

Computer Programming with MATLAB



Lesson 6: Loops

by

Akos Ledeczki and Mike Fitzpatrick



VANDERBILT
UNIVERSITY

Loops

- ▶ The loop is a new control construct that makes it possible to repeat a block of statements a number of times.
- ▶ We have already used loops without knowing it:

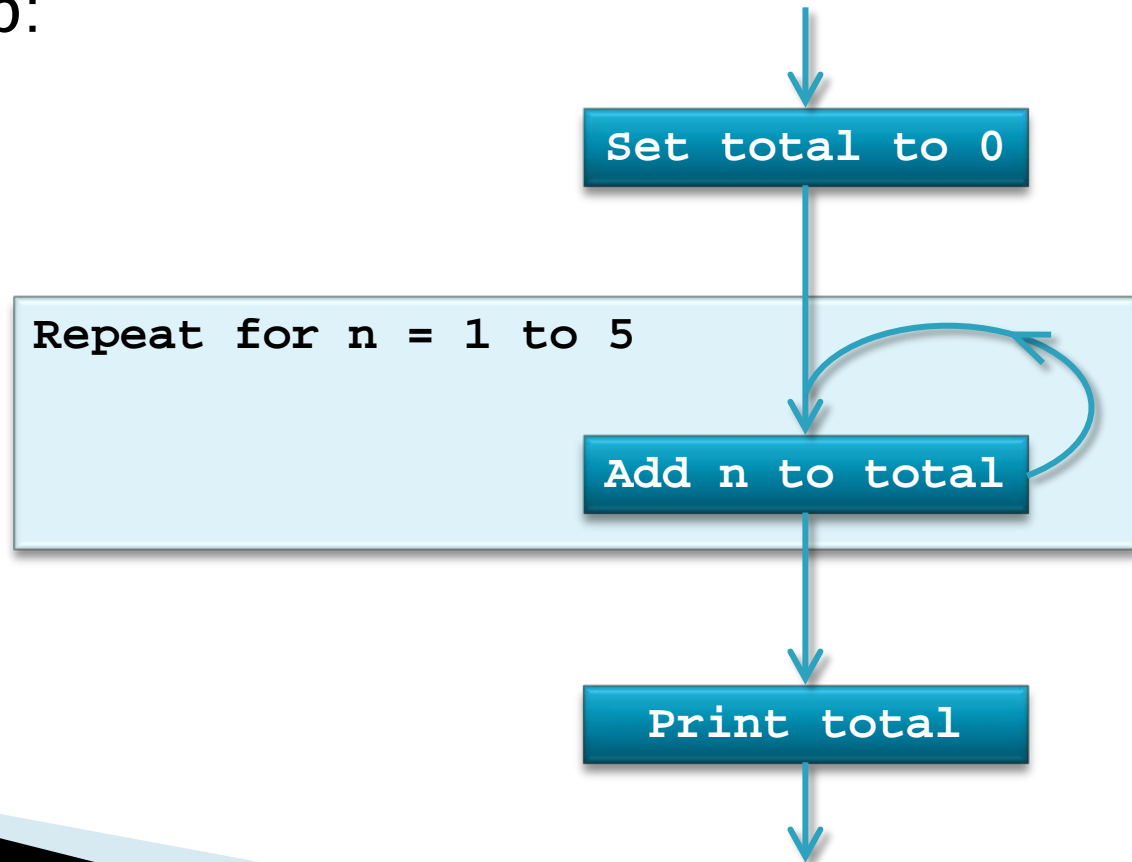
```
>> n = 1:5;  
>> total = sum(n);
```
- ▶ MATLAB uses loops internally both to compute the result of the colon operator and to compute the sum of the elements of the vector **n** above.
- ▶ Implicit loop



VANDERBILT
UNIVERSITY

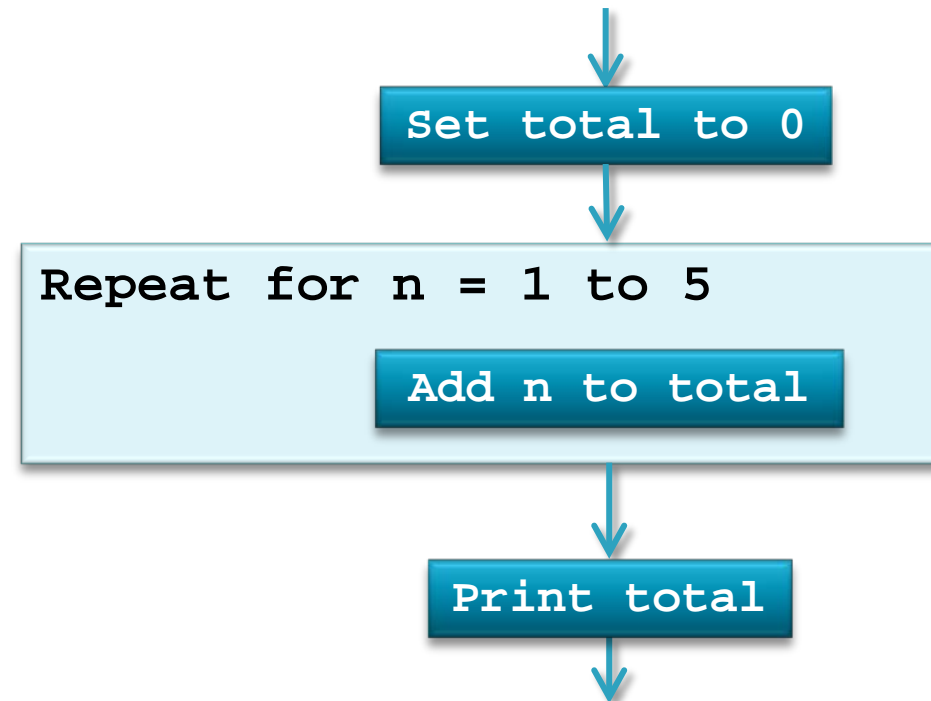
Schematic of a loop

- ▶ Let's compute the sum of 1 through 5 *without* using the built-in sum function!
- ▶ Use a loop:



Execution of a loop

- ▶ Set total to 0
- ▶ Set n to 1
- ▶ Execute Add n to total (total equals 1)
- ▶ Set n to 2
- ▶ Execute Add n to total (total equals 3)
- ▶ Set n to 3
- ▶ Execute Add n to total (total equals 6)
- ▶ Set n to 4
- ▶ Execute Add n to total (total equals 10)
- ▶ Set n to 5
- ▶ Execute Add n to total (total equals 15)
- ▶ Print total



for-loop

- ▶ MATLAB implementation using a for-loop:

```
total = 0;  
for n = 1:5  
    total = total + n;  
end  
fprintf('total equals %d\n',total);
```



VANDERBILT
UNIVERSITY

Parts of a for-loop

```
total = 0;
loop { for n = 1:5      _____ control statement
      total = total + n; _____ body
      end
      fprintf('total equals %d\n',total);
```

Colon operator is not required

- ▶ Here is another example:

```
list = rand(1,5); % assigns a row vector of random numbers
for x = list
    if x > 0.5
        fprintf('Random number %f is large.\n',x)
    else
        fprintf('Random number %f is small.\n',x)
    end
end
```

```
Random number 0.141890 is small.
Random number 0.421760 is small.
Random number 0.915740 is large.
Random number 0.792210 is large.
Random number 0.959490 is large.
```

Example revisited

- ▶ Notice that we do not need the list variable at all:

```
for x = rand(1,5)
    if x > 0.5
        fprintf('Random number %f is large.\n',x)
    else
        fprintf('Random number %f is small.\n',x)
    end
end
```


Observations

- ▶ The values assigned to the loop index do not have to be
 - integers,
 - regularly spaced, or
 - assigned in increasing order,
- ▶ In fact, they do not have to be scalars either:
 - The loop index will be assigned the columns of the array
- ▶ Any other control construct can be used in the body of the for-loop
 - if-statements
 - other loops
 - etc.



VANDERBILT
UNIVERSITY

while-loop

- ▶ for-loops work well when we know the number of necessary iterations before entering the loop
- ▶ Consider this problem:
 - Starting from 1, how many consecutive positive integers do we need to add together to exceed 50?
 - The only way to solve this with a for-loop is to guess a large enough number for the number of iterations and then use a break statement.
 - There is a better solution: a while-loop!



VANDERBILT
UNIVERSITY

while-loop example

```
function [n total] = possum(limit)
total = 0;
n = 0;
while total <= limit
    n = n + 1;
    total = total + n;
end
fprintf('sum: %d    count: %d\n', total, n);
```



VANDERBILT
UNIVERSITY

while-loop example

```
function [n total] = possum(limit)
total = 0;
n = 0;
while total <= limit
    n = n + 1;
    total = total + n;
end
fprintf('sum: %d    count: %d\n', total, n);
```

```
>> possum(50)
sum: 55    count: 10
```

```
ans =
```

```
10
```



VANDERBILT
UNIVERSITY

while-loop example

```
function [n total] = possum(limit)
total = 0;
n = 0;
loop { while total <= limit      control statement
      n = n + 1;
      total = total + n; }      body
end
fprintf('sum: %d    count: %d\n', total, n);
```

General form

```
while conditional  
    block  
end
```

```
if conditional  
    block  
end
```

► Difference:

- while condition is evaluated repeatedly
- block is executed repeatedly as long as condition is true



VANDERBILT
UNIVERSITY

Logical indexing

- ▶ Problem: given a vector, v , of scalars, create a second vector, w , that contains only the non-negative elements of v



VANDERBILT
UNIVERSITY

Logical indexing

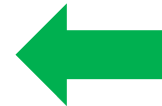
- ▶ Traditional solution:

```
w = [];  
jj = 0;  
for ii = 1:length(v)  
    if v(ii) >= 0  
        jj = jj + 1;  
        w(jj) = v(ii);  
    end  
end
```


Example revisited

- ▶ MATLAB provides a more elegant solution:

```
w = [];  
for ii = 1:length(v)  
    if v(ii) >= 0  
        w = [w v(ii)];  
    end  
end
```



Example with logical indexing

- ▶ The ultimate solution needs only a single line:

```
w = v(v >= 0);
```

- ▶ This is an example of logical indexing.
- ▶ To understand why and how this works, we need to introduce logical arrays