starting out with >>> PYTHON THIRD EDITION

CHAPTER 5

Functions

TONY GADDIS

Topics (cont'd.)

- Introduction to Value-Returning Functions: Generating Random Numbers
- Writing Your Own Value-Returning Functions
- The `math` Module
- Storing Functions in Modules

Introduction to Value-Returning Functions: Generating Random Numbers

- **void function: group of statements within a program for performing a specific task**
  - Call function when you need to perform the task
- **Value-returning function: similar to void function, returns a value**
  - Value returned to part of program that called the function when function finishes executing

Standard Library Functions and the `import` Statement

- **Standard library: library of pre-written functions that comes with Python**
  - *Library functions* perform tasks that programmers commonly need
    - Example: `print`, `input`, `range`
    - Viewed by programmers as a "black box"
- **Some library functions built into Python interpreter**
  - To use, just call the function

Standard Library Functions and the `import` Statement (cont'd.)

- **Modules: files that stores functions of the standard library**
  - Help organize library functions not built into the interpreter
  - Copied to computer when you install Python
- **To call a function stored in a module, need to write an `import` statement**
  - Written at the top of the program
  - Format: `import module_name`

Standard Library Functions and the `import` Statement (cont'd.)

Figure 5-19 A library function viewed as a black box

Generating Random Numbers

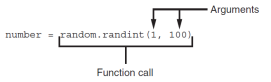
- **Random number are useful in a lot of programming tasks**
- **`random` module: includes library functions for working with random numbers**
- **Dot notation: notation for calling a function belonging to a module**
  - Format: `module_name.function_name()`

Generating Random Numbers (cont'd.)

- **`randint` function: generates a random number in the range provided by the arguments**
  - Returns the random number to part of program that called the function
  - Returned integer can be used anywhere that an integer would be used
  - You can experiment with the function in interactive mode

Generating Random Numbers (cont'd.)

Figure 5-20 A statement that calls the `random` function



Generating Random Numbers (cont'd.)

Figure 5-21 The `random` function returns a value

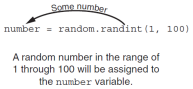
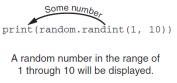


Figure 5-22 Displaying a random number



Generating Random Numbers (cont'd.)

- **`randrange` function: similar to `range` function, but returns randomly selected integer from the resulting sequence**
  - Same arguments as for the `range` function
- **`random` function: returns a random float in the range of 0.0 and 1.0**
  - Does not receive arguments
- **`uniform` function: returns a random float but allows user to specify range**

Random Number Seeds

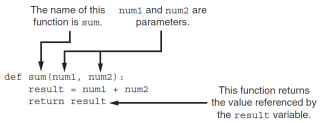
- **Random number created by functions in random module are actually pseudo-random numbers**
- **Seed value: initializes the formula that generates random numbers**
  - Need to use different seeds in order to get different series of random numbers
    - By default uses system time for seed
    - Can use `random.seed()` function to specify desired seed value

Writing Your Own Value-Returning Functions

- **To write a value-returning function, you write a simple function and add one or more `return` statements**
  - Format: `return` *`expression`*
    - The value for *`expression`* will be returned to the part of the program that called the function
  - The expression in the `return` statement can be a complex expression, such as a sum of two variables or the result of another value-returning function

Writing Your Own Value-Returning Functions (cont'd.)

Figure 5-23 Parts of the function



How to Use Value-Returning Functions

- **Value-returning function can be useful in specific situations**
  - Example: have function prompt user for input and return the user's input
  - Simplify mathematical expressions
  - Complex calculations that need to be repeated throughout the program
- **Use the returned value**
  - Assign it to a variable or use as an argument in another function

Using IPO Charts

- **IPO chart: describes the input, processing, and output of a function**
  - Tool for designing and documenting functions
  - Typically laid out in columns
  - Usually provide brief descriptions of input, processing, and output, without going into details
    - Often includes enough information to be used instead of a flowchart

Using IPO Charts (cont'd.)

Figure 5-25 IPO charts for the `getRegularPrice` and `discount` functions

IPO Chart for the `get_regular_price` Function		
Input	Processing	Output
None	Prompts the user to enter an item's regular price	The item's regular price

IPO Chart for the `discount` Function		
Input	Processing	Output
An item's regular price	Calculates an item's discount by multiplying the regular price by the global constant `DISCOUNT_PERCENTAGE`	The item's discount

Returning Strings

- **You can write functions that return strings**
- **For example:**

```
def get_name():
    # Get the user's name.
    name = input('Enter your name: ')
    # Return the name.
    return name
```

Returning Boolean Values

- **Boolean function: returns either `True` or `False`**
  - Use to test a condition such as for decision and repetition structures
    - Common calculations, such as whether a number is even, can be easily repeated by calling a function
  - Use to simplify complex input validation code

Returning Multiple Values

- **In Python, a function can return multiple values**
  - Specified after the `return` statement separated by commas
    - Format: `return` *`expression1, expression2, etc.`*
  - When you call such a function in an assignment statement, you need a separate variable on the left side of the = operator to receive each returned value

The `math` Module

- **`math` module: part of standard library that contains functions that are useful for performing mathematical calculations**
  - Typically accept one or more values as arguments, perform mathematical operation, and return the result
  - Use of module requires an `import math` statement

The `math` Module (cont'd.)

Table 5-2 Many of the functions in the `math` module

`math` Module Function	Description
`acos(x)`	Returns the arc cosine of `x`, in radians.
`asin(x)`	Returns the arc sine of `x`, in radians.
`atan(x)`	Returns the arc tangent of `x`, in radians.
`ceil(x)`	Returns the smallest integer that is greater than or equal to `x`.
`cos(x)`	Returns the cosine of `x` in radians.
`degrees(x)`	Assuming `x` is an angle in radians, the function returns the angle converted to degrees.
`exp(x)`	Returns e^x
`floor(x)`	Returns the largest integer that is less than or equal to `x`.
`hypot(x, y)`	Returns the length of a hypotenuse that extends from (0, 0) to (`x`, `y`).
`log(x)`	Returns the natural logarithm of `x`.
`log10(x)`	Returns the base-10 logarithm of `x`.
`radians(x)`	Assuming `x` is an angle in degrees, the function returns the angle converted to radians.
`sin(x)`	Returns the sine of `x` in radians.
`sqrt(x)`	Returns the square root of `x`.
`tan(x)`	Returns the tangent of `x` in radians.

The `math` Module (cont'd.)

- **The `math` module defines variables `pi` and `e`, which are assigned the mathematical values for *pi* and *e***
  - Can be used in equations that require these values, to get more accurate results
- **Variables must also be called using the dot notation**
  - Example:

```
circle_area = math.pi * radius**2
```


Storing Functions in Modules

- **In large, complex programs, it is important to keep code organized**
- **Modularization: grouping related functions in modules**
  - Makes program easier to understand, test, and maintain
  - Make it easier to reuse code for multiple different programs
    - Import the module containing the required function to each program that needs it

Storing Functions in Modules (cont'd.)

- **Module is a file that contains Python code**
  - Contains function definition but does not contain calls to the functions
    - Importing programs will call the functions
- **Rules for module names:**
  - File name should end in `.py`
  - Cannot be the same as a Python keyword
- **Import module using `import` statement**

Menu Driven Programs

- **Menu-driven program: displays a list of operations on the screen, allowing user to select the desired operation**
  - List of operations displayed on the screen is called a *menu*
- **Program uses a decision structure to determine the selected menu option and required operation**
  - Typically repeats until the user quits

Summary

- **This chapter covered:**
  - The advantages of using functions
  - The syntax for defining and calling a function
  - Methods for designing a program to use functions
  - Use of local variables and their scope
  - Syntax and limitations of passing arguments to functions
  - Global variables, global constants, and their advantages and disadvantages

Summary (cont'd.)

- Value-returning functions, including:
  - Writing value-returning functions
  - Using value-returning functions
  - Functions returning multiple values
- Using library functions and the `import` statement
- Modules, including:
  - The `random` and `math` modules
  - Grouping your own functions in modules