

CSI31 Lecture 18

Topics: *Chapter 8. Loop Structures and Booleans*

8.1 For loops: a quick review

8.2 Indefinite loops

8.3 Common loop patterns: interactive, sentinel

HW14 (due date is Friday, Nov.21st) :

page 262, programming problem 5

**! No class on Wednesday, November 19th
cancelled**

HW13 Due date is extended to Monday, November 17th

8.1 For loops: a quick review

We studied definite loops on lecture 5.

A Python **for** loop has this general form:

```
for <var> in <sequence>:  
    <body>
```

<body> is any sequence of Python statements

<var> is the *loop index*,

takes on each successive value in the **sequence**, and

the statements in the **body** are executed once for each value.

sequence portion often consists of a *list* of values.

example:

```
y = 1  
for counter in [1,2,3,4]:  
    y = y + counter  
    print 'counter = ', counter, ', y = ', y
```

example:

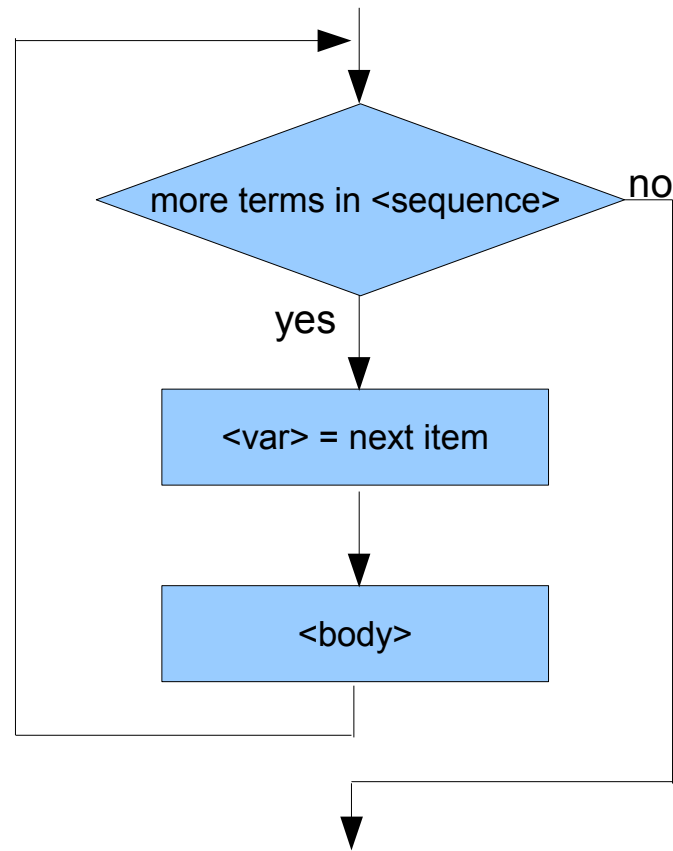
```
for i in range(10):  
    x = 3.9 * x * (1-x)  
    print x
```

If you begin to type in **range(** in the interactive window - you'll see a hint:

```
range([start,] stop[, step]) -> list of integers
```

8.1 For loops: a quick review

Flowchart of the for loop:



8.2 Indefinite loops

Assume that we want to write the program that finds the average of inputted numbers. And I don't know in advance how many numbers will be inputted.

Can we use a for loop here?

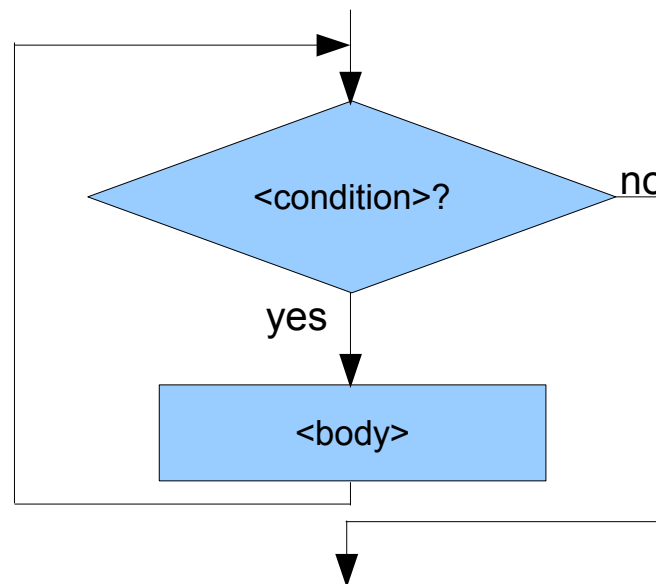
Let's take a look at **indefinite** (conditional) loop:

```
while <condition>:  
    <body>
```

<condition> is a boolean expression (just like in if statements).

The body of the loop executes repeatedly as long as the condition remains true.

Flowchart of the while loop:



8.2 Indefinite loops

example: a program that counts from 0 to 9

```
i=0
while i<=9:
    print i
    i=i+1
```

the for loop would look:

```
for i in range(10):
    print i
```

Now, if we slightly change our original program:

```
i=0
while i<=9:
    print i
    i=i+1
```

- we get an infinite loop

Infinite loops are bad thing.

Two good uses of infinite loops:

- [interactive loops](#)

at each iteration of the loop body ask the user if he/she wants input more data values or it is enough

- [sentinel loop](#)

loop continues to process data until reaching a special value that signals the end (that value is called [sentinel](#))

Let's take a look at two algorithms that employ those uses.

```
sum=0
counter=0
answer='yes'
while answer is 'yes'
    get the next_value
    counter = counter + 1
    sum=sum+next_value
    ask user if he/she wants to continue
average=sum/counter
```

See [average_i.py](#)

```
sum=0
counter=0
next_value=0
while next_value is not -1000
    counter = counter + 1
    sum=sum+next_value
    get the next_value
average=sum/(counter-1)
See average\_s.py, average\_s\_mod.py
```

! we need to be very careful when we use this approach.

Things to check: see next slide

In the second approach (as well as in the first one) it is necessary to pay attention to the following details:

- make sure that the loop runs exactly as many times as it is correct
(not more and not less)
- check what happens if there will be 0 loop iterations
- check what happens if there will be 1 loop iterations
- check what happens if there will be maximum possible for this problem number of loop iterations
- make sure that the loop terminates
- test on different inputs