**Chapter 5 – Functions**

1. Introduction to Functions – A function is a group of statements that exist within a program for the purpose of performing a specific task.
   1. Introduction
      1. Programs are broken down into *functions*
         1. A group of statements within a program to perform a specific task
      2. Instead of writing large programs as many statements, can be written as several small functions
         1. Each one performs a specific task
         2. Called divide and conquer
      3. A program with each task as its own program is a modularized program
   2. Benefits of Modularizing a Program with Functions
      1. Simpler Code
         1. Tends to be simper and easier to understand
         2. Several small functions are easier to read and follow than one long one
      2. Code Reuse
         1. Reduce the duplication of code within a program
         2. If something needs to be done multiple times at different times in a program, one piece of code can be written once and executed whenever necessary
      3. Better Testing
         1. Testing and debugging are simpler when each task is in its own function
         2. Functions can be tested individually to check performance
         3. Easier to isolate and fix errors
      4. Faster Development
         1. Several programmers can work on a program together by each taking a separate function
         2. Programs for simple functions can be written and incorporated into programs that need them
      5. Easier Facilitation of Teamwork
         1. Different programmers can be assigned different functions to work in in a program
   3. Void Functions and Value-Returning Functions
      1. Void Functions
         1. Executes the statements within itself, and then terminates
      2. Value-returning Function
         1. Executes the statements withing itself, and returns a value back
2. Defining and Calling a Void Function – The code for a function is known as a function definition. To execute the function, you write a statement that calls it.
   1. Function Names
      1. Naming functions follows the same rules as naming variables
      2. Should be descriptive so anyone reading can know what the function does
   2. Defining and Calling a Function
      1. “def function\_name():” is known as the function header
         1. Marks the beginning of the function definition
      2. Followed by a block of statements
      3. Calling a Function
         1. To execute a function, it must be called
            1. Ex: to call program 5-1, type message() in to interpreter
         2. Possible to have many functions in a program
         3. Common to have a *main* function call other functions
            1. Program 5-2, pg. 214 shows this
3. Designing a Program to Use Functions – Programmers commonly use a technique known as top-down design to break down an algorithm into functions,
   1. Flowcharting a Program with Functions
      1. Programmers use separate flowcharts for each function in a program
   2. Top-Down Design
      1. Formed with the following steps:
         1. Overall task is broken down into subtasks
         2. Each subtask is checked if it can be broken down further
         3. Once all subtasks are identified, coding starts
   3. Hierarchy Charts
      1. Also known as structure charts
      2. Show the relationship between functions
      3. Figure 5-10, pg. 219 shows a chart
   4. Pausing Execution Until the User Presses Enter
      1. Use input function to pause execution of functions
4. Local Variables – A local variable is created inside a function and cannot be accessed by statements that are outside the function. Different functions can have local variables with the same names because the functions cannot see each other’s local variables.
   1. Introduction
      1. Anytime a variable is assigned within a function, a local variable is created
      2. Belongs to the function in which it is created
      3. Only statements inside the function can see and access local variables
   2. Scope and Local Variables
      1. Variable is only visible to statements in its scope
         1. Local variable’s scope is the function in which the variable is created
      2. No statement outside the function may access the variable
5. Passing Arguments to Functions – An argument is any piece of data that is passed into a function when the function is called. A parameter is a variable that receives an argument that is passed into a function.
   1. Introduction
      1. Pieces of data that are sent into a function are arguments
      2. Function can use its arguments in calculations or other operations
      3. Functions must be equipped with parameter variables
         1. Also known as parameters
   2. Parameter Variable Scope
      1. Scope of a parameter is the function in which the parameter is used
      2. All statements inside of the function can access the parameter, but no statement outside can access it
   3. Passing Multiple Arguments
      1. Program 5-8, pg. 229 shows a function using two arguments
      2. Program 5-9, pg. 231 shows a program passing two strings as arguments
   4. Making Changes to Parameters
      1. Any changes made to the parameter variable will not affect the argument
      2. Commonly called pass by value
   5. Keyword Arguments
      1. Some arguments are passed based on position
         1. Program 5-8 and 5-9 show this
         2. Conventional form of passing arguments
      2. Arguments can be written with the keyword argument format:
         1. parameter\_name=value
      3. Program 5-11, pg. 233 shows this
      4. Position doesn’t matter in keyword format
6. Global Variables and Global Constants – A global variable is accessible to all functions in a program file.
   1. Global Variables
      1. When variable is assigned outside of all functions, a global variable is created
      2. GV usage should be restricted or avoided completely
         1. GVs make debugging hard
            1. If it gets changed in a function, every statement that uses the GV must be found and changed
         2. Functions that use GVs are usually dependent on them to function
            1. To use them in a different program, must be redesigned not to use GVs
         3. GVs make a program hard to understand
            1. If one function uses GV, you must know all other functions that use the same GV
   2. Global Constants
      1. A global name that references a value that can’t be changed
      2. Okay to use in a program since they can’t be changed
      3. Python doesn’t allow global constants to be assigned like GVs
         1. Must simulate with global variables instead
7. Introduction to Value-Returning Functions: Generating Random Numbers – A value-returning function is a function that returns a value back to the part of the program that called it. Python, as well as most other programming languages, proves a library of prewritten functions that perform commonly needed tasks. These libraries typically contain a function that generates random numbers.
   1. Introduction
      1. A special type of function that returns a value back to the program that called it
   2. Standard Library Functions and the *import* Statement
      1. Python comes with a standard library of functions that have already been written
         1. Known as library functions
      2. Make programming easier
         1. Functions that perform tasks that are commonly needed
         2. Ex: print, range, input
      3. Functions are stored as modules
      4. In order to use library functions, programmer must write import statements at the top of the program
   3. Generating Random Numbers
      1. Python has several random number functions
         1. Must write “import random” at the top of the program
      2. Program 5-16 and 5-17, pg. 242 show how to use randint function
   4. Experimenting with Random Numbers in Interactive Mode
      1. Must “import random”
      2. Program 5-19 , pg. 245 shows how to use this to simulate rolling dice
   5. The *randrange, random,* and *uniform* Functions
   6. Random Number Seeds
      1. Uses value as the seed instead of system time
      2. Produces the same set of “random” numbers each time
8. Writing Your Own Value-Returning Functions – A value-returning function has a return statement that returns a value back to the part of the program that called it.
   1. Introduction
      1. Value-return and void functions are written the same way
         1. Except value-return must have a *return* statement
      2. The expression that gets called by return will be sent back to the program that called the function
   2. Making the most of the *return* Statement
      1. Don’t need to return a variable
      2. Can return a value
         1. Ex: return num1+num2
         2. Instead of result=num1+num2\return result
   3. How to Use Value-Returning Functions
      1. VR functions have many benefits:
         1. Simplify code
         2. Reduce duplication
         3. Enhance ability to test code
         4. Increase speed of development
         5. Ease facilitation of teamwork
      2. Can be useful in specific situations
         1. Program 5-22, pg. 253
   4. Using IPO Charts
      1. Simple tool that programmers use to design and document functions
      2. Stands for input, processing, and output
      3. Describes input, processing, and output of a function
      4. IPO charts only show brief descriptions of each column
         1. Do not specify exact steps
      5. Include enough information to be used in place of a flowchart
      6. Choice between IPO and flowchart is left to programmer’s discretion
   5. Returning Strings
      1. Functions can also return strings as well as numbers
         1. Ex: *return* name
   6. Returning Boolean Values
      1. True or False can be returned back to the program that called
   7. Returning Multiple Values
      1. Multiple expressions can be returned in one statement by separating with commas
9. The *math* Module – The Python standard library’s *math* module contains numerous functions that can be used in mathematical calculations
   1. Introduction
      1. Contains several useful math functions
      2. Table 5-2, pg. 263 shows many of the functions in the library
         1. All return float values except ceil and floor
            1. They return int values
   2. The *math.pi* and *math.e* Values
      1. Pi and e are assigned in python and can be used in equations
10. Storing Functions in Modules – A module is a file that contains Python code. Large programs are easier to debug and maintain when they are divided into modules.
    1. Introduction
       1. Storing in modules makes debugging, reading, duplicating, and reusing code easier
       2. Module naming must follow certain guidelines
          1. File name should end in .py
             1. If not, you can’t import to other programs
          2. Module name cannot be the same as a key word
             1. Ex: print, for, in, range, etc.…
    2. Menu Driven Programs
       1. Program 5-25, pg. 267 is an MDP
       2. Allows user to select the operation to perform