L7. Iteration with While-Loops

For-Loop Problems
Introduce While-Loops

Problem Solving With the For-Loop

```
for count variable = expression for starting value : expression for ending value
    The calculation
    to be repeated.
end
```

Question Time

How many lines of output are produced by the following script.

```
for k=100:200
   if rem(k,2)~=0
      disp('k');
   end
end
```

A. 2 B. 50 C. 51 D. 101

There is a line of output for every Odd number between 100 and 200.

Answer = 50

For-Loop Shortcoming

When you use a for-loop, you need to know the exact extent of the repetition.

```
for count variable = expression for starting value : expression for ending value
```

OK for Many Situations

Here is a typical for-loop Problem:

Simulate the tossing of a fair coin 100 times and print the number of "Heads"

```
% Running sum…
H = 0;
for tosses = 1:100
   r = rand;
   if r < .5
% Agree that this means "heads"
       H = H + 1;
   end
end
fprintf('H = %2d\n',H)
```

Not OK in Other Situations

Simulate the game of "Gap10":

Toss a fair coin until

| #Heads - #Tails | = 10

Score = number of required tosses

The number of required tosses is not known in advance.

What We Need

A loop that "shuts down" as soon as |H-T| = =10.

```
H = 0; T = 0; tosses = 0;

while abs(H-T)<10

   r = rand;
   tosses = tosses + 1;
   if r < .5
      H = H + 1;
   else
      T = T + 1;
   end

end
fprintf( … )
```

How a While-Loop Works

Warm-up exercise:

Any for-loop can be written as a while-loop.

A Simple For-loop

```
s = 0;
for k=1:5
   s = s + k;
   fprintf('%2d  %2d\n',k,s))
end
```

```
1   1
2   3
3   6
4  10
5  15
```

The While-loop Equivalent

```
k = 0; s = 0;
while k < 5
   k = k + 1; s = s + k;
   fprintf('%2d  %2d\n',k,s)
end
```

```
1   1
2   3
3   6
4  10
5  15
```

How it Works in Detail

```
k = 0; s = 0;
while k < 5
   k = k + 1; s = s + k;
   fprintf('%2d  %2d\n',k,s)
end
```

k and s are initialized

How it Works in Detail

```
k = 0; s = 0;
while k < 5
   k = k + 1; s = s + k;
   fprintf('%2d  %2d\n',k,s)
end
```

Is k < 5 true? Yes. Execute the loop body.

How it Works in Detail

```
k = 0; s = 0;
while k < 5
   k = k + 1; s = s + k;
   fprintf('%2d  %2d\n',k,s)
end
```

```
1   1
```

Is k < 5 true? Yes. Execute the loop body.

How it Works in Detail

```
k = 0; s = 0;
while k < 5
   k = k + 1; s = s + k;
   fprintf('%2d  %2d\n',k,s)
end
```

```
1   1
2   3
```

Is k < 5 true? Yes. Execute the loop body.

How it Works in Detail

```
k = 0; s = 0;
while k < 5
   k = k + 1; s = s + k;
   fprintf('%2d  %2d\n',k,s)
end
```

```
1   1
2   3
3   6
```

Is k < 5 true? Yes. Execute the loop body.

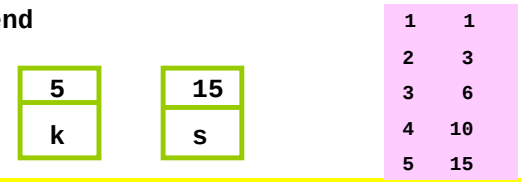
How it Works in Detail

```
k = 0; s = 0;
while k < 5
   k = k + 1; s = s + k;
   fprintf('%2d  %2d\n',k,s)
end
```

Is k < 5 true? Yes. Execute the loop body.

How it Works in Detail

```
k = 0; s = 0;
while k < 5
   k = k + 1; s = s + k;
   fprintf('%2d  %2d\n',k,s)
end
```



Is k < 5 true? No, done with loop.

A Modified Problem

Print the smallest k so that

$$1 + 2 + 3 + \ldots + k >= 30$$

How it Works in Detail

```
k = 0; s = 0;
while s < 30
   k = k + 1; s = s + k;
end
fprintf('%2d  %2d\n',k,s)
```

k and s are initialized

How it Works in Detail

```
k = 0; s = 0;
while s < 30
   k = k + 1; s = s + k;
end
fprintf('%2d  %2d\n',k,s)
```

Is s < 30 true? Yes. Execute loop body.

How it Works in Detail

```
k = 0; s = 0;
while s < 30
   k = k + 1; s = s + k;
end
fprintf('%2d  %2d\n',k,s)
```

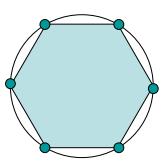
Is s < 30 true? Yes. Execute loop body.

Defining Variables

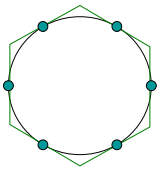
```
k = 0; s = 0;
while s < 30
   %s is the sum 1+ … + k
   k = k + 1; s = s + k;
end
```

This "property" is true all during the loop

Spotting a While Situation



InnerA = $(n/2)\sin(2\pi/n)$



Outer A = $n\tan(\pi/n)$

As *n* increases, InnerA and OuterA approach pi, the area of the unit circle.

When will |OuterA – InnerA| <= .000001?

PseudoCode Development

Repeat while |OuterA – InnerA| >.000001
  Increase n
  Update InnerA
  Update OuterA

Identify the repetition and a criteria that says "keep iterating".

PseudoCode Development

n = 3;
InnerA = area of inscribed triangle
OuterA = area of the circumscribed triangle
Repeat while |OuterA – InnerA| >.000001
  Increase n
  Update InnerA
  Update OuterA

The "players" have to be initialized

PseudoCode Development

n = 3;
InnerA = area of inscribed triangle
OuterA = area of the circumscribed triangle
Repeat while |OuterA – InnerA| >.000001
  Increase n
  Update InnerA
  Update OuterA
Print n

What to do after loop terminates.

Pattern for doing something an Indefinite number of times

```
% Initialization

while ( not-stopping signal )
   % do something

   % update status (variables)

end
```

Question Time

What is the last line of output produced by this script?

```
n = 5
while n>1
   disp('I dunno')
   if rem(n,2)==0
      n = n/2
   else
      n = 3*n+1
   end
end
```

A. 1 B. 2 C. 4 D. 16 E. I dunno

Two More While Examples

Each motivated by the limitations of the for-loop

Example 1: Up/Down Sequence

Pick a random whole number between one and a million. Call the number n and repeat this process:

if n is even, replace n by n/2.
if n is odd, replace n by 3n+1

Does it ever take more than 1000 updates to reach one?

Aside: Random Integers

How do we generate a random integer from an interval?

```
n = ceil(1000000*rand)
```

Need the Built-In Function `ceil`

a	floor(a)	ceil(a)
15.9	15	16
12.0	12	12

floor: next smallest integer
ceil : next biggest integer

Random Integers

```
n = ceil(1000000*rand)
```

```
% x is random real, 0 < x < 1
  x = rand
% y is random real, 0 < y < 10^6
  y = 100000*x
% n is rand integer from 1,…,10^6
  n = ceil(y)
```

The Central Repetition:

```
if rem(n,2)==0
   n = n/2;
else
  n = 3*n+1
end
```

Note cycling once n == 1:
1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, …

Shuts Down When n==1..

```
step = 0;
while n > 1
   if rem(n,2)==0
      n = n/2;
   else
      n = 3*n + 1;
   end
   step = step+1;
   fprintf(' %4d %7d\n',step,n)
end
```

Cycles after n ==1

```
for step = 1:1000
   if rem(n,2)==0
      n = n/2;
   else
      n = 3*n + 1;
   end
   fprintf(' %4d %7d\n',step,n)
end
```

Example 2: Square Roots

Pick a random number x between one and a million. Compute the sqrt(x) by

L = x; W = 1;
Repeat until relative error in L <= 10^-15:
    L = (L+W)/2; W = x/L;
    Print relative error in L

Shuts Down After Convergence

```
s = sqrt(x); L = x; W = 1; k = 0;
while k==0 || relErr > 10^-15
   k = k+1;
   L = (L+W)/2; W = x/L;
   relError = abs(L-s)/s
end
```

Shuts Down After Convergence

```
s = sqrt(x); L = x; W = 1; k = 0;
while          relErr > 10^-15
   k = k+1;
   L = (L+W)/2; W = x/L;
   relError = abs(L-s)/s
end
```

Error: relErr not initialized when the while Loop is entered.

Shuts Down After Convergence

```
s = sqrt(x); L = x; W = 1; k = 0;
while k==0 || relErr > 10^-15
   k = k+1;
   L = (L+W)/2; W = x/L;
   relError = abs(L-s)/s
end
```

During the first check of the condition, k==0 is true. Matlab doesn't bother to check the relErr comparison since the or is true. No prob that relErr uninitialized

Nested Loop Problem

On average, how many coin tosses are there in a game of Gap10?

Estimate by simulating 10,000 games.

PseudoCode

```
sum = 0
for k=1:10000
   Simulate a game of Gap10 and assign
      to the variable tosses the
      number of required tosses.
   sum = sum + tosses;
end
p = sum/10000
```

```
H = 0;  T = 0;  tosses = 0;
while abs(H-T)<10
   r = rand;
   tosses = tosses + 1;
   if r < .5
      H = H + 1;
   else
      T = T + 1;
   end
end
```