8. More on Iteration with For

Topics:
- Using for with range
- Summation
- Computing Min's
- Functions and for-loops
- Graphical iteration

Traversing a String Character-by-Character

```python
s = 'abcd'
for c in s:
    print c
```

Output:
```
a
b
c
d```

In this example, the "for-loop" variable is `c`. One at a time, it takes on the value of each character in `s`.

for-loop Mechanics

```
for <loop variable> in <string>:
    Loop Body
```

If the string has length `n`, then the loop body is executed `n` times.

Note the Similarities

```
n = 4
for k in range(n):
    print k
```

Output:
```
0
1
2
3```

```
s = 'abcd'
for c in s:
    print c
```

Output:
```
a
b
c
d```

More General Iteration with For-Loops

```
n = 4
for k in range(n):
    print k
```

Output:
```
0
1
2
3```

How does this work? What does range(n) mean?
Summation is a Good Example

\[
n = 4 \\
s = 0 \\
\text{for } k \text{ in range}(n): \\
x = 2^k \\
s = s + x \\
\text{print } s \\
\]

Output: 15

1 + 2 + 4 + 8 = 15

for-loop Mechanics with range

\[
\text{for } k \text{ in range}(4): \\
\]

Output: 15

Let \( k = 0 \) and then execute the loop body.
Let \( k = 1 \) and then execute the loop body.
Let \( k = 2 \) and then execute the loop body.
Let \( k = 3 \) and then execute the loop body.

k is called the loop variable a.k.a. the count variable

Let's Derive this Code. It's about adding up powers of two

Summation: How Do We Do It?

Let's add up powers of 2...

\[
1 = 1 \\
3 = 1 + 2 \\
7 = 1 + 2 + 4 \\
15 = 1 + 2 + 4 + 8 \\
\]

And so on

Do we "start from scratch" each time we generate a new sum?

Summation

Let's add up powers of 2...

\[
1 = 1 \\
3 = 1 + 2 \\
7 = 1 + 2 + 4 \\
15 = 1 + 2 + 4 + 8 \\
\]

And so on

1 + 2 + 4 + 8 = 15

Nope! We keep a "running sum" into which we add powers of 2

Summation

\[
s = 0 \\
x = 2**0 \\
s = s+x \\
x = 2**1 \\
s = s+x \\
x = 2**2 \\
s = s+x \\
x = 2**3 \\
s = s+x \\
\]

1 = 0 + 1
3 = 1 + 2
7 = 3 + 4
15 = 7 + 8

And so on

And so on
**Summation**

\[
\begin{align*}
s &= 0 \\
x &= 2^0 \\
s &= s + x \\
x &= 2^1 \\
s &= s + x \\
x &= 2^2 \\
s &= s + x \\
x &= 2^3 \\
s &= s + x
\end{align*}
\]

\[
\begin{align*}
s &= 0 + 1 \\
s &= 1 + 2 \\
s &= 3 + 4 \\
s &= 7 + 8 \\
s &= 15 + 16
\end{align*}
\]

Note the pattern

Let's step through the mechanics of this for-loop:

\[
\begin{align*}
s &= 0 \\
\text{for } k \text{ in range(4):} \\
&\quad x = 2^k \\
&\quad s = s + x \\
&\quad \text{print } s
\end{align*}
\]

1 + 2 + 4 + 8

\[
\begin{align*}
s &= 0 \\
\text{for } k \text{ in range(4):} \\
&\quad x = 2^k \\
&\quad s = s + x \\
&\quad \text{print } s
\end{align*}
\]

Initialize the running sum \( s \).

We enter the loop. The loop variable \( k \) is set to zero.

\[
\begin{align*}
s &= 0 \\
\text{for } k \text{ in range(4):} \\
&\quad x = 2^k \\
&\quad s = s + x \\
&\quad \text{print } s
\end{align*}
\]

\[
\begin{align*}
s &= 0 \\
\text{for } k \text{ in range(4):} \\
&\quad x = 2^k \\
&\quad s = s + x \\
&\quad \text{print } s
\end{align*}
\]

\[
\begin{align*}
s &= 0 \\
\text{for } k \text{ in range(4):} \\
&\quad x = 2^k \\
&\quad s = s + x \\
&\quad \text{print } s
\end{align*}
\]

k<4 is true so we execute the loop body with that value of \( k \).
s = 0
for k in range(4):
    x = 2**k
    s = s + x
print s

k is increased by 1

s = 0
for k in range(4):
    x = 2**k
    s = s + x
print s

k<4 is true so we execute the loop body with that value of k.

s = 0
for k in range(4):
    x = 2**k
    s = s + x
print s

k is increased by 1

s = 0
for k in range(4):
    x = 2**k
    s = s + x
print s

k<4 is true so we execute the loop body with that value of k.

s = 0
for k in range(4):
    x = 2**k
    s = s + x
print s

k is increased by 1
1 + 2 + 4 + 8

\begin{align*}
\text{s} &= 0 \\
\text{for } k \text{ in range(4)}: \\
\quad x &= 2^k \\
\quad s &= s + x \\
\text{print } s
\end{align*}

\begin{itemize}
\item $k < 4$ is true so we execute the loop body with that value of $k$.
\end{itemize}

\begin{align*}
\text{s} &= 0 \\
\text{for } k \text{ in range(4)}: \\
\quad x &= 2^k \\
\quad s &= s + x \\
\text{print } s
\end{align*}

\begin{itemize}
\item $k$ is increased by 1.
\end{itemize}

\begin{align*}
\text{s} &= 0 \\
\text{for } k \text{ in range(4)}: \\
\quad x &= 2^k \\
\quad s &= s + x \\
\text{print } s
\end{align*}

\begin{itemize}
\item $k < 4$ is true so we execute the loop body with that value of $k$.
\end{itemize}

\begin{align*}
\text{s} &= 0 \\
\text{for } k \text{ in range(4)}: \\
\quad x &= 2^k \\
\quad s &= s + x \\
\text{print } s
\end{align*}

\begin{itemize}
\item $k$ is increased by 1.
\end{itemize}

\begin{align*}
\text{s} &= 0 \\
\text{for } k \text{ in range(4)}: \\
\quad x &= 2^k \\
\quad s &= s + x \\
\text{print } s
\end{align*}

\begin{itemize}
\item $k < 4$ is true so we execute the loop body with that value of $k$.
\end{itemize}
1 + 2 + 4 + 8

s = 0
for k in range(4):
    x = 2**k
    s = s + x
print s

k is increased by 1

1 + 2 + 4 + 8

s = 0
for k in range(4):
    x = 2**k
    s = s + x
print s

Output 15

k<4 is False so we exit the loop body and proceed with the next statement after the loop.

1 + 2 + 4 + 8

s = 0
for k in range(4):
    x = 2**k
    s = s + x
print s

More General:
1 + 2 + 4 + ... + 2**(n-1)

n = any positive integer
s = 0
for k in range(n):
    x = 2**k
    s = s+x
print s

for-loop Mechanics with range

for k in range(n):
    Loop Body

Let k = 0 and then execute the loop body.
Let k = 1 and then execute the loop body.
Let k = 2 and then execute the loop body.
... Let k = n-1 and then execute the loop body.
for-loop Mechanics with range

```python
for k in range(n):
x = 2**k
s = s+x
```

Let $k = 0$ and then execute the loop body.
Let $k = 1$ and then execute the loop body.
Let $k = 2$ and then execute the loop body.
Let $k = n-1$ and then execute the loop body.

Counting: A Special Type of Summation

How Many Integers $< 10^{6}$ are there that are divisible by 2, 3, and 5?

```python
N = 0
for k in range(10**6):
    if k%2==0 and k%3==0 and k%5==0:
        N = N+1
print N
```

Output: 33334

Using a For-Loop to Enumerate all Possibilities

“Left-Shifting” a String

```python
s = 'abcd'
n = len(s)
for k in range(n):
t = s[k:]+s[:k]
print t
```

Output:
```
abcd
bcda
cdab
dabc
```

Iteration with strings doesn't always have the form for c in s

Looking for a Minimum
Assume this Function is Available

def dist(t):
    """ Returns a float that is the distance between Earth and a rogue asteroid at time t (days).
    PreC: t is a nonnegative float."""

Which of the numbers dist(0), dist(1), dist(2),…,dist(100000) is the smallest and what is the associated t-value?

Solution

d_min = dist(0)
t_min = 0
for t in range(100001):
    d_current = dist(t)
    if d_current < d_min:
        # A new minimum is found
        d_min = d_current
        t_min = t
print t_min, d_min

More on range

In all our examples, the loop variable steps from 0 to some number.

There are other options.

"Counting from 1"

n = 4
for k in range(n):
    print k

Output: 0 1 2 3

"Counting from Here to (Almost) There"

Here = 20
There = 24
for k in range(Here,There):
    print k

Output: 20 21 22 23

"Counting from 1"

n = 4
for k in range(n):
    print k

Output: 1 2 3
"Counting Down"

Here = 20
There = 24

for k in range(There, Here, -1):
    print k

Output:
24
23
22
21

Now Let Us Look at Functions and For Loops

Recall From simpleMath

```python
def sqrt(x):
    x = float(x)
    L = x
    L = (L + x/L)/2
    L = (L + x/L)/2
    L = (L + x/L)/2
    L = (L + x/L)/2
    return L
```

Let's implement this with a for-loop

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

Another For-Loop Implementation

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

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

Sample Call:  
y = sqrt(12345, 20)

The optional argument allows you to determine the number of iterations.

Now Let Us Look at Graphics Procedures and For Loops
Recall `DrawRect`

This will draw a red square with side \( s \) and center \((x_c,y_c)\):

`DrawRect(xc,yc,s,s,RED)`

This will draw a white square with side \( s \) and center \((x_c,y_c)\):

`DrawRect(xc,yc,s,s,WHITE)`

Let’s Write a Procedure that Can Draw a Checkered Row

Assume \( n \) squares each with side \( s \).
Assume \((x_0,y_0)\) is the center of the leftmost square.
Let \( c_1 \) and \( c_2 \) be the Colors of the first and second square.

Let’s Write a Procedure that Can Draw a Checkered Row

Assume \( n \) squares each with side \( s \).
Assume \((x_0,y_0)\) is the center of the leftmost square.
Let \( c_1 \) and \( c_2 \) be the Colors of the first and second square.

This Draws a Checkered Row

```python
def DrawRow(x0,y0,s,n,c1,c2):
    # Center of next square is (xc,yc)
    xc = x0
    yc = y0
    for k in range(n):
        if k%2==0:
            DrawRect(xc,yc,s,s,color=c1)
        else:
            DrawRect(xc,yc,s,s,color=c2)
        xc = xc+s
```

Now Let's Draw This

This Draws an 8x8 Checker Board

```python
y0 = -4; x0 = -3.5; n = 8; s = 1
#(x0,y0) is the center of the leftmost # square in the next row to draw
for k in range(n):
    # Draw the kth row
    if k%2==0:
        DrawRow(x0,y0,s,n,RED,WHITE)
    else:
        DrawRow(x0,y0,s,n,WHITE,RED)
    # The next row is s units higher
    y0 = y0+s
```