For Statement

Python provides a more convenient way to express a definite loop. The **for** statement iterates over a range of values. These values can be a numeric range, or, as we shall, elements of a data structure like a string, list, or tuple. The above while loop can be rewritten

```
for n in range(1, 11):
   print(n)
```

The expression **range(1, 11)** creates an object known as an iterable that allows the for loop to assign to the variable n the values 1, 2, ..., 10. During the first iteration of the loop, n's value is 1 within the block. In the loop's second iteration, n has the value of 2. The general form of the range function call is

range(begin,end,step)

where:
- **begin** is the first value in the range; if omitted, the default value is 0
- **end** is one past the last value in the range; the end value may not be omitted
- **step** is the amount to increment or decrement; if the **step** parameter is omitted, it defaults to 1 (counts up by ones)

begin, **end**, and **step** must all be integer values; floating-point values and other types are not allowed. The range function is very flexible. Consider the following loop that counts down from 21 to 3 by threes:

```
for n in range(21, 0, -3):
   print(n, '', end='')
```

It prints:

```
21 18 15 12 9 6 3
```

Thus range(21, 0, -3) represents the sequence 21,18,15,12,9,3. The expression range(1000) produces the sequence 0,1,2,...,999.

Assignment

Write code that computes and prints the sum of all the positive integers less than 100.