11. More on While and Boolean-Valued Functions

Topics:
- Reasoning about While Loops
- Designing Boolean-Valued Functions

Four Examples

1. Random Walk
2. Fibonacci numbers and the Golden Ratio
3. A Spiral Problem
4. Detecting streaks in a coin toss sequence

Random Walk

Tiles 1x1
Middle tile has center (0,0)
Robot starts at center tile
Hops according to coin flip
Heads: Hop left
Tails: Hop right
Simulation over when robot hops off runway

from random import randint as randi
x = 0
while abs(x) <= 5:
    r = randi(1, 2)
    if r == 1:
        x = x + 1
    else:
        x = x - 1

Random Walk

-5 -4 -3 -2 -1 0 1 2 3 4 5

x = 0
while abs(x) <= 5:
    r = randi(1, 2)
    if r == 1:
        x = x + 1
    else:
        x = x - 1
Random Walk

x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
Random Walk

```python
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk

```python
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk

```python
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```

Random Walk

```python
x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
```
Random Walk

x = 0
while abs(x)<=5:
    r = randi(1,2)
    if r == 1:
        x = x+1
    else:
        x = x-1
Random Walk

\[ x = 0 \]
\[
\text{while abs}(x) \leq 5:
\]
\[
\text{if } r = \text{randi}(1,2):
\]
\[
\quad x = x+1
\]
\[
\text{else:}
\]
\[
\quad x = x-1
\]
2. 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”

\[
\phi = \frac{1 + \sqrt{5}}{2}
\]

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:

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

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

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

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

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

Print First Fibonacci Number

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

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 fibonacci number >= million

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

```
1346269
```

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
```

```
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 Fibonacci 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
```

```
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
```

Most Pleasing Rectangle

```
(1+sqrt(5))/2
```

3. A Spiral Problem

Recall:

```
DrawSpiral(N,t,c1,c2,c3)
draws a spiral and
SpiralRadius(N,t)
computes its radius.
```
The Gist of SpiralRadius

\[ xc = 0; \ yc = 0; \ r = 0 \]
for k in range(N):
    \[
    \text{theta} = (k*t) * \text{math.pi/180} \\
    L = k+1 \\
    \# (xc,yc) = \text{end of the kth edge} \\
    xc = xc + L*\text{math.cos(theta)} \\
    yc = yc + L*\text{math.sin(theta)} \\
    \text{dist} = \text{math.sqrt(xc**2+yc**2)} \\
    r = \max(r,\text{dist}) \\
    \]
return r

The Heading

For the k-th edge, here is the heading in radians:

\[
\text{theta} = (k*t) * \text{math.pi/180} \\
\]
t is the turn angle in degrees

The Ending Endpoint

Before: (xc,yc) is where the kth edge starts

\[
xc = xc + L*\text{math.cos(theta)} \\
yc = yc + L*\text{math.sin(theta)} \\
\]
After: (xc,yc) is where the kth edge ends

The Gist of SpiralRadius

\[ xc = 0; \ yc = 0; \ r = 0 \]
for k in range(N):
    \[
    \text{theta} = (k*t) * \text{math.pi/180} \\
    L = k+1 \\
    \# (xc,yc) = \text{end of the kth edge} \\
    xc = xc + L*\text{math.cos(theta)} \\
    yc = yc + L*\text{math.sin(theta)} \\
    \text{dist} = \text{math.sqrt(xc**2+yc**2)} \\
    r = \max(r,\text{dist}) \\
    \]
return r
Computing the max Distance

Is the end of the kth edge further away from (0,0) than all previous endpoints?

\[ \text{dist} = \sqrt{x_c^2 + y_c^2} \]
\[ r = \max(r, \text{dist}) \]

\[ \text{dist} = \sqrt{x_c^2 + y_c^2} \]
if dist > r:
    \[ r = \text{dist} \]

A Reverse Problem

Given the turn angle \( t \) and a radius \( r \), what is the largest \( N \) so that

\[ \text{DrawSpiral}(N, t, c1, c2, c3) \]
fits inside the circle

\[ x^2 + y^2 = r^2 \]

Example

The circle has radius \( r = 500 \).
\[ \text{DrawSpiral}(513, 62, ...) \]
just fits inside

Example

The circle has radius \( r = 500 \).
\[ \text{DrawSpiral}(856, 162, ...) \]
just fits inside

Let's Design a Function that Returns This Integer

\[ k = 0 \quad \# \text{Index of current edge} \]
Compute endpoint distance to (0,0)
while endpoint inside circle
\[ k = k+1 \]
    \[ \text{Compute endpoint dist to (0,0)} \]
\[ N = k \]
return N

The Body of \( nEdges \)
4. Streaks in a Coin Toss Sequence

```python
k = 0
xc = 1
yc = 0
d = math.sqrt(xc**2 + yc**2)
while d<=r:
    k = k+1
    theta = (k*t)*math.pi/180
    xc = xc + (k+1)*math.cos(theta)
    yc = yc + (k+1)*math.sin(theta)
    d = math.sqrt(xc**2 + yc**2)
return k-1
```

**Coin Toss Strings**

*A coin toss string is made up of H's and T's.*

\[
s = \text{''HHTHTTTTHTHHTHTTTT''}
\]

\[
s = \text{''HHTHTTTTHTHHTHTTT''}
\]

**Streaks**

\[
s = \text{''HHTHTTTTHTHHTHTTTT''}
\]

\[
s[0:2] \quad \text{a length-2 streak}
\]

\[
s[4:7] \quad \text{a length-3 streak}
\]

\[
s = \text{''HHTHTTTTHTHHTHTTTT''}
\]

\[
s[12:17] \quad \text{a length-5 streak}
\]
**Streak Definition**

\[ s[k:k+n] \] is a length-\( n \) streak if

1. \( k+n \leq \text{len}(s) \)
2. It is either all T’s or all H’s
3. If there is a character before \( s[k] \), it is different from \( s[k] \).
4. If there is a character after \( s[k+n] \), it is different from \( s[k+n] \).

**Streaks**

\[ s = 'HHTHTTHTHTHTTTTT' \]

\[ s[5:7] \] is NOT a length-2 streak

**isStreak(s,k,n)**

\[
t = s[k:k+n]
\]

if \( k+n > \text{len}(s) \):
    return False
elif \( t\text{.count('H')} < n \) and \( t\text{.count('T')} < n \):
    return False
elif \( k > 0 \) and \( s[k-1] == s[k] \):
    return False
elif \( (k+n < \text{len}(s)) \) and \( s[k+n-1] == s[k+n] \):
    return False
else:
    return True

**Using isStreak to Find Streaks**

\[ s = 'HHTHTTHTHTHTTTTT' \]

\[ k \text{ isStreak}(s,k,3) \]

<table>
<thead>
<tr>
<th>( k )</th>
<th>( \text{isStreak}(s,k,3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>1</td>
<td>False</td>
</tr>
</tbody>
</table>

\[ s = 'HHTHTTHTHTHTTTTT' \]

\[ k \text{ isStreak}(s,k,3) \]

<table>
<thead>
<tr>
<th>( k )</th>
<th>( \text{isStreak}(s,k,3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>1</td>
<td>False</td>
</tr>
<tr>
<td>2</td>
<td>False</td>
</tr>
</tbody>
</table>
Using `isStreak` to Find Streaks

\[ s = \text{`HHTTHTHHHTHTTT'} \]

<table>
<thead>
<tr>
<th>k</th>
<th><code>isStreak(s,k,3)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>1</td>
<td>False</td>
</tr>
<tr>
<td>2</td>
<td>False</td>
</tr>
<tr>
<td>3</td>
<td>False</td>
</tr>
</tbody>
</table>

Using `isStreak` to Find Streaks

\[ s = \text{`HHTTHTHHHTHTTT'} \]

<table>
<thead>
<tr>
<th>k</th>
<th><code>isStreak(s,k,3)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>1</td>
<td>False</td>
</tr>
<tr>
<td>2</td>
<td>False</td>
</tr>
<tr>
<td>3</td>
<td>False</td>
</tr>
<tr>
<td>4</td>
<td>True</td>
</tr>
</tbody>
</table>

Using `isStreak` to Find Streaks

```python
def FindStreak(s,n):
    k=0
    while k<len(s) and (not `isStreak(s,k,n)`):
        # s[k:k+n] is not a streak
        k = k+1
    if k<len(s):
        # `isStreak(s,k,n)` is True
        return k
    else:
        # k==len(s) is True
        return -1
```