11. Iteration: The `while`-Loop

Topics:

- Open-Ended repetition
- the `while` statement
- Example 1: The sqrt Problem
- Example 2: The UpDown Sequence
- Example 3. The Fibonacci Sequence

Open-Ended Iteration

So far, we have only addressed iterative problems in which we know (in advance) the required number of repetitions.

Not all iteration problems are like that.

Some iteration problems are open-ended.

Stir for 5 minutes vs Stir until fluffy.

Example 1

The Square Root Problem (Again!)

For-Loop Solution

```
def sqrt(x):
    x = float(x)
    L = x
    W = 1
    for k in range(5):
        L = (L + W)/2
        W = x/L
    return L
```

The number of iterations is ``hardwired" into the implementation.

5 may not be enough-- an accuracy issue

5 may be too big-- efficiency issue

What we Really Want

```
def sqrt(x):
    x = float(x)
    L = x
    W = 1
    for k in range(5):
        L = (L + W)/2
        W = x/L
    return L
```

Iterate until L and W are really close.

What we Really Want

Not this:

```
for k in range(5):
    L = (L + W)/2
    W = x/L
```

But this:

```
while abs(L-W)/L > 10**-12
    L = (L + W)/2
    W = x/L
```

What we Really Want

```
while abs(L-W)/L > 10**-12
      L = (L + W)/2
      W = x/L
```

This says:

"Keep iterating as long as the discrepancy relative to L is bigger than 10**(-12)"

What we Really Want

```
while abs(L-W)/L > 10**-12
      L = (L + W)/2
      W = x/L
```

When the loop terminates, the discrepancy relative to L will be less than 10**(-12)

Template for doing something an Indefinite number of times

```
# Initializations
```

while *not-stopping condition* :

```
    # do something
```

A Common Mistake

```
while abs(L-W)/L < 10**-12
      L = (L + W)/2
      W = x/L
```

Forgetting that we want a "NOT stopping" condition

Example 2

The "Up/Down" Sequence

The Up/Down Sequence Problem

Pick a random whole number between one and a million. Call the number n and repeat this process until n ==1:

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

The Up/Down Sequence Problem

99	741	157	20	1
298	2224	472	10	4
149	1112	136	5	2
438	556	68	16	1
219	278	34	8	etc
658	139	17	4	
329	418	52	2	
988	209	26	1	
494	628	13	4	
247	314	40	2	

The Central Repetition

```
if m%2 == 0:
    m = m/2
else:
    m = 3*m+1
```

Note cycling once m == 1:

1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, ...

Shuts Down When m==1

```
n = input('m = ')
m = n
nSteps = 0
while m > 1:
    if m%2==0:
        m = m/2
    else:
        m = 3*m + 1
    nSteps = nSteps+1
print n,nSteps,m
```

nSteps keeps track of the number of steps

Avoiding Infinite Loops

```
nSteps = 0
maxSteps = 200
while m > 1 and nSteps<maxSteps:
    if m%2==0:
        m = m/2
    else:
        m = 3*m + 1
    nSteps = nStep+1
```

Example 3

Fibonacci Numbers and the Golden Ratio

Fibonacci Numbers and the Golden Ratio

Here are the first 12 Fibonacci Numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

The Fibonacci ratios 1/1, 2/1, 3/2, 5/3, 8/5 get closer and closer to the "golden ratio"

phi = (1 + sqrt(5))/2

Fibonacci Ratios 2/1, 3/2, 5/3, 8/5

1

2

2

3

3

5

5

8

Generating Fibonacci Numbers

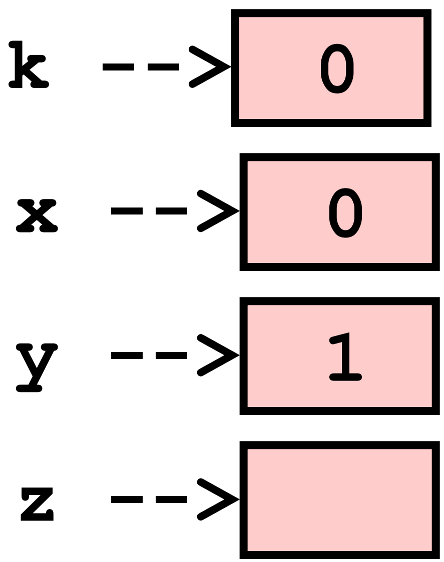
Here are the first 12 Fibonacci Numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

Starting here, each one is the sum
of its two predecessors

Generating Fibonacci Numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144




```
x = 0
y = 1
for k in range(10):
    z = x+y
    x = y
    y = z
```

Generating Fibonacci Numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

```
k --> 0
x --> 1
y --> 1
z --> 1
```

```
x = 0
y = 1
for k in range(10):
    z = x+y
    x = y
    y = z
```

Generating Fibonacci Numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

```
k  -->  1
x  -->  1
y  -->  1
z  -->  1
```

```
x = 0
y = 1
for k in range(10):
    z = x+y
    x = y
    y = z
```

Generating Fibonacci Numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

```
k -->  1
x -->  1
y -->  2
z -->  2
```

```
x = 0
y = 1
for k in range(10):
    z = x+y
    x = y
    y = z
```

Generating Fibonacci Numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

```
k --> 2
x --> 1
y --> 2
z --> 2
```

```
x = 0
y = 1
for k in range(10):
    z = x+y
    x = y
    y = z
```

Generating Fibonacci Numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

```
k --> 2
x --> 2
y --> 3
z --> 3
```

```
x = 0
y = 1
for k in range(10):
    z = x+y
    x = y
    y = z
```

Generating Fibonacci Numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

k --> 3

x --> 2

y --> 3

z --> 3

```
x = 0
y = 1
for k in range(10):
    z = x+y
    x = y
    y = z
```

Generating Fibonacci Numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

```
k --> 3
x --> 3
y --> 5
z --> 5
```

```
x = 0
y = 1
for k in range(10):
    z = x+y
    x = y
    y = z
```

Generating Fibonacci Numbers

```
x = 0
print x
y = 1
print y
for k in range(6):
    z = x+y
    x = y
    y = z
    print z
```

```
 0
 1
 1
 2
 3
 5
 8
13
```

Generating Fibonacci Numbers

```
x = 0
print x
y = 1
print y
for k in range(6):
    z = x+y
    x = y
    y = z
    print z
```

```
x = 0
print x
y = 1
print y
k = 0
while k<6:
    z = x+y
    x = y
    y = z
    print z
    k = k+1
```

Print First Fibonacci Number >= 1000000

```
x = 0
y = 1
z = x + y
while y < 1000000:
    x = y
    y = z
    z = x + y
print y
```

Print First Fibonacci Number >= 1000000

```
past = 0
current = 1
next = past + current
while current < 1000000:
    past = current
    current = next
    next = past + current
print current
```

```
1346269
```

Print First Fibonacci Number >= 1000000

```
past = 0
current = 1
next = past + current
while current < 1000000:
    past = current
    current = next
    next = past + current
print current
```

Reasoning. When the while loop terminates, it will be the first time that `current>=` 1000000 is true. By print out current we see the first fib >= **million**

Print Largest Fibonacci Number < 1000000

```
past = 0
current = 1
next = past + current
while next <= 1000000:
    past = current
    current = next
    next = past + current
print current
```

```
832040
```

Print Largest Fibonacci Number < 1000000

```
past = 0
current = 1
next = past + current
while next < 1000000:
    past = current
    current = next
    next = past + current
print current
```

Reasoning. When the while loop terminates, it will be the first time that **next>=** 1000000 is true. Current has to be < 1000000. And it is the largest fib with this property

Fibonacci Ratios

```
past = 0
current = 1
next = past + current
while next <= 1000000:
    past = current
    current = next
    next = past + current
    print next/current
```

```
1.000000000000
2.000000000000
1.500000000000
1.666666666667
1.600000000000
1.625000000000
1.615384615385
1.619047619048
1.617647058824
1.618181818182
1.617977528090
1.618055555556
1.618025751073
1.618037135279
1.618032786885
      :
```

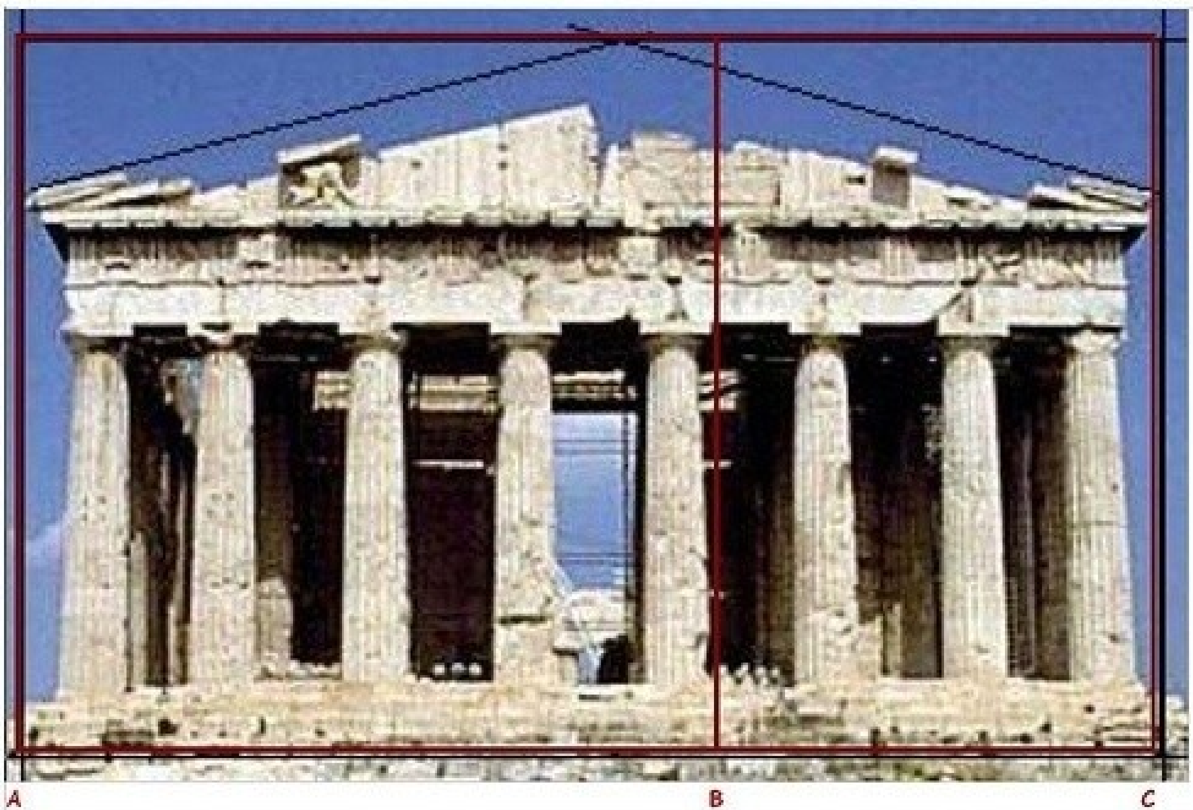
Heading towards the Golden ratio = (1+sqrt(5))/2

Fibonacci Ratios

```
past = 0
current = 1
next = past + current
k = 1
phi = (1+math.sqrt(5))/2
while abs(next/current - phi) > 10**-9
    past = current
    current = next
    next = past + current
    k = k+1
print k,next/current
```

23	1.618033988749

Most Pleasing Rectangle



1

(1+sqrt(5))/2