LECTURE 6

FOR AND WHILE LOOPS

MCS 260 Fall 2021

David Dumas

REMINDERS

- No class Monday
- Homework 1 scores & Worksheet 2 solutions posted
- Homework 2 available, due next **Wed** at 10am (schedule change due to Labor day)
- Homework 2 autograder only checks syntax, and is only advisory (no points). Actual grading will be manual.

WHILE LOOPS

The syntax

```
while condition:
    statement
    statement
```

will repeatedly do the following:

1. Evaluate condition; if False, skip the rest of this list and move on. Otherwise,
2. Execute the statements in the block.
3. Return to the first step.

Called a **loop** because it returns to a previous line.

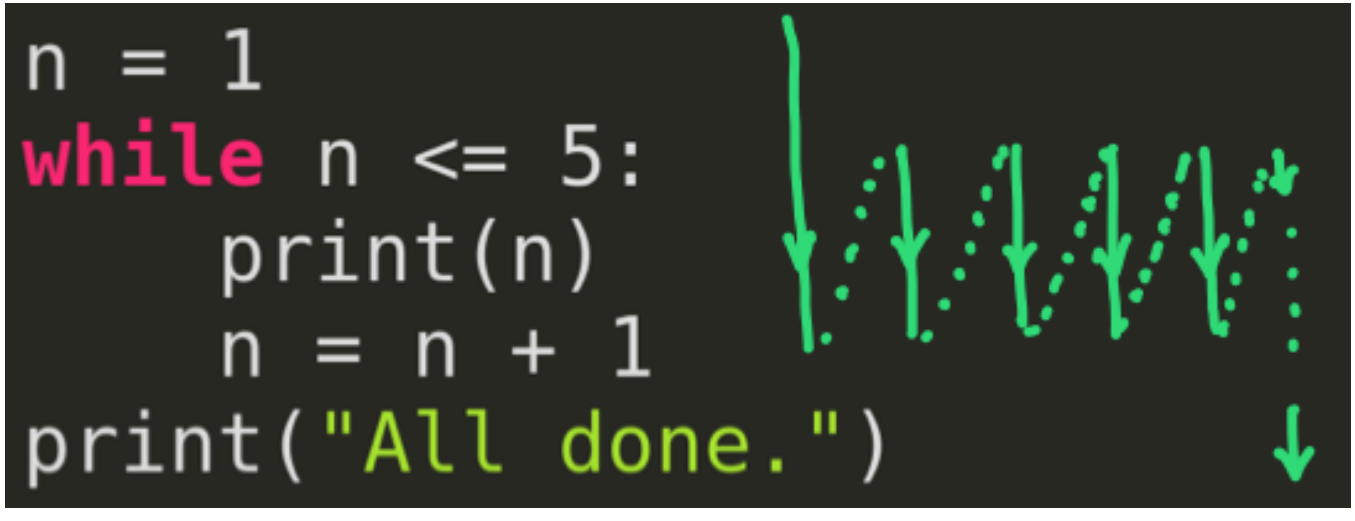
The code block following a while is called the **body** of the loop.

Most while loops will change a variable in the body, affecting the condition.

```
n = 1
while n <= 5:
    print(n)
    n = n + 1
print("All done.")
```

This prints the numbers from 1 to 5.

FLOW OF EXECUTION



INFINITE LOOPS

It's possible to write a `while` loop that will never end, e.g.

```
n = 1
while n <= 5:
    print("ESB is the best Star Wars film")
```

Such accidental **infinite loops** cause a program to appear to be stuck.

Control-C will interrupt and exit a stuck Python program.

FOR LOOPS

The syntax

```
for varname in container:
    statement
    statement
```

can be used with any sequence as `container`.

It takes an item from `container`, assigns it to the variable `varname`, runs the loop body, and then repeats the whole process until each element of container has been used exactly once.

EXAMPLE

Let's write a program that will read a string from the user and classify each character into one of the categories:

- digit (0,1,2,3,4,5,6,7,8,9)
- space ( )
- other (e.g. A-Z,a-z,...)

EXITING A LOOP

Sometimes it is helpful to exit a loop before it would end on its own, or from the middle of the body.

The **break** statement does this. When it executes, the immediate surrounding loop stops and control goes to the first statement after that loop.

```
n=1
while True:
    n = n + 1
    if n > 9:
        break
print(n)
```

ITERABLES

Besides lists and strings, some other containers are allowed in for loops.

A thing allowed in a for loop is called an *iterable*.

Some iterables generate their items one by one, rather than computing everything in advance.

RANGE

`range(N)` is an iterable that generates the integers from 0 to $N-1$.

```
for n in range(10):
    print(n+1)
```

The following is slow, as it creates a list of 50 million items:

```
L = list(range(50_000_000))
for x in L:
    # do stuff with x
    # possibly exit the loop early
```

Better way:

```
for x in range(50_000_000):
    print(x)
    if x > 10:
        break
```

This is very fast (only 12 items generated).

ENUMERATED ITERATION

What if you need the index during a for loop?

This method works, but is not recommended:

```
L = [9,8,2,4,1,1,5]
for i in range(len(L)):
    print("At index",i,"we have item",L[i])
```

Another way:

Use an extra index variable, increment it manually.

```
L = [9,8,2,4,1,1,5]
i = 0
for x in L:
    print("At index",i,"we have item",x)
    i = i + 1
```

Best way:

Use the **`enumerate()`** function. It turns a sequence like `[7,6,5]` into an enumerated sequence `[ (0,7), (1,6), (2,5) ]`.

```
L = [9,8,2,4,1,1,5]
for i,x in enumerate(L):
    print("At index",i,"we have item",x)
```

AVOID RANGE(LEN())

When you see

```
for i in range(len(L)):     # not recommended!
    # do stuff with L[i]
```

in Python code, it should usually be replaced with

```
for x in L:
    # do stuff with x
```

or

```
for i,x in enumerate(L):
    # do stuff with x and/or i
```

For and while loops allow you to write programs that process a collection of data / events / etc.

If/elif/else allow processing to be customized to the data.

Together these constructs give a lot of control over program execution.

REFERENCES

- In *Downey*:
  - Chapter 7 is devoted to a detailed discussion of loops
  - Section 8.3 and Section 10.3 contain additional examples of for loops.

REVISION HISTORY

- 2020-09-02 Initial publication