Announcements

• Homework 1 due Tuesday 9/10 at 5pm; Late homework is not accepted!

• Quiz on Wednesday 9/11 released at 1pm, due Thursday 9/12 at 11:59pm
  
  • Open-computer: You can use the Python interpreter, watch course videos, and read the online text (http://composingprograms.com).

  • No external resources: Please don't search for answers, talk to your classmates, etc.

  • Content Covered: Lectures through last Friday 9/6; Same topics as Homework 1.

• Project 1 due next Thursday 9/19 at 11:59pm
Iteration Example
def fib(n):
    """Compute the nth Fibonacci number, for n >= 2."""
    predecessor, current = 0, 1  # First two Fibonacci numbers
    k = 2  # Tracks which Fibonacci number is called current
    while k < n:
        predecessor, current = current, (predecessor + current)
        k = k + 1
    return current

Example: http://goo.gl/vfymhd
Discussion Question

Complete the following definition by placing an expression in ________________.

```python
def choose(total, selection):
    """Return the number of ways to choose SELECTION items from TOTAL.
    
    choose(n, k) is typically defined in math as:  n! / (n-k)! / k!
    
    >>> choose(5, 2)
    10
    >>> choose(20, 6)
    38760
    """
    ways = 1
    selected = 0
    while selected < selection:
        selected = selected + 1
        ways, total = ways * n · (n − 1) · (n − 2) · ... · (n − k + 1)  
                      k · (k − 1) · (k − 2) · ... · 2 · 1
        total // selected, total - 1
    return ways
```

Example: http://goo.gl/38ch3o
Default Arguments

(Demo)
Designing Functions
Characteristics of Functions

**def square(x):**

"""Return X * X."""

**def choose(n, d):**

"""Return the number of ways to choose D of N items."""

A function's **domain** is the set of all inputs it might possibly take as arguments.

- **x is a number**
- **n and d are positive integers with n greater than or equal to d.**

A function's **range** is the set of output values it might possibly return.

- **return value is a positive number**
- **return value is a positive integer**

A pure function's **behavior** is the relationship it creates between input and output.

- **return value is the square of the input**
- **return value is the number of ways to choose d of n items.**
A Guide to Designing Function

Give each function exactly one job.

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

Define functions generally.
Generalization
Generalizing Patterns with Arguments

Regular geometric shapes relate length and area.

**Shape:**

- Square: \( r \)
- Circle: \( r \)
- Hexagon: \( r \)

**Area:**

- Square: \( \frac{1}{1} \cdot r^2 \)
- Circle: \( \pi \cdot r^2 \)
- Hexagon: \( \frac{3\sqrt{3}}{2} \cdot r^2 \)

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."
    total, k = 0, 1
    while k <= n:
        total, k = total + term(k), k + 1
    return total

>>> summation(5, cube)
225
"""
Functions as Return Values
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

>>> add_three = make_adder(3)
>>> add_three(4)
7

A function that returns a function

The name add_three is bound to a function

A local def statement

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

An expression that evaluates to a function

Operator

An expression that evaluates to any value

Operand

make_adder(1)     (         2         )

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

17
The Purpose of Higher-Order Functions

**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
The Game of Hog

(Demo)