**CHAPTER 4 “lab gems”**

* **Often have to write code that performs the same task multiple times**
  + **Disadvantages to duplicating code**
    - **Makes program large**
    - **Time consuming**
    - **May need to be corrected in many places**
* **Repetition structure: makes computer repeat included code as necessary**
  + **Includes condition-controlled loops and count-controlled loops**
* **while loop: while condition is true, do something**
  + **Two parts:** 
    - **Condition tested for true or false value**
    - **Statements repeated as long as condition is true**
  + **In flow chart, line goes back to previous part**
  + **General format:**

**while *condition*:**

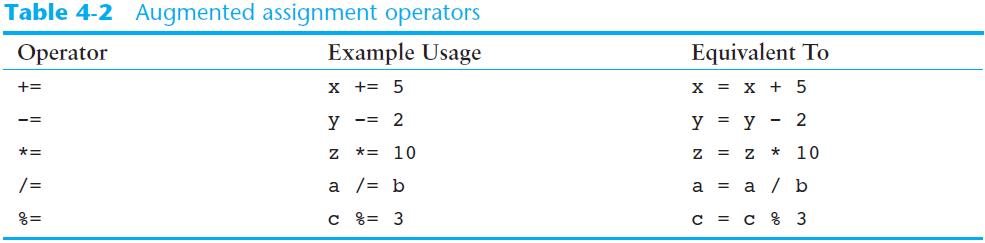
***statements***

* **In order for a loop to stop executing, something has to happen inside the loop to make the condition false**
* **Iteration: one execution of the body of a loop**
* **while loop is known as a *pretest* loop**
  + **Tests condition before performing an iteration**
    - **Will never execute if condition is false to start with**
    - **Requires performing some steps prior to the loop**
* **Loops must contain within themselves a way to terminate**
  + **Something inside a while loop must eventually make the condition false**
* **Infinite loop: loop that does not have a way of stopping**
  + **Repeats until program is interrupted**
  + **Occurs when programmer forgets to include stopping code in the loop**
* **Count-Controlled loop: iterates a specific number of times**
  + **Use a for statement to write count-controlled loop** 
    - **Designed to work with sequence of data items**
      * **Iterates once for each item in the sequence**
    - **General format:**

**for *variable* in *[val1, val2, etc]*:**

***statements***

* + - **Target variable: the variable which is the target of the assignment at the beginning of each iteration**
* **The range function simplifies the process of writing a for loop**
  + **range returns an iterable object**
    - **Iterable: contains a sequence of values that can be iterated over**
* **range characteristics:**
  + **One argument: used as ending limit**
  + **Two arguments: starting value and ending limit**
  + **Three arguments: third argument is step value**
* **Purpose of target variable is to reference each item in a sequence as the loop iterates**
* **Target variable can be used in calculations or tasks in the body of the loop**
  + **Example: calculate square root of each number in a range**
* **Sometimes the programmer does not know exactly how many times the loop will execute**
* **Can receive range inputs from the user, place them in variables, and call the range function in the for clause using these variables**
  + **Be sure to consider the end cases: range does not include the ending limit**
* **The range function can be used to generate a sequence with numbers in descending order**
  + **Make sure starting number is larger than end limit, and step value is negative**
  + **Example: range (10, 0, -1)**
* **Programs often need to calculate a total of a series of numbers**
  + **Typically include two elements:**
    - **A loop that reads each number in series**
    - **An *accumulator* variable**
  + **Known as program that keeps a running total: accumulates total and reads in series**
  + **At end of loop, accumulator will reference the total**
* **In many assignment statements, the variable on the left side of the = operator also appears on the right side of the = operator**
* **Augmented assignment operators: special set of operators designed for this type of job**
  + **Shorthand operators**

****

* **Sentinel: special value that marks the end of a sequence of items**
  + **When program reaches a sentinel, it knows that the end of the sequence of items was reached, and the loop terminates**
  + **Must be distinctive enough so as not to be mistaken for a regular value in the sequence**
  + **Example: when reading an input file, empty line can be used as a sentinel**
* **Computer cannot tell the difference between good data and bad data**
  + **If user provides bad input, program will produce bad output**
  + **GIGO: garbage in, garbage out**
  + **It is important to design program such that bad input is never accepted**
* **Input validation: inspecting input before it is processed by the program**
  + **If input is invalid, prompt user to enter correct data**
  + **Commonly accomplished using a while loop which repeats as long as the input is bad**
    - **If input is bad, display error message and receive another set of data**
    - **If input is good, continue to process the input**
* **Nested loop: loop that is contained inside another loop**
  + **Example: analog clock works like a nested loop**
    - **Hours hand moves once for every twelve movements of the minutes hand: for each iteration of the “hours,” do twelve iterations of “minutes”**
    - **Seconds hand moves 60 times for each movement of the minutes hand: for each iteration of “minutes,” do 60 iterations of “seconds”**
* **Key points about nested loops:**
  + **Inner loop goes through all of its iterations for each iteration of outer loop**
  + **Inner loops complete their iterations faster than outer loops**
  + **Total number of iterations in nested loop: number\_iterations\_inner x**

**number\_iterations\_outer**