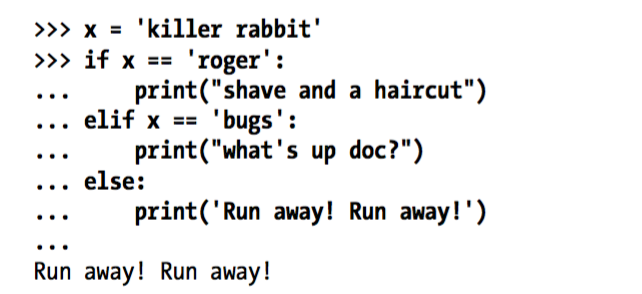
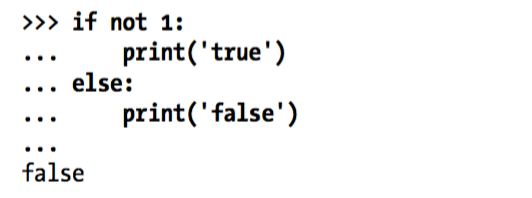
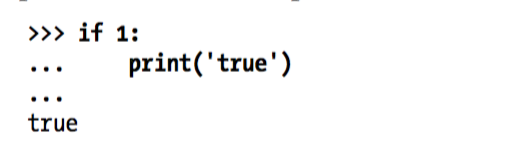
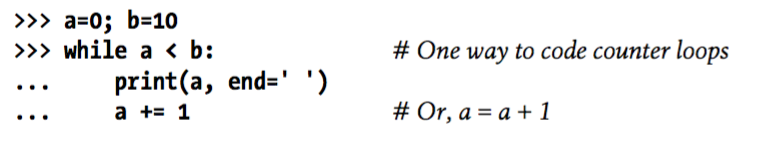
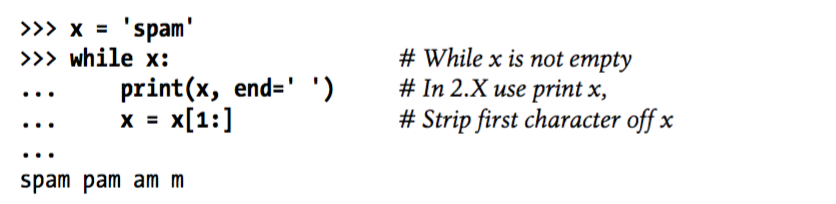
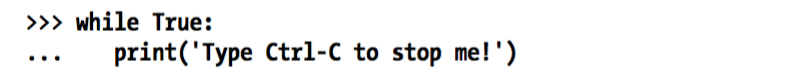
**IF**

**--**

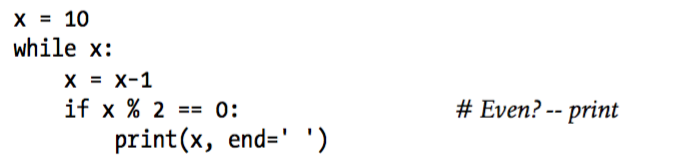
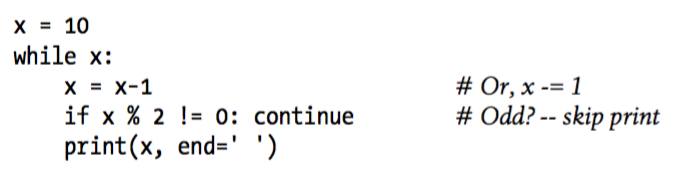


**WHILE**

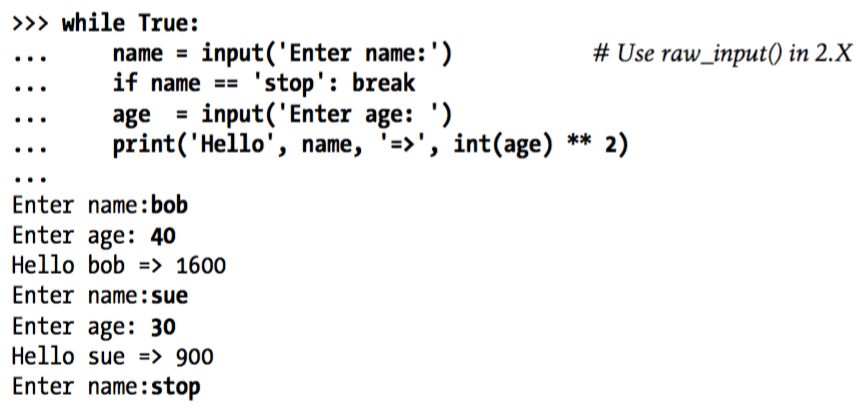
**---------**



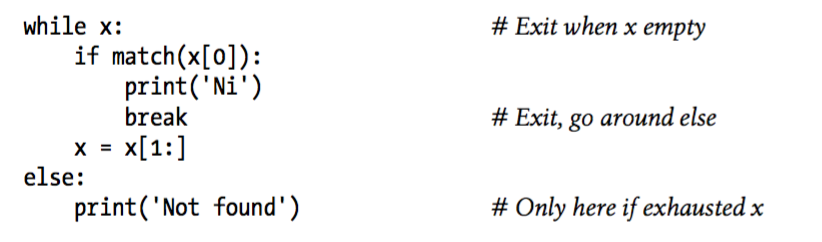
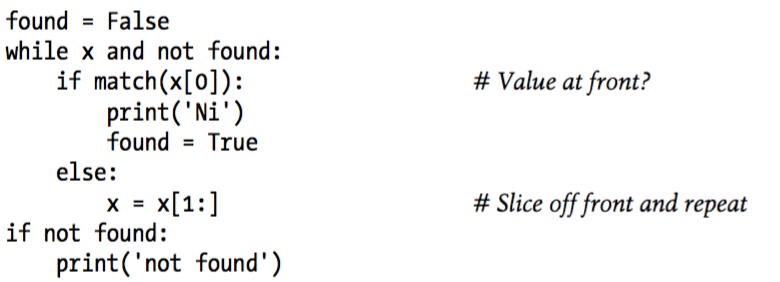
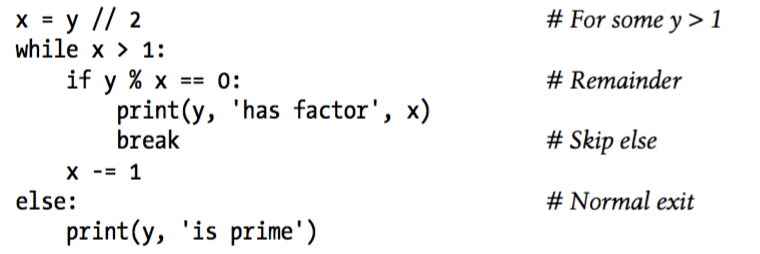






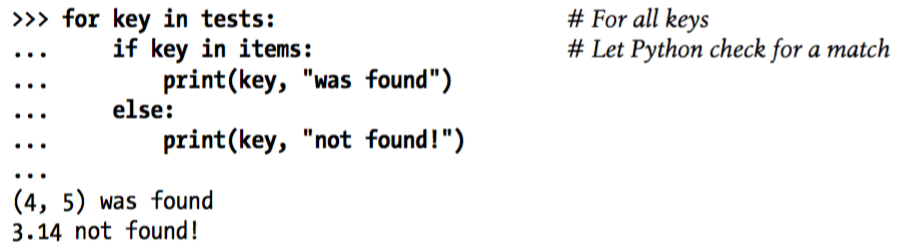
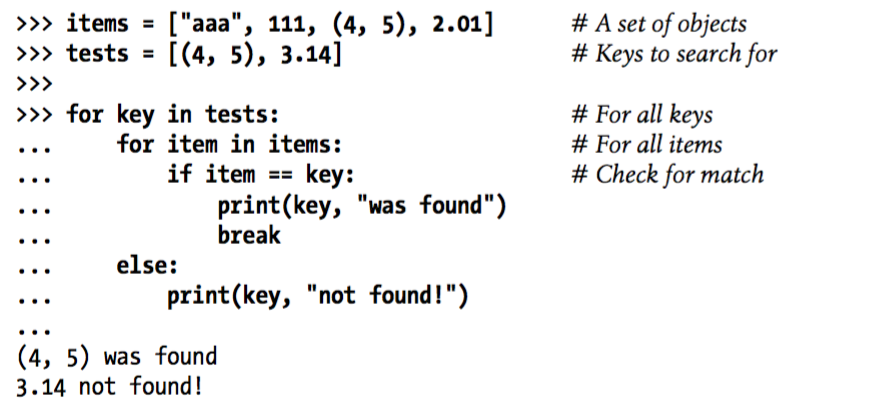
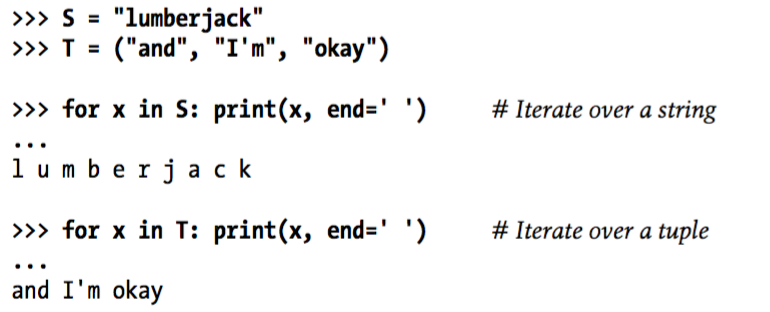
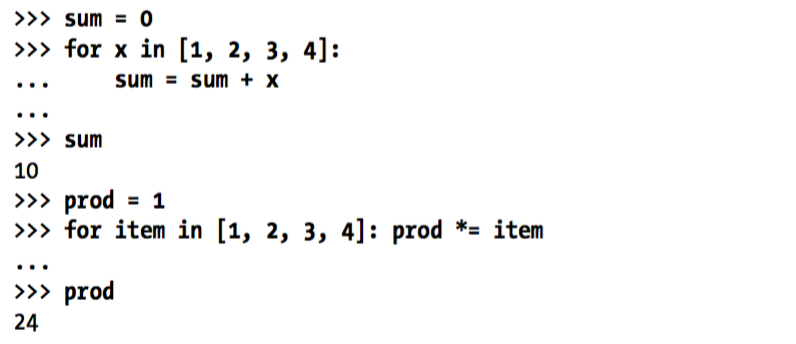
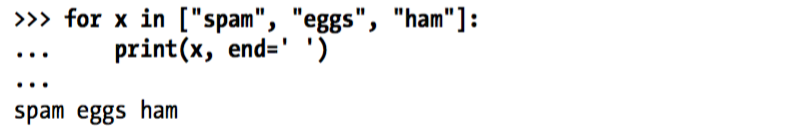


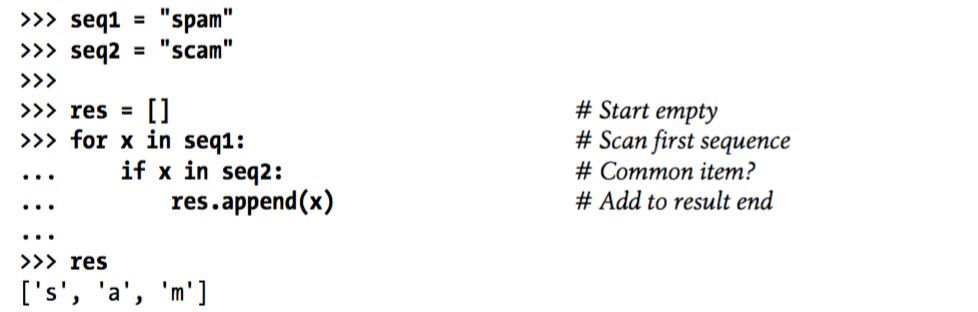




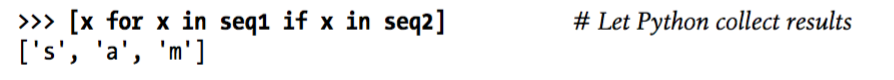
**FOR**

**------**

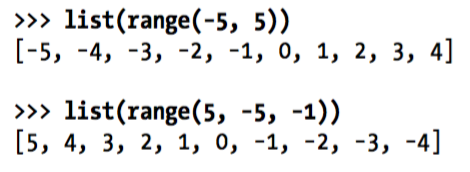
****

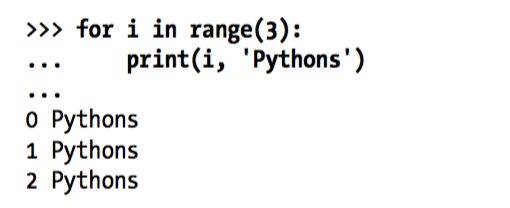
****

**List Comprehension -** collecting a results list with an iteration and optional filter test

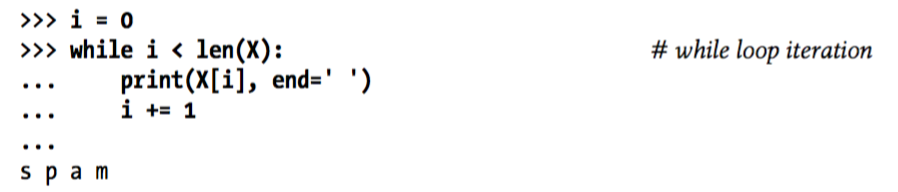
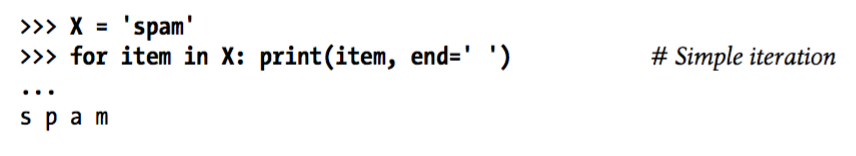


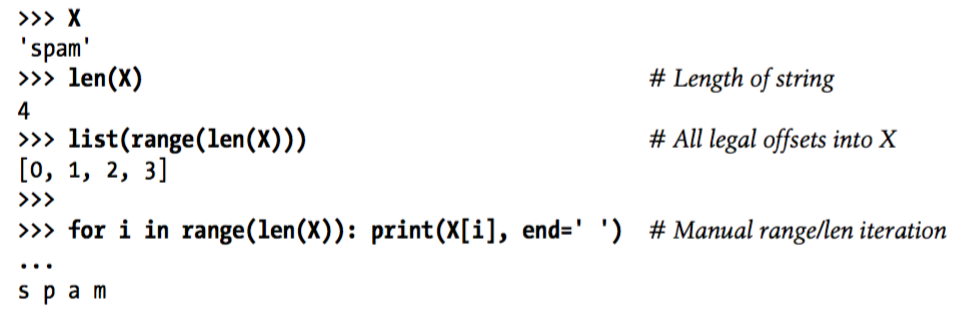






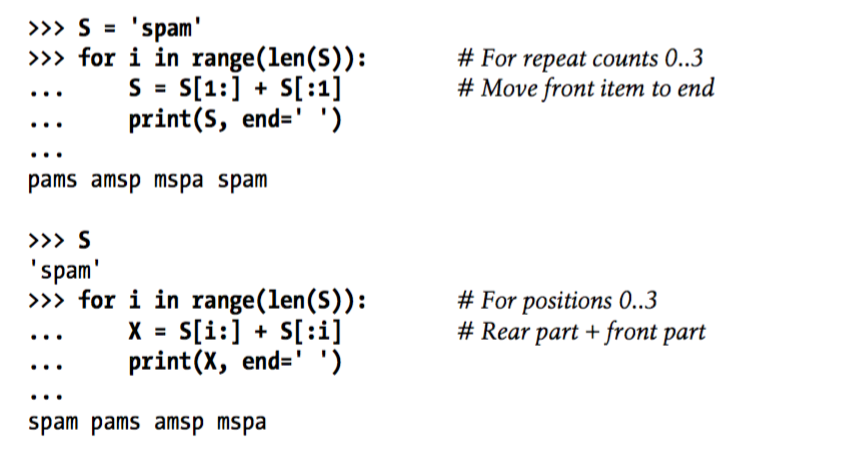
**The easiest and generally fastest way to step through a sequence exhaustively is always with a simple for, as Python handles most of the details for you.**





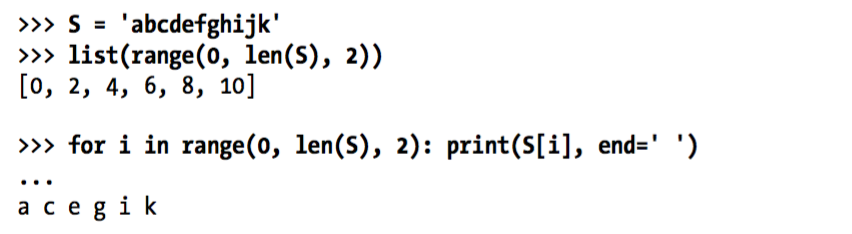
**As a general rule, use for instead of while whenever possible, and don’t use range calls in for loops except as a last resort.**

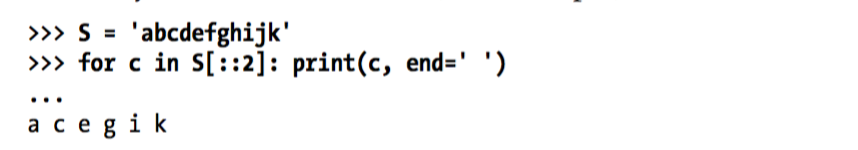
**Sometimes we want to manipulate indices.**

****

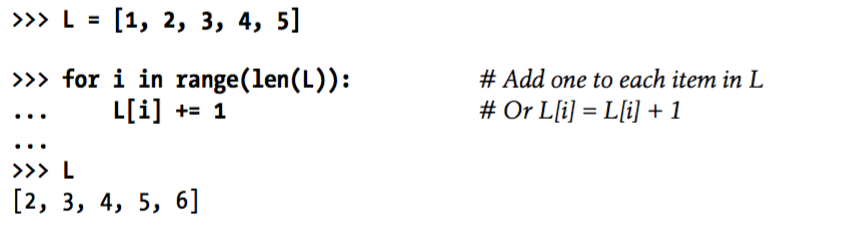
**Sometimes our searches are non exhaustive.**

**We can use range steps or slicing steps. Slicing is preferred.**

****

****

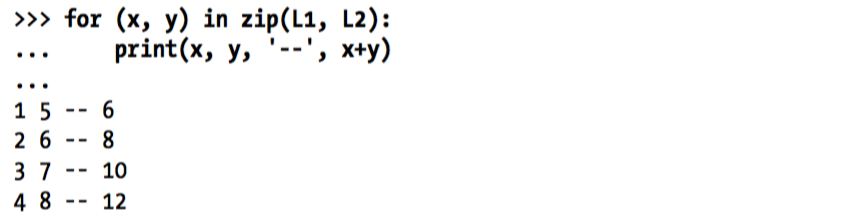
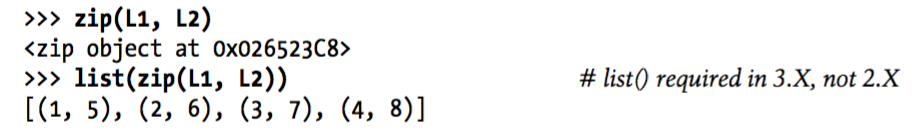
**Changing Lists.**

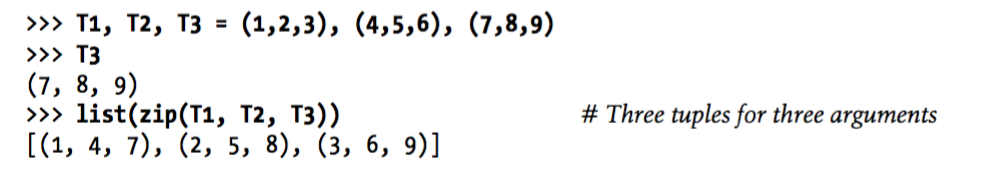
****

**Range works but list comprehensions are better.**

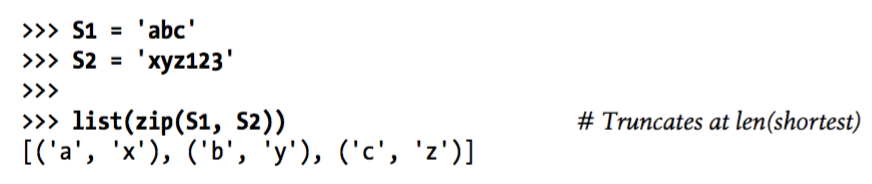
****

**Zip**

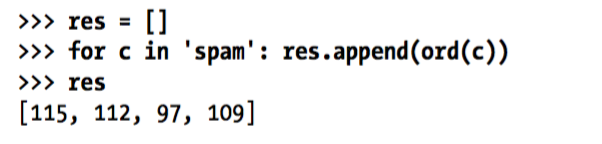
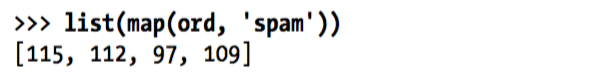
****

****

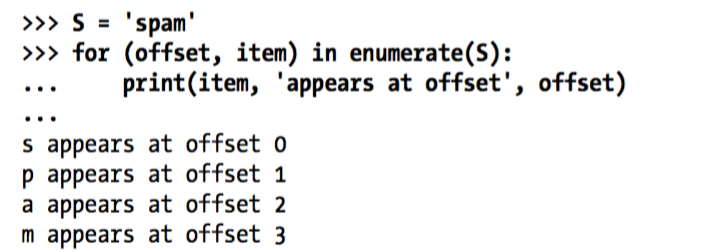
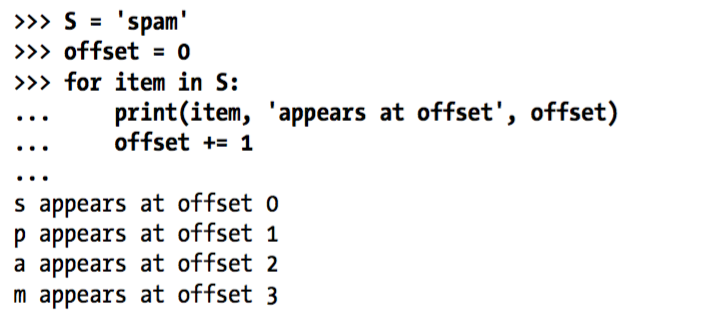
**zip truncates results tuples at length of the shortest sequence when the argument lengths differ.**

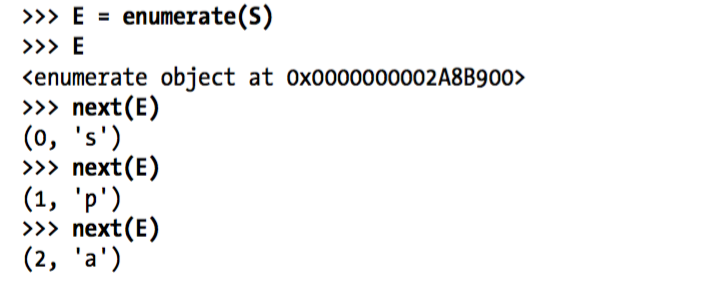
****

**Map**

****

**Enumerate**

****

**The enumerate function returns a generator object—a kind of object that supports the iteration protocol. It has a method called by the next built-in function, which returns an (*index*, *value*) tuple each time through the loop. The for steps through these tuples automatically.**