High Order Functions

Contents

- Functions as arguments
- Nested functions
- Functions as return values
- The operator module

Passing a function as an argument to another function

- What a higher-order function is
- How functions are objects

Higher-order functions

- Achieve expressive, succinct programs using higher-order functions
- These are functions that accept a function as an argument or return a function as a value.
- Used to create composite functions from simpler functions.

A higher-order function is a function that operates on other functions

Objects

Objects everywhere

- In Python,
  - (Almost) everything is an object
  - Can be passed as an argument to a function

Nesting a function in another function

- How to define nested functions
- How nested functions affect variable scope
- How to control variable scope

Scope

Who can see what?

- If you define a variable inside a function
- You cannot use it outside of this function
- Phrased differently, the scope of the variable is the function

Nesting

Functions inside functions

- A nested function is a function that is defined inside another function
- This is no different from defining a variable inside a function

Returning a function from another function

- How to return a function from another function
- An example – Creating a function that calls several other functions

Objects

Objects everywhere

- Because functions are objects
- They cannot only passed to functions as arguments
- But also returned by functions as return values

Operator as Regular functions

- The problem that operators are not objects
- Effects of the problem
- Solution – The Operator module

Operators

Syntax or functions?

- Operators are things like +, -, /, *, and so on
- These symbols are part of Python syntax
- They are not objects
- You cannot pass them as arguments to functions