**CHAPTER 5 Part 1 “lab gems”**

* **Function: group of statements within a program that perform as specific task**
  + **Usually one task of a large program**
  + **Functions can be executed in order to perform overall program task**
  + **Known as *divide and conquer* approach**
* **Modularized program: program wherein each task within the program is in its own function**
  + **The benefits of using functions include:**
  + **Simpler code**
  + **Code reuse**
  + **write the code once and call it multiple times**
  + **Better testing and debugging**
  + **Can test and debug each function individually**
  + **Faster development**
  + **Easier facilitation of teamwork**
  + **Different team members can write different functions**
* **A void function:**
  + **Simply executes the statements it contains and then terminates.**
* **A value-returning function:**
  + **Executes the statements it contains, and then it returns a value back to the statement that called it.**
  + **The input, int, and float functions are examples of value-returning functions.**
* **Functions are given names** 
  + **Function naming rules:**
  + **Cannot use key words as a function name**
  + **Cannot contain spaces**
  + **First character must be a letter or underscore**
  + **All other characters must be a letter, number or underscore**
  + **Uppercase and lowercase characters are distinct**
* **Function name should be descriptive of the task carried out by the function**
  + **Often includes a verb**
  + **Function definition: specifies what function does**
  + **def *function\_name*():**

**statement**

**statement**

* **Function header: first line of function**
  + **Includes keyword def and function name, followed by parentheses and colon**
* **Block: set of statements that belong together as a group**
  + **Example: the statements included in a function**
* **Call a function to execute it**
  + **When a function is called:**
    - **Interpreter jumps to the function and executes statements in the block**
    - **Interpreter jumps back to part of program that called the function**
      * **Known as function return**
* **main function: called when the program starts**
  + **Calls other functions when they are needed**
  + **Defines the *mainline logic* of the program**
* **Each block must be indented**
  + **Lines in block must begin with the same number of spaces**
    - **Use tabs or spaces to indent lines in a block, but not both as this can confuse the Python interpreter**
    - **IDLE automatically indents the lines in a block**
  + **Blank lines that appear in a block are ignored**
* **In a flowchart, function call shown as rectangle with vertical bars at each side**
  + **Function name written in the symbol**
  + **Typically draw separate flow chart for each function in the program**
    - **End terminal symbol usually reads Return**
* **Top-down design: technique for breaking algorithm into functions**
* **Hierarchy chart: depicts relationship between functions**
  + **AKA structure chart**
  + **Box for each function in the program, Lines connecting boxes illustrate the functions called by each function**
  + **Does not show steps taken inside a function**
* **Use input function to have program wait for user to press enter**
* **Local variable: variable that is assigned a value inside a function**
  + **Belongs to the function in which it was created**
    - **Only statements inside that function can access it, error will occur if another function tries to access the variable**
* **Scope: the part of a program in which a variable may be accessed**
  + **For local variable: function in which created**
* **Local variable cannot be accessed by statements inside its function which precede its creation**
* **Different functions may have local variables with the same name** 
  + **Each function does not see the other function’s local variables, so no confusion**
* **Argument: piece of data that is sent into a function**
  + **Function can use argument in calculations**
  + **When calling the function, the argument is placed in parentheses following the function name**
* **Parameter variable: variable that is assigned the value of an argument when the function is called**
  + **The parameter and the argument reference the same value**
  + **General format:**
  + **def *function*\_*name*(*parameter*):**
  + **Scope of a parameter: the function in which the parameter is used**
* **Python allows writing a function that accepts multiple arguments**
  + **Parameter list replaces single parameter**
    - **Parameter list items separated by comma**
* **Arguments are passed *by position* to corresponding parameters**
  + **First parameter receives value of first argument, second parameter receives value of second argument, etc.**
* **Changes made to a parameter value within the function do not affect the argument**
  + **Known as *pass by value***
  + **Provides a way for unidirectional communication between one function and another function**
    - **Calling function can communicate with called function**
* **Keyword argument: argument that specifies which parameter the value should be passed to**
  + **Position when calling function is irrelevant**
  + **General Format:**
  + **function\_name(parameter=value)**
* **Possible to mix keyword and positional arguments when calling a function**
  + **Positional arguments must appear first**
* **Global variable: created by assignment statement written outside all the functions**
  + **Can be accessed by any statement in the program file, including from within a function**
  + **If a function needs to assign a value to the global variable, the global variable must be redeclared within the function**
    - **General format: global *variable\_name***
* **Reasons to avoid using global variables:**
  + **Global variables making debugging difficult**
    - **Many locations in the code could be causing a wrong variable value**
  + **Functions that use global variables are usually dependent on those variables**
    - **Makes function hard to transfer to another program**
  + **Global variables make a program hard to understand**
* **Global constant: global name that references a value that cannot be changed**
  + **Permissible to use global constants in a program**
  + **To simulate global constant in Python, create global variable and do not re-declare it within functions**