KEY TERMS

actual parameter (i.e., argument): the variables or data to substitute formal parameters when invoking a function.

argument: same as actual parameter.

caller: the main program or a function that calls another function.

default argument: default value for a parameter if the argument is not specified when the function is invoked.

divide and conquer: divides the problem into subproblems, solved the subproblems, then combine the solutions of the subproblems to obtain the solution for the entire problem.

formal parameter (i.e., parameter): the variables defined in the function signature.

function: a function performs actions.

a collection of statements grouped together to perform an operation.

function abstraction: a technique in software development that hides detailed implementation. function abstraction is defined as separating the use of a function from its implementation. the client can use a function without knowing how it is implemented. if you decide to change the implementation, the client program will not be affected.

function header: the combination of the name of a function and the list of its parameters.

global variable: a variable declared outside all functions and are accessible to all functions in its scope.

immutable objects: an object whose contents cannot be changed.

information hiding: the details of the implementation are encapsulated in the function and hidden from the client that invokes the function. this is known as information hiding or encapsulation.

keyword arguments: pass arguments associated with the parameter names.

local variable: a variable declared inside a function.

**None**: a special value assigned to a variable, meaning that the variable does not hold any value.

**None** function: a function that does not return a value.

parameter: variables defined in the function signature.

positional arguments: pass arguments to parameter according based on the positions.

return value: a value returned from a function using the return statement.

stepwise refinement: when writing a large program, you can use the “divide and conquer” strategy, also know as stepwise refinement, to decompose it into subproblems. the subproblems can be further decomposed into smaller, more manageable problems.

stub: a simple, but not a complete version of the function. the use of stubs enables you to test invoking the function from a caller.

void function: a function that does not return a value.