Note: the inner `for` loop has zero iteration

Syntax of while loops

```
>> while (logical expression)
>> % The code to be repeated goes here.
>> % At each iteration, the logical
>> % expression is evaluated. If it evaluates
>> % to true, the code here is executed. If it
>> % evaluates to false, the code here is not
>> % executed and the loop exits.
>> end
```

Syntax of while loops

```
>> while (logical expression)
>> % The code to be repeated goes here.
>> % At each iteration, the logical
>> % expression is evaluated. If it evaluates
>> % to true, the code here is executed. If it
>> % evaluates to false, the code here is not
>> % executed and the loop exits.
>> end
```

Notes:

- The `while` and `end` keywords are mandatory

Syntax of while loops

```
>> while (logical expression)
>> % The code to be repeated goes here.
>> % At each iteration, the logical
>> % expression is evaluated. If it evaluates
>> % to true, the code here is executed. If it
>> % evaluates to false, the code here is not
>> % executed and the loop exits.
>> end
```

Notes:

- The `while` and `end` keywords are mandatory
- If the logical expression never evaluates to false, then the while loop never exits ($\rightarrow$ "infinite loop")

Examples of while loops

```
function index = my_index_while(vector, value)

% Returns the index of the first occurrence of non-NaN "value"
% in "vector" (returns -1 if value is not in vector)

n = numel(vector);
index = 0;
keep_looking = true;

while keep_looking
    index = index + 1;
    keep_looking = (vector(index) ~= value & index < n);
end

if vector(index) ~= value
    index = -1;
end

end
```

Other example: `my_integer_guess.m`

While loops: practice question

What will the value of the variable "y" be after executing the following code?

```
>> vector = 1;
>> while sum(vector) < 14
>>     vector(end+1) = vector(end) + 1;
>> end
>> y = numel(vector);
```

(A) 4

(B) 5

(C) 13

(D) 14

(E) This is an infinite loop!

While loops: practice question

What will the value of the variable "y" be after executing the following code?

```
>> vector = 1;
>> while sum(vector) < 14
>>     vector(end+1) = vector(end) + 1;
>> end
>> y = numel(vector);
```

(A) 4

[✓ **(B)** 5]

(C) 13

(D) 14

(E) This is an infinite loop!

While loops: practice question

What will the value of the variable “y” be after executing the following code?

```
>> vector = 1;
>> while sum(vector) ~= 14
>>     vector(end+1) = vector(end) + 1;
>> end
>> y = numel(vector);
```

(A) 4

(B) 5

(C) 13

(D) 14

(E) This is an infinite loop!

While loops: practice question

What will the value of the variable “y” be after executing the following code?

```
>> vector = 1;
>> while sum(vector) ~= 14
>>     vector(end+1) = vector(end) + 1;
>> end
>> y = numel(vector);
```

(A) 4

(B) 5

(C) 13

(D) 14

(E) [✓ This is an infinite loop!]

The break command

Use the command `break` to immediately exit a `for` or `while` loop (only the inner-most loop containing the `break` command is exited)

```
function index = my_index_break(vector, value)

% Returns the index of the first occurrence of non-NaN "value"
% in "vector" (returns -1 if value is not in vector)

for index = 1:numel(vector)
    if vector(index) == value
        break
    end
end

if vector(index) ˜= value
    index = -1;
end

end
```

The break command: practice question

What will the value of the variable “var” be after executing the following code?

```
>> for var = 1:2:100
>>     if sum(1:var) > 50
>>         break
>>     end
>> end
```

(A) 9

(B) 11

(C) 49

(D) 98

(E) 100

The break command: practice question

What will the value of the variable "var" be after executing the following code?

```
>> for var = 1:2:100
>>     if sum(1:var) > 50
>>         break
>>     end
>> end
```

(A) 9

(B) [✓ 11]

(C) 49

(D) 98

(E) 100

For loop versus while loop

When to use a `for` loop versus a `while` loop?

For loop versus while loop

When to use a `for` loop versus a `while` loop?

If you know the number of steps you will need:

- Use a `for` loop

For loop versus while loop

When to use a `for` loop versus a `while` loop?

If you know the number of steps you will need:

- Use a `for` loop

If you know the maximum number of steps you may need, but you may need fewer steps than that:

- Use a `for` loop with a `break`, or a `while` loop

For loop versus while loop

When to use a `for` loop versus a `while` loop?

If you know the number of steps you will need:

- Use a `for` loop

If you know the maximum number of steps you may need, but you may need fewer steps than that:

- Use a `for` loop with a `break`, or a `while` loop

If you do not know the number of steps that you will need:

- Use a `while` loop