```
"""Lecture 4 examples: Iteration and higher-order functions."""

from ucb import interact, main

# Iteration

def fib(n):
    """Compute the nth Fibonacci number, for n >= 2.

    >>> fib(8)
    13
    """
    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


# Generalizing patterns using arguments

from math import pi, sqrt

def area_square(r):
    """Return the area of a square with side length r."""
    return r * d

def area_circle(r):
    """Return the area of a circle with radius r."""
    return r * r * pi

def area_hexagon(r):
    """Return the area of a regular hexagon with side length r."""
    return r * r * 3 * sqrt(3) / 2

def area(r, shape_constant):
    """Return the area of a shape from length measurement r."""
    assert r >= 0, 'A length cannot be negative'
    return r * r * shape_constant

def area_square(r):
    return area(r, 1)

def area_circle(r):
    return area(r, pi)

def area_hexagon(r):
    return area(r, 3 * sqrt(3) / 2)


# Functions as arguments

def sum_naturals(n):
    """Sum the first n natural numbers.

    >>> sum_naturals(5)
    15
    """
    total, k = 0, 1
    while k <= n:
        total, k = total + k, k + 1
    return total

def sum_cubes(n):
    """Sum the first n cubes of natural numbers.

    >>> sum_cubes(5)
    225
    """
    total, k = 0, 1
    while k <= n:
        total, k = total + pow(k, 3), k + 1
```

```
    return total

def identity(k):
    return k

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

def pi_term(k):
    return 8 / (k * 4 - 3) / (k * 4 - 1)


# Local function definitions; returning functions

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

    >>> add_three = make_adder(3)
    >>> add_three(4)
    7
    """
    def adder(k):
        return k + n
    return adder

def compose1(f, g):
    """Return a function that composes f and g.

    f, g -- functions of a single argument
    """
    def h(x):
        return f(g(x))
    return h


@main
def run():
    interact()
```