# User-Computer Dialog Rules

One of programming skills you will acquire primarily through trial and error is how to structure the user-computer dialog. Although messages and instructions generated by a computer appear to be intelligent, whatever intelligence they exhibit comes from the program designer and programmer. Since these people will likely not be available when a user interacts with the computer, the user-computer dialog needs to be self-explanatory, specific, unambiguous, complete, and carefully structured. The less cryptic is often the better.

Rules like those below are not standardized; these come from the instructor’s experience and are a requirement for this course.

Many programs require users to enter data and, after the program does its processing, will present output to the user. The entire dialog should occur in the same screen or window; do not tell the user what the program will do in a pop up window and then ask for input from the console, or *vice versa*.

1. Prompt the user for input with a message indicating
   1. Very briefly what the program will do or is for,
   2. what input is expected, such as a number (integer), a file name, or a String such as the name of a student, and
   3. special formatting instructions, such as separating entries with spaces or commas when more than one entry is expected as in, for instance, “Enter the x- and y-coordinate of a point separated by a space and without parentheses.”
2. The program descriptions and prompts should be right before the user is expected to enter the inputs. If pop-up windows are used the program description and prompt should be in the same window as the entry box into which the user is expected to enter the inputs. Multiple prompt windows are acceptable but a single one is preferred.

Two of the basic “rules” of dialog are that

1. All input to a program must be echoed back to the user, preferably in the same pop-up window as the output, and labeled for what it is.
2. All output is to be labeled sufficiently so the user can identify and interpret it properly.

These rules are not always practical for large amounts of input/output but for almost all problems in this course they should be followed. If the output does not make sense to the user, it is frequently because of input errors; being able to see the input right along with the output helps users to diagnose the problem and possibly correct their input and/or assumptions.

In addition, all input, even if only implicitly, has rules for validity (passwords often have to greater than a certain length, contain some specified number of letters, numbers, and punctuation marks, etc.).

1. These validity rules should be communicated to the user explicitly, either as part of the prompt or in associated documentation. For applications in the real world often both are supplied.
2. Starting with exercises from Chapter 3, your program should check the user input for such validity and when input fails to follow these rules what is not valid must be communicated to the user.
3. Starting with exercises from Chapter 4, after invalid input is identified the program should allow the user to re-enter the input. This loop should continue until the program executes successfully. From this assignment on exception handling is to be used for input validation. Assignment 4 contains videos and example files.

After the first assignment you will be asked to use popup windows for both input and output. The sooner you master these the more appealing your programs will be. Your instructor will provide suggestions as you go along.