- Previous Lecture:
  - Nesting `if`-statements
  - Boolean operations (relational, logical)
  - Logical operators short-circuit
- Today's Lecture:
  - Iteration using `for`
- Announcements
  - Submit Project 2 in CMS tonight before 11pm
  - Use office hours or consulting hours if you have questions
  - Register your clicker!

Question

A stick of unit length is split into two pieces. The breakpoint is randomly selected. On average, how long is the shorter piece?

Physical experiment?◆
Thought experiment? → analysis
Computational experiment! → simulation◆

◆Need to repeat many trials!

Simulation:
use code to imitate the physical experiment

```
% one trial of the experiment
breakPt= rand(1);
if  breakPt<0.5
   shortPiece= breakPt;
else
   shortPiece= 1-breakPt;
end
```

```
% one trial of the experiment
breakPt= rand(1);
shortPiece= min(breakPt, 1-breakPt);
```

Want to do many trials, add up the lengths of the short pieces, and then divide by the number of trials to get the average length.

Repeat n times

```
% one trial of the experiment
breakPt= rand(1);
shortPiece= min(breakPt, 1-breakPt);
```

Take average

Print result

```
n= 10000;  % number of trials
total= 0;  % accumulated length so far

for  k= 1:n
   % one trial of the experiment
   breakPt= rand(1);
   shortPiece= min(breakPt, 1-breakPt);
   total= total + shortPiece;
end

aveLength= total/n;
fprintf('Average length is %f\n', ...
                            aveLength)
```

Example: “Accumulate” a solution

```
% Average 10 numbers from user input

n= 10;     % number of data values

for k= 1:n
  % read and process input value
    num= input('Enter a number: ');
    total= total + num;
end
ave= total/n;  % average of n numbers
fprintf('Average is %f\n', ave)
```

How many passes through the loop will be completed?

A: 0

B: 1

C: 9

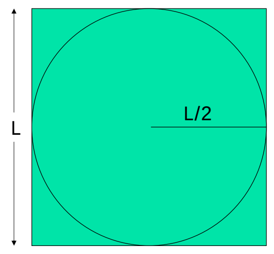
D: 10

E: 11

Important Features of Iteration

- A task can be accomplished if some steps are repeated; these steps form the loop body
- Need a starting point
- Need to know when to stop
- Need to keep track of (and measure) progress—update

Monte Carlo Approximation of π



Throw N darts

Sq. area = $N = L \times L$

Circle area = N_{in}

= $\pi L^2 / 4$

$\pi = 4\ N_{in}\ /\ N$

Monte Carlo Approximation of π

```
For each of N trials
    Throw a dart
    If it lands in circle
        add 1 to total # of hits

Pi is 4*hits/N
```

Monte Carlo π with N darts on L-by-L board

```
for k = 1:N
   % Throw kth dart


   % Count it if it is in the circle



end
myPi = 4*hits/N;
```

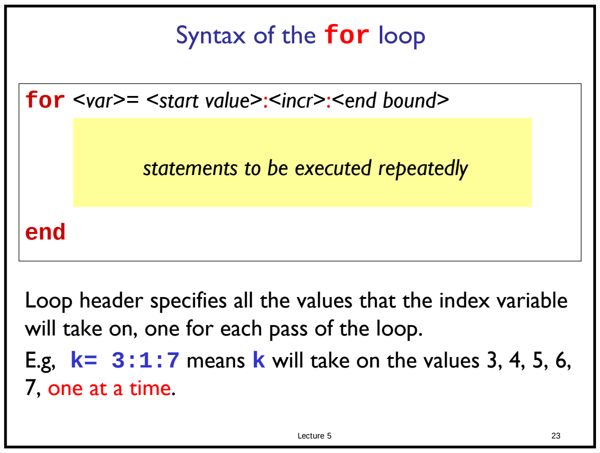
Syntax of the **for** loop

```
for <var>= <start value>:<incr>:<end bound>

      statements to be executed repeatedly

end
```

Loop body



Syntax of the **for** loop

for *<var>= <start value>:<incr>:<end bound>*

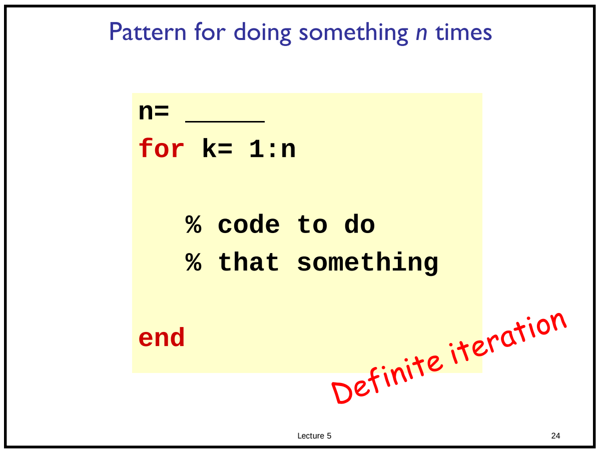
statements to be executed repeatedly

end

Loop header specifies all the values that the index variable will take on, one for each pass of the loop.

E.g, **k= 3:1:7** means **k** will take on the values 3, 4, 5, 6, 7, one at a time.

Lecture 5 23



Pattern for doing something *n* times

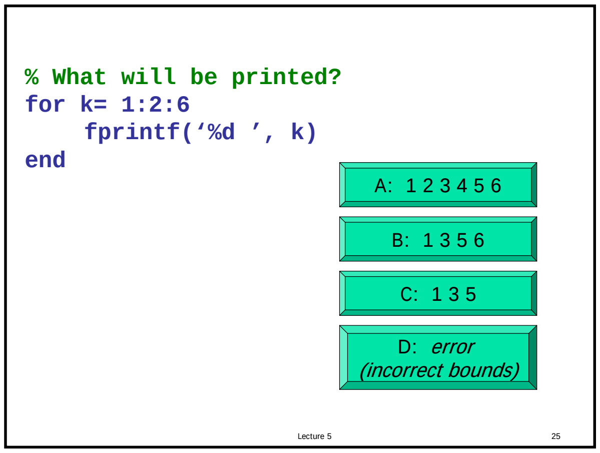
```
n= ______
for k= 1:n

   % code to do
   % that something

end
```

Definite iteration

Lecture 5 24



```
% What will be printed?
for k= 1:2:6
    fprintf('%d ', k)
end
```

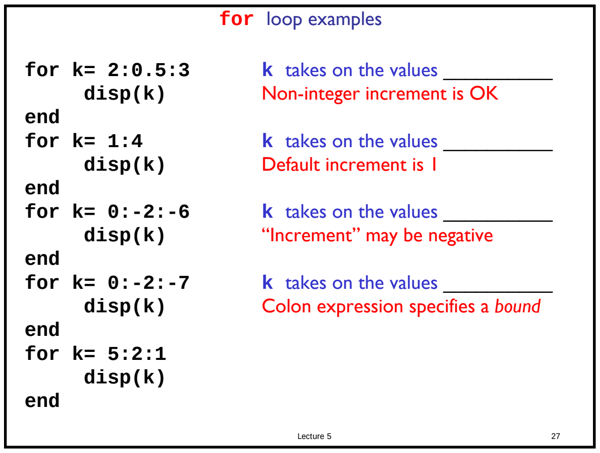
A: 1 2 3 4 5 6

B: 1 3 5 6

C: 1 3 5

D: *error (incorrect bounds)*

Lecture 5 25



for loop examples

`for k= 2:0.5:3`
`    disp(k)`
`end`
	k takes on the values ___________
	Non-integer increment is OK

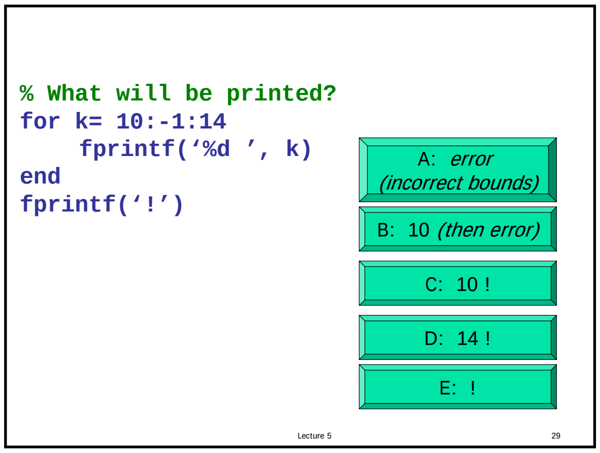
`for k= 1:4`
`    disp(k)`
`end`
	k takes on the values ___________
	Default increment is 1

`for k= 0:-2:-6`
`    disp(k)`
`end`
	k takes on the values ___________
	"Increment" may be negative

`for k= 0:-2:-7`
`    disp(k)`
`end`
	k takes on the values ___________
	Colon expression specifies a *bound*

`for k= 5:2:1`
`    disp(k)`
`end`	

Lecture 5 27



```
% What will be printed?
for k= 10:-1:14
    fprintf('%d ', k)
end
fprintf('!')
```

A: *error (incorrect bounds)*

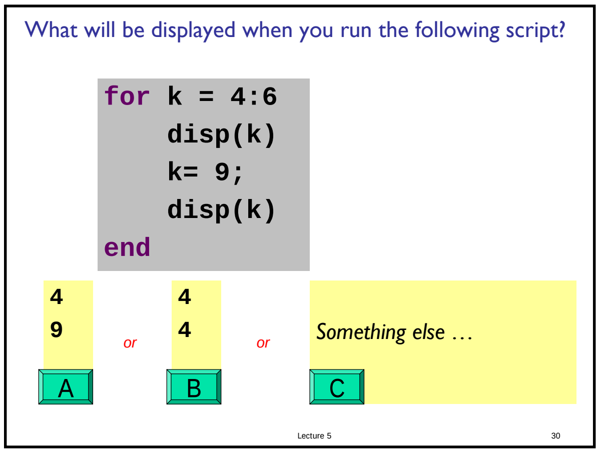
B: 10 *(then error)*

C: 10 !

D: 14 !

E: !

Lecture 5 29



What will be displayed when you run the following script?

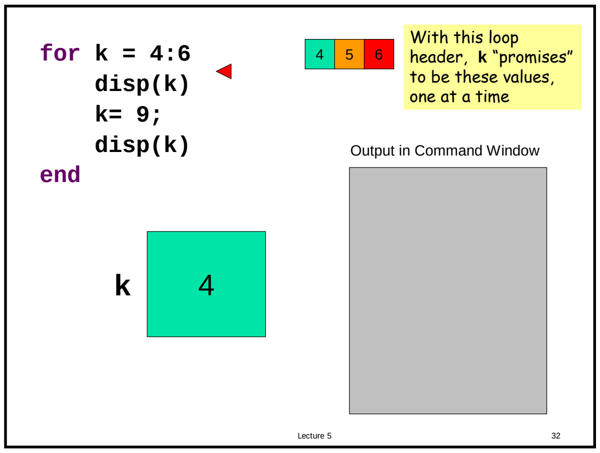
```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

A:
```
4
9
```
or

B:
```
4
4
```
or

C: *Something else …*

Lecture 5 30



```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

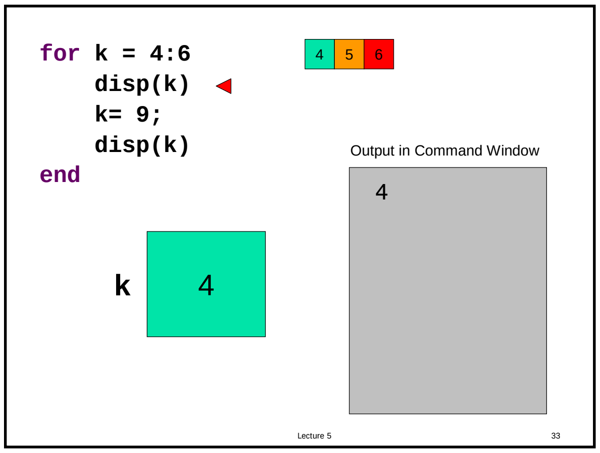
4 5 6

With this loop header, k "promises" to be these values, one at a time

Output in Command Window

k 4

Lecture 5 32



```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

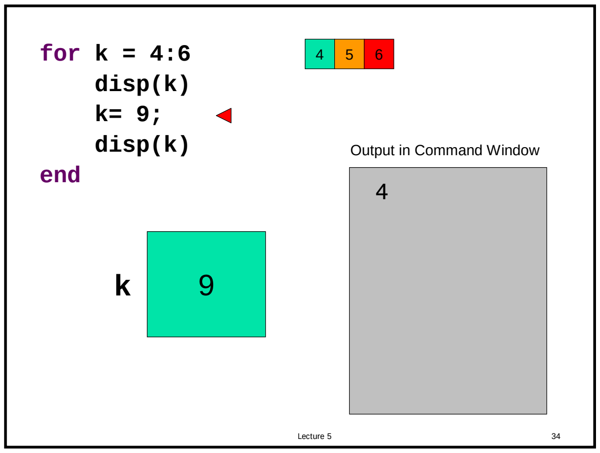
4 5 6

Output in Command Window

```
4
```

k 4

Lecture 5 33



```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

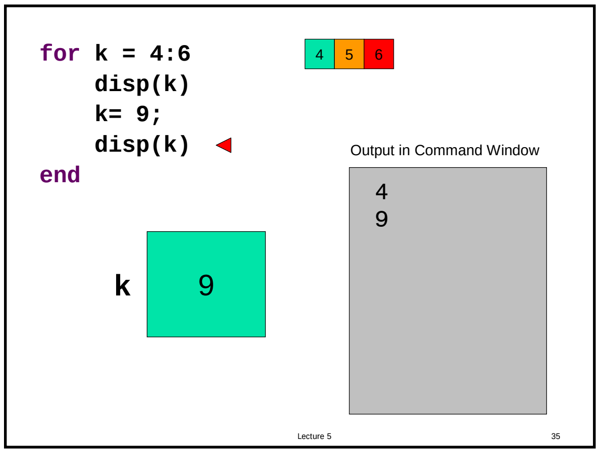
4 5 6

Output in Command Window

```
4
```

k 9

Lecture 5 34



```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

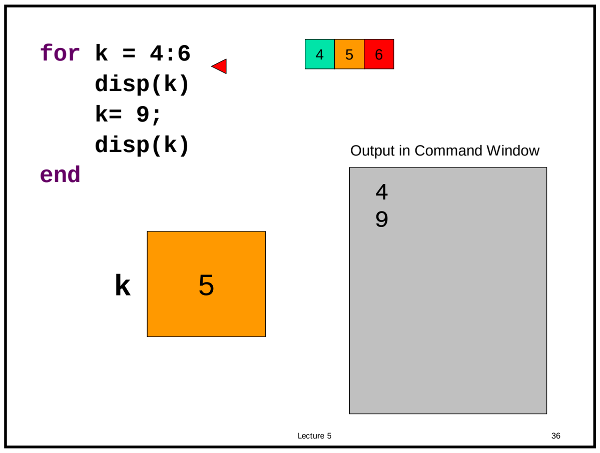
4 5 6

Output in Command Window

```
4
9
```

k 9

Lecture 5 35



```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

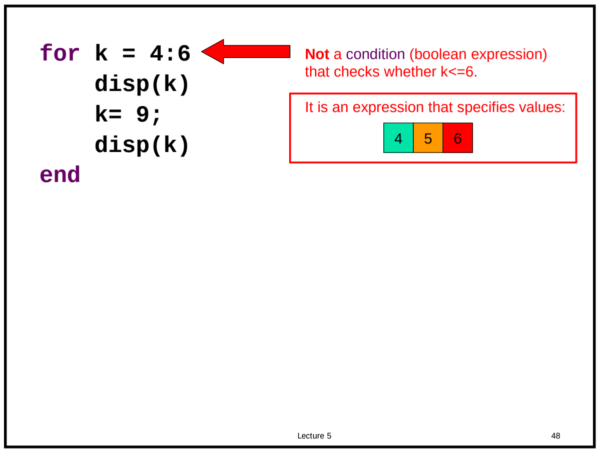
4 5 6

Output in Command Window

```
4
9
```

k 5

Lecture 5 36



```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

Not a condition (boolean expression) that checks whether k<=6.

It is an expression that specifies values:

4 5 6

Lecture 5 48