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
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
Random Walk

```python
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

\[ x = 0 \]

while \( \text{abs}(x) \leq 5 \):
  
  \[ r = \text{randi}(1,2) \]

  if \( r == 1 \):
    \[ x = x+1 \]
  else:
    \[ x = x-1 \]
\[ x = 0 \]

\[ \text{while abs(x) <= 5:} \]

\[ r = \text{randi}(1, 2) \]

\[ \text{if } r == 1: \]

\[ x = x+1 \]

\[ \text{else:} \]

\[ x = x-1 \]
Random Walk

x = 0

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

\[ x = 0 \]

\[ \text{while } \text{abs}(x) \leq 5: \]

\[ r = \text{randi}(1,2) \]

\[ \text{if } r == 1: \]

\[ x = x+1 \]

\[ \text{else:} \]

\[ x = x-1 \]
Random Walk

\[ x = 0 \]

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

\[ x = 0 \]

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

\[
\begin{align*}
\text{x} &= 0 \\
\text{while abs(x) <= 5:} & \\
& \quad r = \text{randi}(1,2) \\
& \quad \text{if r == 1:} \\
& \quad \quad \text{x = x+1} \\
& \quad \text{else:} \\
& \quad \quad \text{x = x-1}
\end{align*}
\]
Random Walk

\[ x = 0 \]

\[
\text{while abs}(x) <= 5:\n\]

\[
\text{if } r == 1: \]

\[
x = x+1
\]

\[
\text{else:}
\]

\[
x = x-1
\]
Random Walk

x = 0

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

\[ x = 0 \]

while abs(x) <= 5:

\[ r = \text{randi}(1,2) \]

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

\[ x = 0 \]

while abs(x) <= 5:
    \[ r = \text{randi}(1,2) \]
    if r == 1:
        \[ x = x + 1 \]
    else:
        \[ x = x - 1 \]
Random Walk

\[ x = 0 \]

while abs(x) <= 5:
    \[ r = \text{randi}(1,2) \]
    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

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

\[ x = 0 \]

\[ \text{while abs}(x) \leq 5: \]

\[ r = \text{randi}(1,2) \]

\[ \text{if } r == 1: \]

\[ x = x + 1 \]

\[ \text{else:} \]

\[ x = x - 1 \]
Random Walk

\[
x = 0 \\
\text{while abs}(x) \leq 5:\n\text{if } r == 1:\n\quad x = x+1 \\
\text{else:}\n\quad x = x-1
\]
Random Walk

\[ x = 0 \]

while abs(x) <= 5:
    \[ r = \text{randi}(1,2) \]
    if r == 1:
        \[ x = x + 1 \]
    else:
        \[ x = x - 1 \]
Random Walk

\[
\begin{align*}
x &= 0 \\
\text{while abs}(x) &\leq 5: \\
r &= \text{randi}(1,2) \\
\text{if } r &= 1: \\
\quad &x = x+1 \\
\text{else:} \\
\quad &x = x - 1
\end{align*}
\]
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:
    \[
r = \text{randi}(1,2)
    \]
    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

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:
    \[ r = \text{randi}(1,2) \]
    if r == 1:
        \[ x = x + 1 \]
    else:
        \[ x = x - 1 \]
x = 0

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

\[ \begin{array}{l}
k \rightarrow 0 \\
x \rightarrow 0 \\
y \rightarrow 1 \\
z \rightarrow \\
\end{array} \]

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

\[
x = 0 \\
y = 1 \\
\text{for } k \text{ in range}(10):
\]

\[
\begin{align*}
z &= x + y \\
x &= y \\
y &= z
\end{align*}
\]
Generating Fibonacci Numbers

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

\[ x = 0 \\
\] \[ y = 1 \]

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

\[
\begin{align*}
    & k \rightarrow 2 \\
    & x \rightarrow 1 \\
    & y \rightarrow 2 \\
    & z \rightarrow 2 \\
\end{align*}
\]

\[
x = 0 \\
y = 1 \\
\text{for } k \text{ in range}(10):
\]

\[
\begin{align*}
    & z = x + y \\
    & x = y \\
    & y = z
\end{align*}
\]
Generating Fibonacci Numbers

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

\[ x = 0 \]
\[ y = 1 \]

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

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

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

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

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

```python
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

```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

```python
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

def main():
    past = 0
    current = 1
    next = past + current
    while next <= 1000000:
        past = current
        current = next
        next = past + current
    print current

if __name__ == '__main__':
    main()
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
```

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
3. A Spiral Problem
A Spiral Problem

Recall:

\textbf{DrawSpiral}(N,t,c_1,c_2,c_3)

draws a spiral and

\textbf{SpiralRadius}(N,t)

computes its radius.
The Gist of SpiralRadius

```python
xc = 0; yc = 0; r = 0
for k in range(N):
    theta = (k*t)*math.pi/180
    L = k+1
    # (xc,yc) = end of the kth edge
    xc = xc + L*math.cos(theta)
    yc = yc + L*math.sin(theta)
    dist = math.sqrt(xc**2+yc**2)
    r = max(r,dist)
return r
```
The Gist of SpiralRadius

\[ xc = 0; \ yc = 0; \ r = 0 \]

\[ \text{for } k \text{ in range}(N): \]

\[ \text{theta} = (k*t) * \text{math.pi} / 180 \]

\[ L = k+1 \]

\[ \# (xc,yc) = \text{end of kth edge} \]

\[ xc = xc + L*\text{math.cos}(\text{theta}) \]

\[ yc = yc + L*\text{math.sin}(\text{theta}) \]

\[ \text{dist} = \text{math.sqrt}(xc**2+yc**2) \]

\[ r = \max(r,\text{dist}) \]

\[ \text{return } r \]
The Heading

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

\[ \theta = (k \times t) \times \frac{\text{math.pi}}{180} \]

\( t \) is the turn angle in degrees
The Gist of SpiralRadius

xc = 0; yc = 0; r = 0
for k in range(N):
    theta = (k*t)*math.pi/180
    L = k+1
    # (xc,yc) = end of kth edge
    xc = xc + L*math.cos(theta)
    yc = yc + L*math.sin(theta)
    dist = math.sqrt(xc**2+yc**2)
    r = max(r,dist)
return r
The Ending Endpoint

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

\[
\begin{align*}
x_c &= x_c + L \cdot \text{math.cos}(\theta) \\
y_c &= y_c + L \cdot \text{math.sin}(\theta)
\end{align*}
\]

After: \((xc, yc)\) is where the kth edge ends
The Gist of SpiralRadius

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

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

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

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

\[
x^2 + y^2 = r^2
\]
The circle has radius \( r = 500 \).

\textbf{DrawSpiral}\((513,62,...)\)

just fits inside
The circle has radius $r = 500$.

$\text{DrawSpiral}(856,162,...)$ just fits inside
Let's Design a Function that Returns This Integer $nEdges$.

$t = \text{turn angle}$

$r = \text{radius}$

$N = \text{max number edges so spiral radius} \leq r$
The Body of \texttt{nEdges}

\begin{verbatim}
\texttt{k = 0  \# Index of current edge}
Compute endpoint distance to (0,0)
while endpoint inside circle
    \texttt{k = k+1}
    Compute endpoint dist to (0,0)
\texttt{N = k}
return N
\end{verbatim}
\[ k = 0 \]
\[ xc = 1 \]
\[ yc = 0 \]
\[ d = \sqrt{xc^2 + yc^2} \]

while \( d \leq r \):

\[ k = k + 1 \]
\[ \theta = \left( \frac{k \times t}{180} \right) \pi \]
\[ xc = xc + (k+1) \times \cos(\theta) \]
\[ yc = yc + (k+1) \times \sin(\theta) \]
\[ d = \sqrt{xc^2 + yc^2} \]

return \( k - 1 \)
4. Streaks in a Coin Toss Sequence
Coin Toss Strings

$S$ is a coin toss string if it is made up of $H$'s and $T$'

$$s = \text{`HHTHTTTTHHTHTTTT'}$$
Streaks

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

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

\[ s = 'HHTHTTTHTHTHTHTTTT' \]

\[ s[4:7] \] a length-3 streak
Streaks

\[
\begin{align*}
&\text{s} = 'HHTHTTTTHTHHTTTTT' \\
&\text{s}[12:17] \quad \text{a length-5 streak}
\end{align*}
\]
Streak Definition

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

(1) \( k+n \leq \text{len}(s) \)

and

(2) It is either all T’s or all H’s

and

(3) If there is a character before \( s[k] \), it is different from \( s[k] \).

and

(4) If there is a character after \( s[k+n] \), it is different from \( s[k+n] \).
Streaks

\[ s = 'HHTHTTTTHHTHTHTTTT' \]

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

Rule 3: If there is a character before \( s[k] \), it is different from \( s[k] \).
def isStreak(s, k, n):
    t = s[k:k+n]
    if k+n > len(s):
        return False
    elif t.count('H') < n and t.count('T') < n:
        return False
    elif k > 0 and (s[k-1] == s[k]):
        return False
    elif (k+n < len(s)) and (s[k+n-1] == s[k+n]):
        return False
    else:
        return True

A function can have more than one return
Using `isStreak` to Find Streaks

\[ s = 'HHTHTTTTHHHTHTHTTT' \]

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

-------------------------------------------
0                   False
Using `isStreak` to Find Streaks

\[ s = 'HHTHTTTTHTHHHTHTTT' \]

<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>
</tbody>
</table>
Using `isStreak` to Find Streaks

\[
s = 'HHTHTTTTHHHTHTTT'
\]

<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>
</tbody>
</table>
Using `isStreak` to Find Streaks

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

<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

\[
s = 'HHTHTTTTHTHHHTHTTT'
\]

<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

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