9A. Iteration with `range`

Topics:

- Using `for` with `range`
- Summation
- Computing Min's
- Functions and for-loops
- A Graphics Applications

Iterating Through a String

```
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.

We learned about this in the previous lecture.

Iterating Through a Range

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

Output:

```
0
1
2
3
```

How does this work? What does range(n) mean?

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
```

Summation is a Good Example

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

Output:

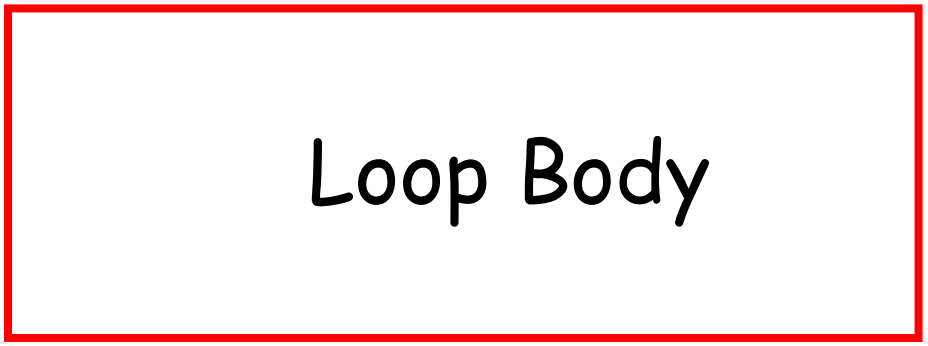
15

1 + 2 + 4 + 8 = 15

We are repeating the purple box 4 times

for-loop Mechanics with range

```
for k in range(4):
```



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

Summation

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

Output:

15

1 + 2 + 4 + 8 = 15

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

$1 = 0 + 1$

$3 = 1 + 2$

$7 = 3 + 4$

$15 = 7 + 8$

And so on

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

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

Note the pattern

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
```

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

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

1 + 2 + 4 + 8

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

s -> 0

Initialize the running sum s.

1 + 2 + 4 + 8

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

s -> 0

k -> 0

We enter the loop.

The loop variable **k** is set to zero

1 + 2 + 4 + 8

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

s -> 0

k -> 0

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

1 + 2 + 4 + 8

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

s -> 1

k -> 0

x -> 1

1 + 2 + 4 + 8

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

s -> 1

k -> 0

x -> 1

k is increased by 1

1 + 2 + 4 + 8

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

s -> 1

k -> 1

x -> 1

1 + 2 + 4 + 8

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

s -> 1

k -> 1

x -> 1

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

1 + 2 + 4 + 8

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

s -> 3

k -> 1

x -> 2

1 + 2 + 4 + 8

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

s -> 3

k -> 1

x -> 2

k is increased by 1

1 + 2 + 4 + 8

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

s -> 3

k -> 2

x -> 2

1 + 2 + 4 + 8

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

s -> 3

k -> 2

x -> 2

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

1 + 2 + 4 + 8

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

s -> 7

k -> 2

x -> 4

1 + 2 + 4 + 8

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

s -> 7

k -> 2

x -> 4

k is increased by 1

1 + 2 + 4 + 8

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

s -> 7

k -> 3

x -> 4

1 + 2 + 4 + 8

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

s -> 7

k -> 3

x -> 4

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

1 + 2 + 4 + 8

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

s -> 15

k -> 3

x -> 8

1 + 2 + 4 + 8

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

s -> 15

k -> 3

x -> 8

k is increased by 1

1 + 2 + 4 + 8

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

s -> 15

k -> 4

x -> 8

1 + 2 + 4 + 8

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

s -> 15

k -> 4

x -> 8

`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
```

s -> 15

k -> 4

x -> 8

Output

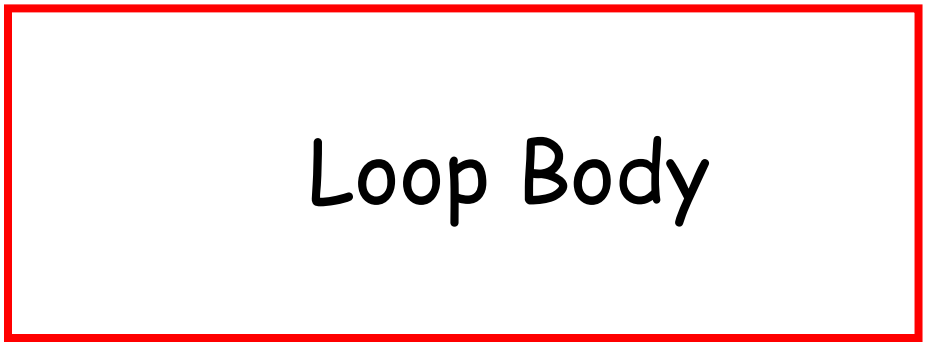
```
15
```

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):
```



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

```
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?

```
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

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

Output:

```
abcd
bcda
cdab
dabc
```

If `k==2`, then `s[2:]+s[:2]` looks like this: 'cd' + 'ab'

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."""
```

Problem: Which of the numbers

`dist(0), dist(1), dist(2),…,dist(100000)`

is the smallest and what is its 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
        # Remember the day it occurred
        t_min = t
print t_min, d_min
```

We need `range(100001)` because we want to check `dist(100000)`

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
```

```
n = 4
for k in range(1,n):
    print k
```

Output:

```
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 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

```
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
    L = (L + x/L)/2
    return L
```

Let's implement this with a for-loop

For-Loop Implementation

```
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
    L = (L + x/L)/2
    return L
```

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

Another For-Loop Implementation

```
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`

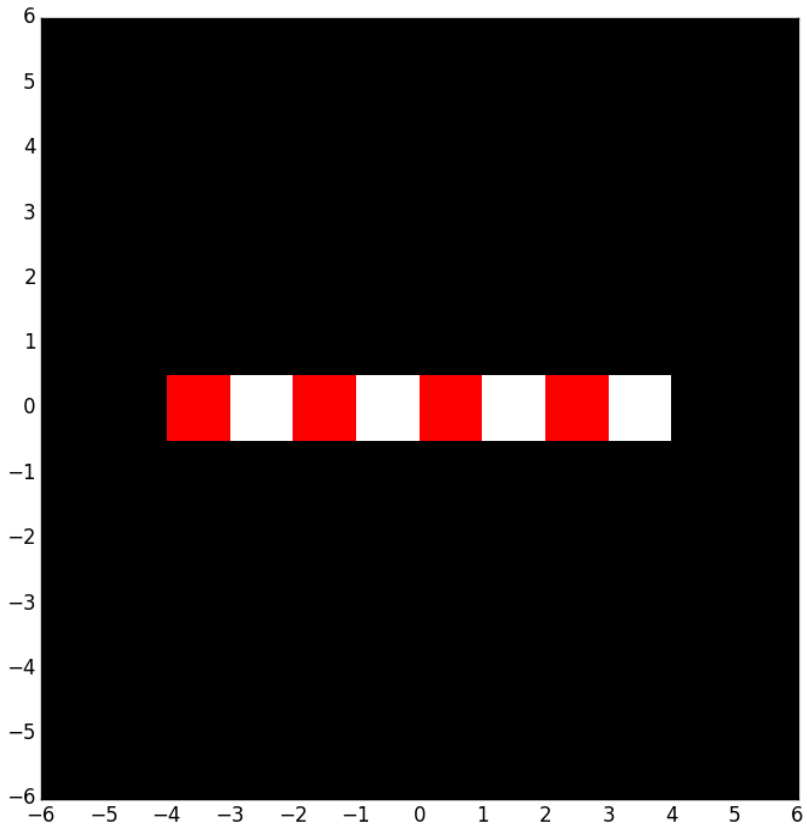
This will draw a red square with side s and center (xc,yc):

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

This will draw a white square with side s and center (xc,yc):

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

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



Assume n squares each with side s.

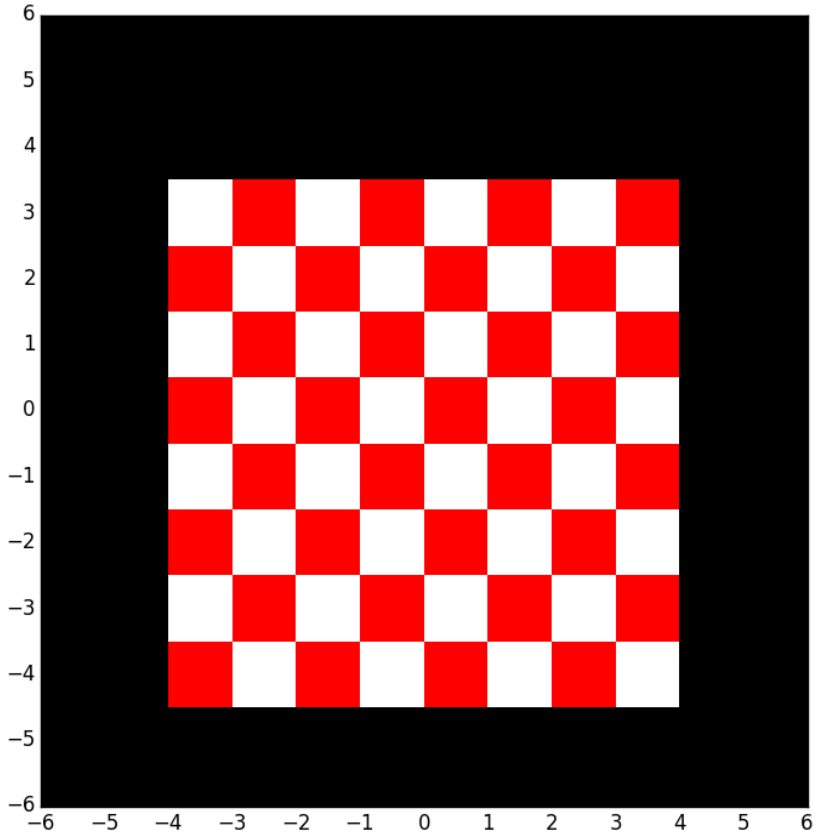
Assume (x0,y0) is the center of the leftmost square.

Let c1 and c2 be the Colors of the first and second square

Solution

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

Now Let's Draw This



This Draws an 8x8 Checker Board

```
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
```