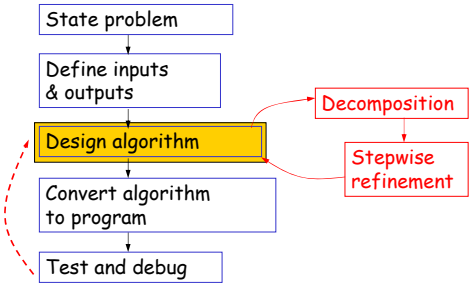
- Previous Lecture:
  - Branching
  - Logical operators and values
- Today's Lecture:
  - Iteration using **`for`**
  - Introduce **`while`**
- Announcement
  - Register your clicker!
  - Adhere to the Code of Academic Integrity
  - Section in classrooms this week

Top-Down Design



An algorithm is an idea. To use an algorithm you must choose a programming language and implement the algorithm.

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!

Question

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

A: .000001

B: .25

C: .333333

D: .499999

E: none of the above


```
% 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);
```

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)
```


Pattern for doing something *n* times

```
n= _____
for k= 1:n
    % code to do
    % that something
end
```

Definite iteration

Syntax of the **for** loop

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

statements to be executed

end

Loop body

Syntax of the **for** loop

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

statements to be executed

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.

for loop examples

```
for k= 2:0.5:3
     disp(k)
end
```
k takes on the values 2,2.5,3
Non-integer increment is OK

```
for k= 1:4
     disp(k)
end
```
k takes on the values 1,2,3,4
Default increment is 1

```
for k= 0:-2:-6
     disp(k)
end
```
k takes on the values 0,-2,-4,-6
"Increment" may be negative

```
for k= 0:-2:-7
     disp(k)
end
```
k takes on the values 0,-2,-4,-6
Colon expression specifies a *bound*

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

```
% 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)*


```
% 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: !

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

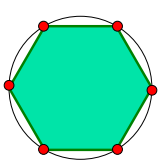
D: 10

E: 11

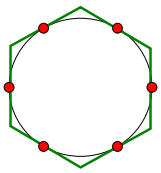
Important Features of Iteration

- 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

Example: *n*-gon → circle



Inscribed hexagon
$(n/2)\sin(2\pi/n)$



Circumscribed hexagon
$n\tan(\pi/n)$

As *n* approaches infinity, the inscribed and circumscribed areas approach the area of a circle. How big should *n* be?

Find *n* such that `outerA` and `innerA` converge

First, itemize the tasks:

- *define how close is close enough*
- *select an initial n*
- *calculate innerA, outerA for current n*
- *diff= outerA – innerA*
- *close enough?*
- *if not, increase n, repeat above tasks*