Loops

CS 101 – Spring 2016
Department of Computer Science
Middlebury College
Factorial

00 read r1          # get # from user to r1
01 setn r2 1        # put our result into r2
02 jeqzn r1 07      # jump to line 7 if r1 == 0
03 mul r2 r2 r1     # make r2 = r2 * r1
04 addn r1 -1       # make r1 = r1 - 1
05 jumpn 02         # jump back to line 2
06 nop
07 write r2         # write out the result, r2
08 halt
Factorial

00 read r1          # get # from user to r1
01 setn r2 1        # put our result into r2
02 jeqzn r1 07      # jump to line 7 if r1 == 0
03 mul r2 r2 r1     # make r2 = r2 * r1
04 addn r1 -1       # make r1 = r1 - 1
05 jumpn 02         # jump back to line 2
06 nop
07 write r2         # write out the result, r2
08 halt

r1 = n
r2 = 1
r1 == 0

r1 = r1 - 1
r2 = r1 * r2

return r2
Factorial

00 read r1          # get # from user to r1
01 setn r2 1        # put our result into r2
02 jeqzn r1 07      # jump to line 7 if r1 == 0
03 mul r2 r2 r1     # make r2 = r2 * r1
04 addn r1 -1       # make r1 = r1 - 1
05 jumpn 02         # jump back to line 2
06 nop
07 write r2         # write out the result, r2
08 halt

r1 = n
r2 = 1
r1 == 0
r2 = r1 * r2
r1 = r1 -1
return r2

How to write this in Python?
def factorial(n):
    f = 1
    while n > 0:
        f = f * n
        n = n - 1
    return f
The **while** loop

As with conditional statements, this takes a conditional statement — one that evaluates to either *True* or *False*.

Like functions and conditionals, loops use **blocks** using colons and indentation. If the condition is *True*, all of the statements in the block will be executed. When the end of the block is reached, the conditional will be checked again. If it is still *True*, the process will repeat.

One of these statements must have the ability to change the condition, or this is an **infinite loop**.

```
while condition:
    statement 1
    statement 2
    ...
```
```python
def countEs(str):
    count = 0
    index = 0
    while index < len(str):
        current = str[index]
        if current == 'e' or current == 'E':
            count += 1
        index += 1
    return count
```
def countEs(str):
    count = 0
    for current in str:
        if current == 'e' or current == 'E':
            count += 1
    return count
The `for` loop

```
for variable in sequence:
    statement 1
    statement 2
    ...
```

This variable contains the current item from the sequence.

The loop is run once for every item in this sequence.

Yet another block-based statement.