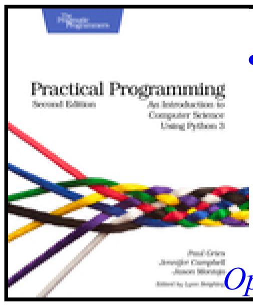
while loops



- Chapter 9.6 Looping Until a Condition Is Reached

Optional reading

The general form of a `while` loop:

```
while expression:
    statements
```

The `while` condition, `num < 100`, is evaluated, and if it is `True` the statements in the loop body are executed. The loop condition is rechecked and if found to be `True`, the body executes again. This continues until the loop condition is checked and is `False`. For example:

```
>>> num = 2
>>> while num < 100:
        num = num * 2
        print(num)


4
8
16
32
64
128
```

In the example above, there are 6 *iterations*: the loop body executes 6 times.

Loops Conditions and Lazy Evaluation

The problem: print the characters of `str s`, up to the first vowel in `s`.

The first attempt at solving this problem works nicely when `s` contains one or more vowel, but results in an error if there are no vowels in `s`:

```
>>> i = 0
>>> s = 'xyz'
>>> while not (s[i] in 'aeiouAEIOU'):
        print(s[i])
        i = i + 1


x
y
z
Traceback (most recent call last):
  File "<pyshell#73>", line 1, in <module>
    while not (s[i] in 'aeiouAEIOU'):
IndexError: string index out of range
```

In the code above, the error occurs when `s` is indexed at `i` and `i` is outside of the range of valid indices. To prevent this error, add an additional condition is added to ensure that `i` is within the range of valid indices for `s`:

```
>>> i = 0
>>> s = 'xyz'
>>> while i < len(s) and not (s[i] in 'aeiouAEIOU'):
        print(s[i])
        i = i + 1
```

```
x
y
z
```

Because Python evaluates the `and` using lazy evaluation, if the first operand is `False`, then the expression evaluates to `False` and the second operand is not even evaluated. That prevents the `IndexError` from occurring.

Jennifer Campbell • Paul Gries
University of Toronto