The Fibonacci Sequence

```python
def fib(n):
    """Compute the nth Fibonacci number, for n >= 2."""
    pred, curr = 0, 1  # First two Fibonacci numbers
    k = 2  # Tracks which Fib number is curr
    while k < n:
        pred, curr = curr, pred + curr
        k = k + 1
    return curr
```

Example: [http://goo.gl/dcaf0](http://goo.gl/dcaf0)

Practical Guidance: the Art of the Function

Give each function exactly one job.

Don’t repeat yourself (DRY). Implement a computational process just once, but execute it many times.

Define functions generally.

Generalizing Patterns with Arguments

Regular geometric shapes relate length and area.

Shape:  
- $r^2$  
- $r^2$  
- $\frac{3\sqrt{3}}{2}r^2$

Finding common structure allows for shared implementation

Generalizing Over Computational Processes

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

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

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

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

Summation Example

```python
def cube(k):
    return pow(k, 3)

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

>>> summation(5, cube)
225
```

Function of a single argument (not called term)

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

A formal parameter that will be bound to a function

```
def summation(n, term):
    """Sum the first n terms of a sequence."""
```

The cube function is passed as an argument value

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

The function bound to `term` gets called here

```
def cube(k):
    return pow(k, 3)
```
Locally Defined Functions

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

A function that returns a function

def make_adder(n):
    return adder

The name add_three is bound to a function

def add_three(k):
    return adder

Can refer to names in the enclosing function

def adder(k):
    return k + n

Call Expressions as Operator Expressions

make_adder(1)(2)

An expression that evaluates to any value

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

make_adder(1)(2)

Pig Introduction

Functions are first-class: Functions can be manipulated as values in our programming language.

Higher-order function: A function that takes a function as an argument value or returns a function as a return value

Higher-order functions:

- Express general methods of computation
- Remove repetition from programs
- Separate concerns among functions