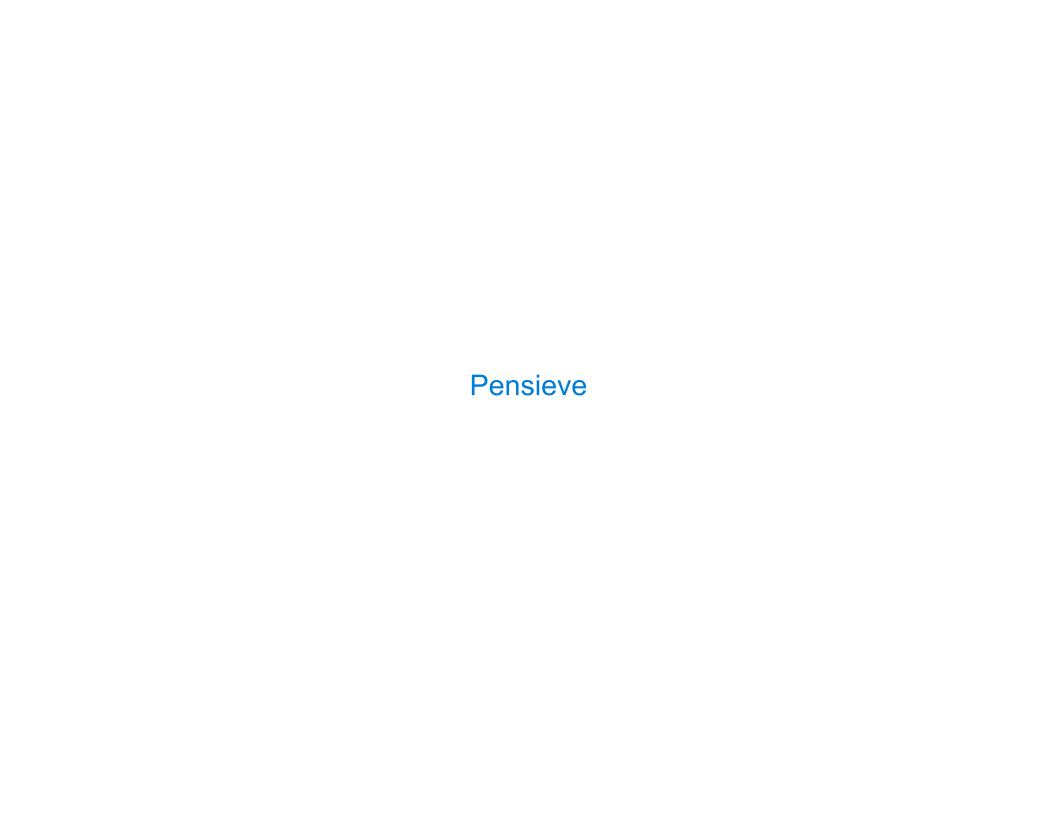


Office	Hours:	You	Shoul	ld Gol
	illouis.	I O U	OHOU	iu Ou:

You are not alone!

https://cs61a.org/office-hours/



Designing Functions

Describing Functions

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

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

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

def square(x):
 """Return X * X."""

x is a number

square returns a nonnegative real number

square returns the square of x

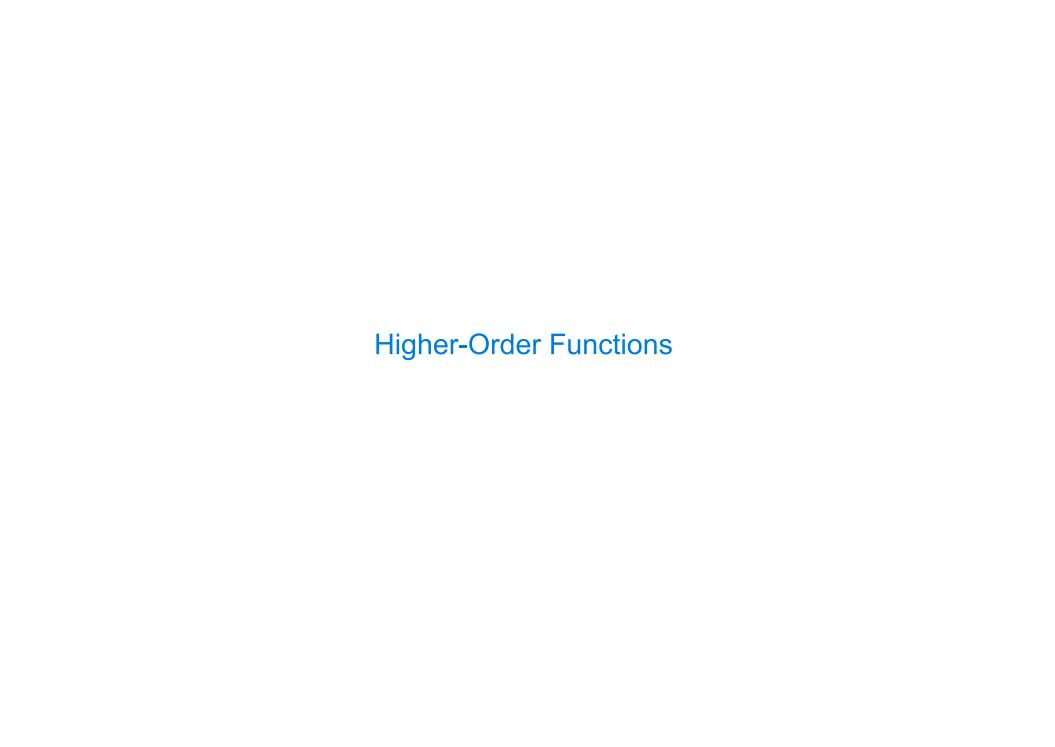
A Guide to Designing Function

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

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

(Demo)

-



Summation Example

```
Function of a single argument
def cube(k):
                                 (not called "term")
     return pow(k, 3)
                            A formal parameter that will
def summation(n, term)
                               be bound to a function
     """Sum the first n terms of a sequence.
     >>> summation(5, cube)
     225
                           The cube function is passed
     11 11 11
                              as an argument value
     total, k = 0, 1
     while k <= n:
          total, k = total + term(k), k + 1
     return total
                             The function bound to term
  0 + 1 + 8 + 27 + 64 + 125
                                 gets called here
```

9

Program Design

Modularity

Abstraction

Separation of Concerns

Twenty-One Rules

Two players alternate turns, on which they can add 1, 2, or 3 to the current total

The total starts at 0

The game end whenever the total is 21 or more

The last player to add to the total loses

(Demo)

Functions as Return Values

(Demo)

Locally Defined Functions

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

```
A function that
 returns a function
def make_adder(n):
    """Return a function that takes one argument k and returns k+n.
    >>> (add_three = make_adder(3)) 
                                         The name add_three is bound
                                                to a function
    >>> add three(4)
    11 11 11
    def adder(k):
                           A def statement within
         return(k + n)
                           another def statement
    return adder
               Can refer to names in the
                   enclosing function
```

Twenty-One Strategies

(Demo)