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

```python
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
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 \( \frac{\text{abs}(L-W)}{L} > 10^{-12} \)

\[
L = \frac{(L + W)}{2} \\
W = \frac{x}{L}
\]

This says:

“Keep iterating as long as the discrepancy relative to \( L \) is bigger than \( 10^{**}(-12) \)”
What we Really Want

\[
\text{while } \frac{|L-W|}{L} > 10^{-12} \\
L = \frac{(L + W)}{2} \\
W = \frac{x}{L}
\]

When the loop terminates, the discrepancy relative to L will be less than \(10^{-12}\)
Template for doing something an Indefinite number of times

```plaintext
# Initializations

while not-stopping condition :

# do something
```
A Common Mistake

while \( \text{abs}(L-W)/L < 10^{-12} \)

\[
L = (L + W)/2
\]

\[
W = x/L
\]

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

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>99</td>
<td>741</td>
<td>157</td>
<td>20</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>298</td>
<td>2224</td>
<td>472</td>
<td>10</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>149</td>
<td>1112</td>
<td>136</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>438</td>
<td>556</td>
<td>68</td>
<td>16</td>
<td>1</td>
<td>etc</td>
</tr>
<tr>
<td>219</td>
<td>278</td>
<td>34</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>658</td>
<td>139</td>
<td>17</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>329</td>
<td>418</td>
<td>52</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>988</td>
<td>209</td>
<td>26</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>494</td>
<td>628</td>
<td>13</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>247</td>
<td>314</td>
<td>40</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Central Repetition

\[
\begin{align*}
\text{if } m \% 2 &= 0: \\
&\quad \quad \quad m = m/2 \\
\text{else:} \\
&\quad \quad \quad m = 3*m+1
\end{align*}
\]

Note cycling once \( m == 1 \):

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

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

\[ \phi = \frac{1 + \sqrt{5}}{2} \]
Fibonacci Ratios 2/1, 3/2, 5/3, 8/5

1  2  3  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

\[ x = 0 \]
\[ y = 1 \]
\[ \text{for } k \text{ in range}(10): \]
\[ z = x+y \]
\[ x = y \]
\[ y = z \]
Generating Fibonacci Numbers

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

\[
x = 0 \\
y = 1 \\
\text{for } k \text{ in range}(10): \\
\quad z = x + y \\
\quad x = y \\
\quad y = z
\]
Generating Fibonacci Numbers

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

\[\begin{array}{c|c}
k & 1 \\
x & 1 \\
y & 1 \\
z & 1 \\
\end{array}\]

\[\begin{array}{c}
x = 0 \\
y = 1 \\
\text{for } k \text{ in range}(10):
\end{array}\]
\[\begin{array}{c}
z = x+y \\
x = y \\
y = z \\
\end{array}\]
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

\[ x = 0 \\ y = 1 \\ \text{for } k \text{ in range(10)}: \]
\[ z = x+y \\ x = y \\ y = z \]
Generating Fibonacci Numbers

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

\[ x = 0 \]
\[ y = 1 \]
\[ \text{for } k \text{ in range}(10): \]
\[ \quad z = x+y \]
\[ \quad x = y \]
\[ \quad y = z \]
Generating Fibonacci Numbers

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

\[
\begin{align*}
x & = 0 \\
y & = 1 \\
\text{for } k \text{ in range}(10): \\
z & = x+y \\
x & = y \\
y & = z
\end{align*}
\]
Generating Fibonacci Numbers

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

\[ \begin{align*}
  k & \rightarrow 3 \\
  x & \rightarrow 3 \\
  y & \rightarrow 5 \\
  z & \rightarrow 5
\end{align*} \]

\[ \begin{align*}
  x &= 0 \\
  y &= 1 \\
  \text{for } k \text{ in range}(10): \\
  z &= x+y \\
  x &= y \\
  y &= z
\end{align*} \]
Generating Fibonacci Numbers

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

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>13</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Generating Fibonacci Numbers

\begin{align*}
x & = 0 \\
\text{print } x \\
y & = 1 \\
\text{print } y \\
\text{for } k \text{ in range}(6): \\
& \quad z = x+y \\
& \quad x = y \\
& \quad y = z \\
& \quad \text{print } z
\end{align*}

\begin{align*}
x & = 0 \\
\text{print } x \\
y & = 1 \\
\text{print } y \\
k & = 0 \\
\text{while } k < 6: \\
& \quad z = x+y \\
& \quad x = y \\
& \quad y = z \\
& \quad \text{print } z \\
& \quad k = k+1
\end{align*}
Print First Fibonacci Number
>= 1000000

```python
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
\[ \geq 1000000 \]

\[
\begin{align*}
past &= 0 \\
current &= 1 \\
next &= past + current \\
\text{while } current < 1000000: \\
    past &= current \\
    current &= next \\
    next &= past + current \\
\text{print } current
\end{align*}
\]

Reasoning. When the while loop terminates, it will be the first time that 
\[ current \geq 1000000 \] is true. By print out current we see the first fib \[ \geq \text{ million} \]
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

```python
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

Heading towards the Golden ratio = (1+\sqrt{5})/2
Fibonacci Ratios

```python
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

\[(1+\sqrt{5})/2\]