### 10. Iteration: The while-Loop

#### Topics:
- Open-Ended repetition
- the while statement
- Random Walk Simulation

---

### 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.

**Examples**

Keep tossing a coin until the number of heads and the number of tails differs by 10.

Compute the square root of 2...

L = 2; W = 1
Repeat this until \(|L - W| \leq 0.00001:
L = (L + W)/2
W = x/L

---

### The Random Walk Idea

Starting at the center tile, a robot hops from tile to tile according to a coin flip.
Heads: Hop right one tile.
Tails: Hop left one tile.
The simulation over when robot reaches either end (a.k.a. the boundary) of the runway.

### The While Loop

We introduce an alternative to the for-loop called the while-loop.
The while loop is more flexible and is essential for "open ended" iteration.
How Does a While-Loop Work?

A simple warm-up example:
Sum the first 5 whole numbers and display the summation process.

Two Solutions

```python
k = 0
s = 0
while k < 5:
    k = k + 1
    s = s + k
    print k, s
```

```python
s = 0
for k in range(1,6):
    s = s + k
    print k, s
```

The While-Loop Solution

```python
k = 0
s = 0
while k < 5:
    k = k + 1
    s = s + k
    print k, s
```

The Solution

```python
k = 0
s = 0
while k < 5:
    k = k + 1
    s = s + k
    print k, s
```

Observation: \( k \) is used for counting, \( s \) is used for the running sum, and the \( \text{while} \) is used to control the repetition of the indented code.

We call this the "loop body".

Trace the Execution

At the start, \( k \) and \( s \) are initialized

Is the boolean condition true?
Trace the Execution

```
k = 0
s = 0
while k < 5:
    k = k + 1
    s = s + k
    print k, s
```

Yes, so execute the loop body

```
k = 0
s = 0
while k < 5:
    k = k + 1
    s = s + k
    print k, s
```

Is the boolean condition true?

```
k = 0
s = 0
while k < 5:
    k = k + 1
    s = s + k
    print k, s
```

Yes, so execute the loop body

```
k = 0
s = 0
while k < 5:
    k = k + 1
    s = s + k
    print k, s
```

Is the boolean condition true?
Trace the Execution

```python
k = 0
s = 0
while k < 5:
    k = k + 1
    s = s + k
    print k, s
```

Trace the Execution

```python
k = 0
s = 0
while k < 5:
    k = k + 1
    s = s + k
    print k, s
```

Yes, so execute the loop body

Trace the Execution

```python
k = 0
s = 0
while k < 5:
    k = k + 1
    s = s + k
    print k, s
```

Is the boolean condition true?

Trace the Execution

```python
k = 0
s = 0
while k < 5:
    k = k + 1
    s = s + k
    print k, s
```

Yes, so execute the loop body

Trace the Execution

```python
k = 0
s = 0
while k < 5:
    k = k + 1
    s = s + k
    print k, s
```

Is the boolean condition true?
Trace the Execution

```
k = 0
s = 0
while k < 5:
    k = k + 1
    s = s + k
    print k, s
```

Yes, so execute the loop body

```
k = 0
s = 0
while k < 5:
    k = k + 1
    s = s + k
    print k, s
```

4

k - 1 

s - 10

Trace the Execution

```
k = 0
s = 0
while k < 5:
    k = k + 1
    s = s + k
    print k, s
```

5

k - 1

s - 15

The While-Loop Mechanism

```
while A Boolean Expression :
    The Loop Body
```

The Boolean expression is checked. If it is true, then the loop body is executed. The process is repeated until the Boolean expression is false. At that point the iteration terminates.

The Broader Context

```
Code that comes before the loop
while A Boolean Expression :
    The Loop Body
Code that comes after the loop
```

Every variable involved in the Boolean expression must be initialized.

```
Code that comes before the loop
while A Boolean Expression :
    The Loop Body
Code that comes after the loop
```

After the loop terminates the next statement after the loop is executed.
The Broader Context

while A Boolean Expression:

The Loop Body

Code that comes after the loop

Back to Our Example

k = 0
s = 0
while k < 5:
k = k + 1
s = s + k
print k,s

Random Walks

A very important type of random simulation.
A good example to showcase the while loop.

The Random Walk Idea
We have a "runway" made up of 1x1 tiles.
There are 2L+1 tiles. (L = 5 in the above.)
We call L the "length of the runway.
The center tile is located at x = 0.

The Random Walk Idea
Starting at the center tile, a robot hops from tile to tile according to a coin flip.
Heads: Hop right one tile.
Tails: Hop left one tile.
The simulation over when robot reaches either end (a.k.a. the boundary) of the runway.
The Random Walk Idea

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

Question:
Given the runway length L, what is the average number of hops required for the robot to reach the boundary?

Implement ShowRandomWalk.py

from random import randint as randi

def RandomWalk(L):
    # Returns the number of hops for a single random walk.

def AveRandomWalk(L,n):
    # Simulate n length-L random walks and returns average number of required hops

if __name__ == '__main__':
    # Display the value of AveRandomWalk for various values of L

The Function RandomWalk(L)

def RandomWalk(L):
    hops = 0; x = 0
    while abs(x) < L:
        r = randi(0,1)
        if r == 0:
            x = x + 1
        else:
            x = x - 1
        hops += 1
    return hops

Initializations. The robot starts at x=0.

The Function RandomWalk(L)

def RandomWalk(L):
    hops = 0; x = 0
    while abs(x) < L:
        r = randi(0,1)
        if r == 0:
            x = x + 1
        else:
            x = x - 1
        hops += 1
    return hops

If the condition is True, the robot has not yet reached the boundary and we keep iterating.

The Function RandomWalk(L)

def RandomWalk(L):
    hops = 0; x = 0
    while abs(x) < L:
        r = randi(0,1)
        if r == 0:
            x = x + 1
        else:
            x = x - 1
        hops += 1
    return hops

We simulate the coin toss by picking 0 or 1 at random.

The Function RandomWalk(L)

def RandomWalk(L):
    hops = 0; x = 0
    while abs(x) < L:
        r = randi(0,1)
        if r == 0:
            x = x + 1
        else:
            x = x - 1
        hops += 1
    return hops

Hop right

Hop left
The While Loop
To more fully understand how this works, let's look at the execution of this while loop:

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

To more fully understand how this works, let's look at the execution of this while loop:

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

Assume r = 0
Coin = Heads
Hop Right

The value of x is increased from 0 to 1.

Assume r = 1
Coin = Tails
Hop Left

abs(x)< 5 is true.
Robot not at boundary.
Loop continues.

The absolute value of x is less than 5.
Robot not at boundary.
Loop continues.

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

The robot moves to the right.

The robot moves to the left.

-5 -4 -3 -2 -1 0 1 2 3 4 5
Understanding the While Loop

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

The value of $x$ is decreased from 1 to 0.

abs($x$) < 5 is true. Robot not at boundary. Loop continues

Assume $r = 0$
Coin = Heads Hop Right

The value of $x$ is increased from 0 to 1.

abs($x$) < 5 is true. Robot not at boundary. Loop continues

Assume $r = 0$
Coin = Heads Hop Right

The value of $x$ is increased from 0 to 1.
Understanding the While Loop

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

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

The value of x is increased from 1 to 2.

Understanding the While Loop

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

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

abs(x) < 5 is true.
Robot not at boundary.
Loop continues

Understanding the While Loop

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

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

Assume r = 0
Coin = Heads
Hop Right

The value of x is increased from 2 to 3.

Understanding the While Loop

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

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

Assume r = 1
Coin = Tails
Hop Left

Understanding the While Loop

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

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

abs(x) < 5 is true.
Robot not at boundary.
Loop continues

Understanding the While Loop

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

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

The value of x is increased from 1 to 2.

Understanding the While Loop

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

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

The value of x is increased from 2 to 3.
x = 0
while abs(x) < 5:
    r = randi(0,1)
    if r == 0:
        x = x+1
    else:
        x = x-1

The value of x is decreased from 3 to 2.

abs(x)< 5 is true.
Robot not at boundary.
Loop continues

Assume r = 1
Coin = Heads
Hop Right

The value of x is increased from 2 to 3.

abs(x)< 5 is true.
Robot not at boundary.
Loop continues

Assume r = 0
Coin = Heads
Hop Right
Understanding the While Loop

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

The value of x is increased from 3 to 4.

Assume r = 0
Coin = Heads
Hop Right

abs(x) < 5 is False.
Robot is on the boundary.
Loop TERMINATES

abs(x) < 5 is True.
Robot not at boundary.
Loop continues

The Application Script

Check out the cases L = 5, 10, 15, 20, 25, 30, 35, 40:

```python
if __name__ == '__main__':
    n = 1000  # Number of trials
    for L in range(5, 45, 5):
        print L,
        AveRandomWalk(L, n)
```
The Function
AveRandomWalk(L,n)

def AveRandomWalk(L,n):
    s = 0
    for k in range(0,n):
        RequiredHops = RandomWalk(L)
        s += RequiredHops
        ave = float(s)/float(n)
    return ave

Sample Output

<table>
<thead>
<tr>
<th>L</th>
<th>Ave</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>24</td>
</tr>
<tr>
<td>10</td>
<td>93</td>
</tr>
<tr>
<td>15</td>
<td>219</td>
</tr>
<tr>
<td>20</td>
<td>399</td>
</tr>
<tr>
<td>25</td>
<td>649</td>
</tr>
<tr>
<td>30</td>
<td>917</td>
</tr>
<tr>
<td>35</td>
<td>1259</td>
</tr>
<tr>
<td>40</td>
<td>1594</td>
</tr>
</tbody>
</table>

Looks like doubling L increases the average by a factor of 4.

Insight through Computing!