Module 04: Loop Statement

Chul Min Yeum
Assistant Professor
Civil and Environmental Engineering
University of Waterloo, Canada

UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING

Module 4: Learning Outcomes

- Describe a problem that requires a loop statement
- Explain how a for-loop structure is operated
- Identify a break command and its operation
- Illustrate the difference between for-loop and while statement
- Solve and design problems using a loop statement and if-statement

For-Loop Statement

- Used as a *counted* loop (We know how many times are repeated)
- *Repeats an action* a specified number of times
- An *iterator* or loop variable specifies how many times to repeat the action
- General form:

```
for loopvar = range
    action
end
```

- The `range` is specified by a **vector**.
- The `action` is repeated for every value of `loopvar` in the specified vector
- If it is desired to repeat the process of prompting the user and reading input a specified number of times (N), a for loop is used:

```
for ii = 1:N
    % do something with it!
end
```

Example: Summation of Values

Q. Sum 1 to 3 and assign the value to 'sumv'

```
1  sumv = 0;
2  for ii=1:3
3      sumv = sumv + ii;
4  end
5
```

Step	Operation	Workspace
1	line1: assign 0 to sumv	sumv → 0
2	line2: ii becomes 1	sumv → 0, ii → 1
3	line3: add ii to sumv and assign the value to sumv	sumv → 1, ii → 1
4	line4: end	sumv → 1, ii → 1
5	go back to line2 and ii becomes 2	sumv → 1, ii → 2
6	line3: add ii to sumv and assign the value to sumv	sumv → 3, ii → 2
7	line4: end	sumv → 3, ii → 2
8	go back to line2 and ii becomes 3	sumv → 3, ii → 3
9	line3: add ii to sumv and assign the value to sumv	sumv → 6, ii → 3
10	line4: end	sumv → 6, ii → 3
11	no more value in range and go to line5	sumv → 6, ii → 3

📖: sumv → 0 indicates that sumv contain a value 0.

Example: Summations of Values in a Vector

Q. Sum all values in a vector named 'vec' and assign the value to 'sumv'

```
1  vec = [2 3 7 11];
2  sumv = 0;
3  for ii=1:4
4      sumv = sumv + vec(ii);
5  end
```

Step	Operation	Workspace
1	line 1 and 2: assign `[1 3 7 11]` to `vec` and `0` to `sumv`	`vec → [1 3 7 11], sumv → 0`
2	line 3: `ii` becomes 1	`sumv → 0, ii → 1`
3	line 4: add `vec(ii)` to `sumv` and assign the value to `sumv`	`sumv → 2, ii → 1`
4	line 5: `end`	`sumv → 2, ii → 1`
5	line 3: `ii` becomes 2	`sumv → 2, ii → 2`
6	line 4: add `vec(ii)` to `sumv` and assign the value to `sumv`	`sumv → 5, ii → 2`
7	line 5: `end`	`sumv → 5, ii → 2`
:	continue to run until `ii` becomes 4.	
9	line 5: `end`	`sumv → 23, ii → 4`

Example: Summations of Values in a Vector (Continue)

Two different ways of designing a loop statement.

```
1  vec = [2 3 7 11];
2  sumv = 0;
3  nvec = numel(vec);
4  for ii=1:nvec
5      sumv = sumv + vec(ii);
   end
```

```
1  vec = [2 3 7 11];
2  sumv = 0;
3  for val=vec
4      sumv = sumv + val;
5  end
```

```
for ii = 1:N
    % do something!
end
```

```
for loopvar = range
    % do something!
end
```

Nested For-Loop Statement

- A nested **for** loop is one inside of (as the action of) another **for** loop
- General form of a nested **for** loop:

```
for loopvar1 = range1
    action1
    for loopvar2 = range2
        action2
    end
end
```

- **The inner loop action is executed in its entirety for every value of the outer loop variable**

Example: Summation of Values in a Matrix

Q. Sum all values in a matrix named '`mat1`' and assign the value to '`sumv`'

`mat1`

2	8	7
1	5	6

```
1  mat1 = [2 8 7; 1 5 6];
2  sumv = 0;
3  for ii=1:2
4      for jj=1:3
5          sumv = sumv + mat1(ii,jj);
6      end
7  end
```

Step	Operation	Workspace
1	line1: assign values to `mat1`	`mat1 → [2 8 7;1 5 6]`
2	line2: assign `0` to `sumv`	`mat1 → [2 8 7;1 5 6], sumv → 0`
3	line3: `ii` becomes 1	`sumv → 0, ii → 1`
4	line4: `jj` becomes 1	`sumv → 0, ii → 1, jj → 1`
5	line5: Read a value at the 1st row and 1st column in `mat1` and add the value to `sumv` and assign the value to `sumv`	`sumv → 2, ii → 1, jj → 1`
6	line6: `end`	`sumv → 2, ii → 1, jj → 1`
7	line4: `jj` becomes 2	`sumv → 2, ii → 1, jj → 2`

Omit **`mat1`** after Step 2 in Workspace

Example: Summation of Values in a Matrix (Continue)

Q. Sum all values in a matrix named '`mat1`' and assign the value to '`sumv`'

`mat1`

2	8	7
1	5	6

```
1  mat1 = [2 8 7; 1 5 6];
2  sumv = 0;
3  for ii=1:2
4      for jj=1:3
5          sumv = sumv + mat1(ii,jj);
6      end
7  end
```

Step	Operation	Workspace
7	line4: `jj` becomes 2	`sumv → 2, ii → 1, jj → 2`
8	line5: Read a value at the 1st row and 2nd column in `mat1` and add the value to `sumv` and assign the value to `sumv`	`sumv → 10, ii → 1, jj → 2`
9	line6: `end`	`sumv → 10, ii → 1, jj → 2`
10	line4: `jj` becomes 3	`sumv → 10, ii → 1, jj → 3`
11	line5: Read a value at the 1st row and 3rd column in `mat1` and add the value to `sumv` and assign the value to `sumv`	`sumv → 17, ii → 1, jj → 3`
12	line6: `end`	`sumv → 17, ii → 1, jj → 3`

Example: Summation of Values in a Matrix (Continue)

Q. Sum all values in a matrix named '`mat1`' and assign the value to '`sumv`'

`mat1`

2	8	7
1	5	6

```
1  mat1 = [2 8 7; 1 5 6];
2  sumv = 0;
3  for ii=1:2
4      for jj=1:3
5          sumv = sumv + mat1(ii,jj);
6      end
7  end
```

Step	Operation	Workspace
12	line6: `end`	`sumv → 17, ii → 1, jj → 3`
13	line7: `end`	`sumv → 17, ii → 1, jj → 3`
14	line3: `ii` becomes 2	`sumv → 17, ii → 2, jj → 3`
15	line4: `jj` becomes 1	`sumv → 17, ii → 2, jj → 1`
16	line5: Read a value at the 2nd row and 1st column in `mat1` and add the value to `sumv` and assign the value to `sumv`	`sumv → 18, ii → 2, jj → 1`
17	line6: `end`	`sumv → 18, ii → 2, jj → 1`

Example: Summation of Values in a Matrix (Continue)

Q. Sum all values in a matrix named '`mat1`' and assign the value to '`sumv`'

mat1

2	8	7
1	5	6

```
1  mat1 = [2 8 7; 1 5 6];
2  sumv = 0;
3  for ii=1:2
4      for jj=1:3
5          sumv = sumv + mat1(ii,jj);
6      end
7  end
```

Step	Operation	Workspace
17	line6: `end`	`sumv → 18, ii → 2, jj → 1`
18	line4: `jj` becomes 2	`sumv → 18, ii → 2, jj → 2`
19	line5: Read a value at the 2nd row and 2st column in `mat1` and add the value to `sumv` and assign the value to `sumv`	`sumv → 23, ii → 2, jj → 2`
20	line6: `end`	`sumv → 23, ii → 2, jj → 2`
21	line4: `jj` becomes 3	`sumv → 23, ii → 2, jj → 3`
22	line5: Read a value at the 2nd row and 3rd column in `mat1` and add the value to `sumv` and assign the value to `sumv`	`sumv → 29, ii → 2, jj → 3`
23	line6: `end`, line7: `end`	`sumv → 29, ii → 2, jj → 3`

Summation of Values using Linear Indexing

Challenging

Q. Sum all values in a matrix named '`mat1`' and assign the value to '`sumv`'

```
mat1 = [2 8 7; 1 5 6];
sumv = 0;
for ii=1:2
    for jj=1:3
        sumv = sumv + mat1(ii,jj);
    end
end
```

```
mat1 = [2 8 7; 1 5 6];
sumv = 0;
n_val = numel(mat1);
for ii=1:n_val
    sumv = sumv + mat1(ii);
end
```

`mat1`

2	8	7
1	5	6

Element

(1,1)	(1,2)	(1,3)
(2,1)	(2,2)	(2,3)

Subscripted indexing

1	3	5
2	4	6

Linear indexing

Pre-allocating a Vector

- Preallocating sets aside enough memory for a vector to be stored
- The alternative, extending a vector, is very inefficient because it requires finding new memory and copying values every time
- Many functions can be used to pre-allocate, although it is common to use `zeros`
- For example, to preallocate a vector `vec` to have `N` elements:
- `vec = zeros(1,N);`
- (Highly recommended) if you knew the array size that you allocate values computed from a loop !!!

Example: Summation of Values in Each Row of a Matrix

Q. Sum all values in each row of the matrix named '`mat1`' and assign the value to the corresponding row of a row vector named '`rvec`' .

```
mat1 = [2 8; 1 3; 2 3];
rvec = zeros(1,3);
for ii=1:3
    sumr = 0;
    for jj=1:2
        sumr = sumr + mat1(ii,jj);
    end
    rvec(ii) = sumr; % identical with rvec(ii,1)
end
```

Name	Value
mat1	[2 8; 1 3; 2 3]
rvec	[10 4 5]

```
mat1 = [2 8; 1 3; 2 3];
for ii=1:3
    sumr = 0;
    for jj=1:2
        sumr = sumr + mat1(ii,jj);
    end
    rvec(ii) = sumr;
end
```

⚠: `rvec` is not pre-allocated. Thus, in each iteration, the size of `rvec` will be changed

Working but not recommended

Example: Summation of Values in Each Row of a Matrix (Continue)

Q. Sum all values in each row of the matrix named '`mat1`' and assign the value to the corresponding row of a row vector named '`rvec`' .

```
mat1 = [2 8; 1 3; 2 3];
rvec = zeros(1,3);
for ii=1:3
    sumr = 0;
    for jj=1:2
        sumr = sumr + mat1(ii,jj);
    end
    rvec(ii) = sumr; % identical with rvec(ii,1)
end
```

Name	Value
mat1	[2 8; 1 3; 2 3]
rvec	[10 4 5]

```
mat1 = [2 8; 1 3; 2 3];
rvec = zeros(1,3);

for ii=1:3
    for jj=1:2
        rvec(ii) = rvec(ii) + mat1(ii,jj);
    end
end
```

Combining For-Loop and If

- **for** loops and **if** statements can be combined
  - the action of a loop can include an **if** statement
  - the action of an **if** statement can include a **for** loop
- This is also true for nested **for** loops; **if** statements can be part of the action(s) of the outer and/or inner loops
- This is done if an action is required on an element (of a vector or matrix) only if a condition is met

```
for loopvar1 = range1
    if condition
       action
    end
end
```

📖: Again, `action` is only executed if `condition` is `true`. This form is to execute `action` using selective values in `range1`.

Example: Combining For-Loop and If-elseif Statement

Q. Change 1 to 5, 2 to 7, and the rest to 10 in a given vector named ‘vec’

```
vec = [1 1 2 1 3 1 6 7 5];

n_vec = numel(vec);
for ii=1:n_vec
    if vec(ii) == 1
        vec(ii) = 5;
    elseif vec(ii) == 2
        vec(ii) = 7;
    else
        vec(ii) = 10;
    end
end
```

Name	Value
vec	[5 5 7 5 10 5 10 10 10]
n_vec	9

Example: Value Replacement

Q. If values in 'vec1' are larger than and equal to 0 and less than 50, replace the values to 10. Otherwise, replace them to 5.

```
vec = [1 10 70 80 2];

n_vec = numel(vec);

for ii=1:n_vec
    t_val = vec(ii);
    if (t_val >= 0) && (t_val < 50)
        vec(ii) = 10;
    else
        vec(ii) = 5;
    end
end
```

Name	Value
vec	[1 10 70 80 2]
n_vec	[10 10 5 5 10]

Example: Find a Value

Q. Find index(es) of a vector where 5 is located. The vector named as 'vec' is given and the indexes are assigned to 'loc'.

```
vec = [1 5 6 4 8 5 3 7 5];

n_vec = numel(vec);
loc = [];
for ii=1:n_vec
    if vec(ii) == 5
        loc = [loc ii];
    end
end
```

Option 1

Name	Value
loc	[2 6 9]

```
vec = [1 5 6 4 8 5 3 7 5];

n_vec = numel(vec);
loc = [];
for ii=1:n_vec
    if vec(ii) == 5
        loc(end+1) = ii;
    end
end
```

Option 2

```
vec = [1 5 6 4 8 5 3 7 5];

n_vec = numel(vec);
loc = 0;
count = 1;
for ii=1:n_vec
    if vec(ii) == 5
        loc(count) = ii;
        count = count +  1;
    end
end
```

Option 3

Example: Find a Value (Continue)

Q. Find index(es) of a vector where 5 is located. The vector named as 'vec' is given and the indexes are assigned to 'loc'.

```
vec = [1 5 6 4 8 5 3 7 5];

n_vec = numel(vec);
loc = [];
for ii=1:n_vec
    if vec(ii) == 5
        loc = [loc ii];
    end
end
```

Option 1

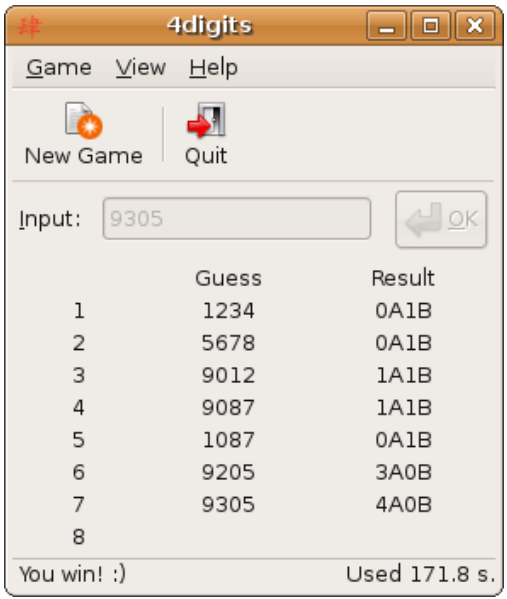
📖: Since you do not know the size of loc in advance, you can pre-allocate the variable with its maximum size and delete the "unused" elements.

```
vec = [1 5 6 4 8 5 3 7 5];

n_vec = numel(vec);
loc = zeros(1, n_vec);

count = 0;
for ii=1:n_vec
    if vec(ii) == 5
        count = count + 1;
        loc(count) = ii;
    end
end

loc = loc(1:count);
```

Option 4

Example: Bulls and Cows

Bulls and Cows is a mind game played by two players. In the game, a random, 4-digit number is chosen and it's values are compared to those of another trial number. **All four digits of the number are different**. If any digit in the chosen number is the exact same value and in the exact same position as any digit in the trial number, this is called a bull. If the digit is present in both the trial number and chosen number, but is not in the same location, this is called a cow.



Example: Bulls and Cows

```
x_true = [1 2 3 4]; % true
x_test = [3 2 5 6]; % test

numb = 0; % number of Bull
if x_true(1) == x_test(1)
   numb = numb + 1;
end

if x_true(2) == x_test(2)
   numb = numb + 1;
end

if x_true(3) == x_test(3)
   numb = numb + 1;
end

if x_true(4) == x_test(4)
   numb = numb + 1;
end
```

Q: Write a script to compute "Bull" and assign its value to `num_b`. The true and test sequence is in `x_true` and `x_test`, respectively.

```
x_true = [1 2 3 4]; % true
x_test = [3 2 5 6]; % test

numb = 0; % number of Bull
for ii=1:4
    if x_true(ii) == x_test(ii)
        numb = numb + 1;
    end
end
```

😂: Much simple and readable.

Example: Bulls and Cows (Continue)

Q: Write a script to compute “Cows” + “Bulls” and assign its value to `num_c`. In other word, you need to compute how many same digits are present in both sequences. The true and test sequence is in `x_true` and `x_test`, respectively.

```
x_true = [1 2 3 4]; % true
x_test = [3 2 5 6]; % test

numc = 0;
for ii=1:4
    for jj=1:4
        if x_true(ii) == x_test(jj)
            numc = numc + 1;
        end
    end
end
```

Name	Value
x1	**[1 2 3 4]**
x2	**[3 2 5 6]**
numc	**2**

📖: “Cows” becomes “numc” – “numb” obtained from the previous code.

Break

- **break** command can be used to terminate a loop prematurely (while the comparison in the first line is still true.
- A **break** statement will cause termination of the smallest (closest) enclosing while or for loop.

```
1   vec = zeros(1,5);
2
3   for ii=1:5
4
5       if ii == 3
6           break;
7       end
8       vec(ii) = ii;
9
10  end
```

Name	Value
vec	**[1 2 0 0 0]**
ii	**3**

📖: When `ii` is equal to 3, `condition` in line 5 is `true` so `break` command is executed. Then, the code directly goes to `end` at line 10 (skip a script inside a loop that includes the `break` command.

Example: Use Break

Q: Find the first appearance location of ‘a’ in a character vector named as ‘char_seq’ and assign its location (index) to ‘loc’.

```
char_seq = 'Hello I am Matlab';

num_char = numel(char_seq);
loc = 0;
for ii=1:num_char
    if char_seq(ii) == 'a'
        loc = ii;
        break;
    end
end
```

Name	Value
char_seq	‘Hello I am Matlab’
loc	9
num_char	17
ii	9

Q1. Think about an advantage of the use of `break` in this problem

Q2. Think about what value is assigned to `loc` if `break` command is deleted in the script.

While Loop

- used as a **conditional** loop
- used to **repeat** an action when ahead of time it is not known how many times the action will be repeated
- general form:

```
while condition
  action
end
```

- *the **action** is repeated as long as the condition is true*
- Note: since the condition comes before the action, it is possible that the condition will be false the first time and then, the action will not be executed at all

⚠: If `action` fails to make `condition` `true`, an infinite loop can occur. Use **Ctrl-C** to break out of the infinite loop.

Example: Summation

Q. Summation of 1 to 10 using a while loop

```
1  num = 0;
2  val = 0;
3
4  while num < 11
5      val = val + num;
6      num = num+ 1;
7  end
8
```

Name	Value
num	11
val	55

```
while condition
  action
end
```

📖: It is frequently useful to count how many times the action of the loop has been repeated

📖: `condition` is the exist condition for the loop.

Example: Find a Location

Q: Find a location of the first 3rd of 5 in a vector named as `vec` and assign its location (index) to `loc`. Assume that there must be more three 5.

```
vec = [1 2 5 5 2 5 3 5 6];

num_5 = 0; % number of 5
idx = 0;

while num_5 < 3
    idx = idx + 1;
    if vec(idx) == 5
        num_5 = num_5 + 1;
    end
end

loc = idx;
```

Name	Value
vec	[1 2 5 5 2 5 3 5 6]
loc	6
num_5	3
idx	6

Example: Find a Location (Continue)

Optional

If you understand the operation of the while loop, you can design your code in many different ways.

```
vec = [1 2 5 5 2 5 3 5 6];

num_5 = 0; % number of 5
idx = 1;

while num_5 < 3
    if vec(idx) == 5
        num_5 = num_5 + 1;
        loc = idx;
    end
    idx = idx + 1;
end
```

```
vec = [1 2 5 5 2 5 3 5 6];

num_5 = 0; % number of 5
idx = 0;

while num_5 ~= 3
    idx = idx + 1;
    if vec(idx) == 5
        num_5 = num_5 + 1;
    end
end

loc = idx;
```

📖: When `num_5` is equal to 3, the while loop stop repeating but execute all script inside the loop (thus, `idx` becomes 7).

Example: Find a Location (Continue)

Optional

You can build the same operation using for-loop with break.

```
vec = [1 2 5 5 2 5 3 5 6];

num_5 = 0; % number of 5
idx = 0;

while num_5 < 3
    idx = idx + 1;
    if vec(idx) == 5
        num_5 = num_5 + 1;
    end
end

loc = idx;
```

```
vec = [1 2 5 5 2 5 3 5 6];

n_vec = numel(vec);
num_5 = 0;
loc = 0;

for ii=1:n_vec
    if vec(ii) == 5
        num_5 = num_5 + 1;
    end

    if num_5 == 3
        loc = ii;
        break;
    end
end
```

Dot Product

Remind

The dot product operation between two compatible vectors is defined as

$$\vec{a} \bullet \vec{b} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \bullet \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n = \sum_{k=1}^{n} a_k b_k$$

which is the linear combination of all elements in both vectors. The result is a scalar. Interestingly, this linear combination is the same as

$$\vec{a}^T \vec{b} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}^T \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix} \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \end{bmatrix}$$

The dot product is also known as the scalar product or inner product, and sometimes uses the syntax $\left(\vec{a}, \vec{b}\right) = \left\langle \vec{a}, \vec{b} \right\rangle = \vec{a} \bullet \vec{b}$.

Example: Dot Product

Q. Write a script to compute a dot product of A and B, which are the same size row vector. Assign its value to AB.

```
A = [3 4 5 6 8];
B = [5 6 7 7 8];

% Loop structure
n_elem = numel(A);

AB_vec = zeros(1, n_elem);

for ii=1:n_elem
    AB_vec(ii) = A(ii)*B(ii);
end
% or AB_vec = A.*B.

AB = 0;
for ii=1:n_elem
    AB = AB + AB_vec(ii);
end
```

```
A = [3 4 5 6 8];
B = [5 6 7 7 8];

% Loop structure
n_elem = numel(A);

AB = 0;
for ii=1:n_elem
    AB = AB + A(ii)*B(ii);
end
```

```
% Use of a built-in function
AB = dot(A,B);
```

```
% Use of a operator
AB = A * B';
```