Iteration Example
The Fibonacci Sequence

The next Fibonacci number is the sum of the current one and its predecessor.
Is this alternative definition of \texttt{fib} the same or different from the original \texttt{fib}?

\begin{verbatim}
def fib(n):
    """Compute the n-th Fibonacci number."""
    pred, curr = 0, 1
    k = 1
    while k < n:
        pred, curr = curr, pred + curr
        k = k + 1
    return curr
\end{verbatim}

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987
Designing Functions
### Describing Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>def square(x):</code></td>
<td>&quot;&quot;&quot;Return X * X.&quot;&quot;&quot;&quot;</td>
</tr>
<tr>
<td></td>
<td><strong>Domain:</strong> All real numbers.</td>
</tr>
<tr>
<td></td>
<td><strong>Range:</strong> Non-negative real numbers.</td>
</tr>
<tr>
<td></td>
<td><strong>Behavior:</strong> Return value is the square of the input.</td>
</tr>
<tr>
<td><code>def fib(n):</code></td>
<td>&quot;&quot;&quot;Compute the nth Fibonacci number, for N &gt;= 1.&quot;&quot;&quot;&quot;</td>
</tr>
<tr>
<td></td>
<td><strong>Domain:</strong> All integers greater than or equal to 1.</td>
</tr>
<tr>
<td></td>
<td><strong>Range:</strong> All Fibonacci numbers.</td>
</tr>
<tr>
<td></td>
<td><strong>Behavior:</strong> Return value is the nth Fibonacci number.</td>
</tr>
</tbody>
</table>
A Guide to Designing Function

Give each function exactly one job, but make it apply to many related situations

```python
>>> round(1.23)  
1

>>> round(1.23, 1)
1.2

>>> round(1.23, 0)
1

>>> round(1.23, 5)
1.23
```

Don’t repeat yourself (DRY). Implement a process just once, but execute it many times.
Generalization
Generalizing Patterns with Arguments

Regular geometric shapes relate length and area.

Shape:

Area:

Finding common structure allows for shared implementation

(Demo)
Higher-Order Functions
Generalizing Over Computational Processes

The common structure among functions may be a computational process, rather than a number.

\[ \sum_{k=1}^{5} k = 1 + 2 + 3 + 4 + 5 = 15 \]

\[ \sum_{k=1}^{5} k^3 = 1^3 + 2^3 + 3^3 + 4^3 + 5^3 = 225 \]

\[ \sum_{k=1}^{5} \frac{8}{(4k - 3)(4k - 1)} = \frac{8}{3} + \frac{8}{35} + \frac{8}{99} + \frac{8}{195} + \frac{8}{323} = 3.04 \]

(Demo)
def cube(k):  
    return pow(k, 3)

def summation(n, term):
    """Sum the first n terms of a sequence.  
    >>> summation(5, cube)
    225  
    """
    total, k = 0, 1
    while k <= n:
        total, k = total + term(k), k + 1
    return total

0 + 1 + 8 + 27 + 64 + 125
Functions as Return Values

(Demo)
Locally Defined Functions

Functions defined within other function bodies are bound to names in a local frame.

```python
def make_adder(n):
    """Return a function that takes one argument k and returns k + n."
    def adder(k):
        return k + n
    return adder
```

The name `add_three` is bound to a function.

Can refer to names in the enclosing function.
Call Expressions as Operator Expressions

An expression that evaluates to a function

An expression that evaluates to its argument

operator

operand

func make_adder(n)

func adder(k)

make_adder(1) ( 2 )

make_adder(1) ( 2 )

1

make_adder(n):
def adder(k):
    return k + n
    return adder

func adder(k)