Conditional Execution – the IF Statement

In this exercise you will again compute the gross weight and CG position of a small aircraft. This time, you will use the "conditional" statement (also called the "`if`" statement) to change the flow of control of your program, allowing it to make decisions.

In the previous exercise your program computed the total weight and position of the center of gravity (CG) of a small airplane, but it was up to the pilot to know whether the results are acceptable. Using the `if` statement you can have your program warn the pilot if the total weight is too much, and/or if the CG position is unacceptable.

Conditional execution

The general form of a conditional (`if`) statement in Python is

```
if BOOLEAN-EXPRESSION :
    STATEMENT(S)
```

The line begins with "`if`", followed by a mathematical expression which evaluates to either true or false, followed by a colon. (Don't forget the colon! – everybody does at first when learning Python). Those lines of code which follow *which are indented*, are then only executed if the boolean expression is true. If the expression evaluates to "false" then the indented lines of code are skipped.

What is a "Boolean expression"? The term "Boolean" refers to Boolean logic, the kind that involves statements that are either "true" or "false". A Boolean expression can involve mathematical comparison operators to test if something is larger or smaller than something else, or a variety of other conditions. You can also have compound Boolean expressions that use multiple operators, including operators to represent the logical ideas of "and" and "or." Table 1 is a list of the comparison and Boolean operators in Python.

It is also possible to have the `if` statement choose one of two alternatives, by adding an "else" clause after the first indented block of code. The word "`else:`" (remember the colon!) is not indented, but the lines after it which are indented are executed only if the original expression in the `if` statement is false. Here is a simple example:

```
if  x >= 0 :
    print("x is not negative")
else:
    print("x is negative")
```

`==`	equal
`!=`	not equal
`>`	greater than
`<`	less than
`>=`	greater than or equal to
`<=`	less than or equal to
`and`	and – both must be true
`or`	or – either may be true

Table 1: Comparison and Boolean operators in Python.

There is no standard number of columns to indent in Python, but it is usually suggested that you indent by 4 characters, since that is enough to make it clear when anyone reads your code that those statements are indented, but not too much (which becomes important when you have `if` statements inside of `if` statements, and so on). Indenting is a very important part of Python, because it is mandatory in order to group together lines of code (what is sometimes called "scope," which is specified in many other computer languages using curly brackets).

IDLE Configuration Options

By default, when you launch IDLE without editing a particular file, it starts you out with an interactive Python shell. This might be what you want, but not if your intention is to create a new script. It's not difficult to then create a new file (just open the "File" menu to "New File"). But it's also possible to have IDLE start out with an empty script instead of an interactive shell. To do this, simply pull down the "Options" menu to "Configure IDLE". Press the "General" tab and then for "At Startup" select "Open Edit Window" instead of "Open Shell Window". There are other options you can configure, but this is probably the one you will want to look at first.

Reading Data from the Keyboard

In this exercise your script should read input data from the keyboard. For each station you will want to read two numbers, the arm and the weight. To get started, the easiest way to do so is to read these one value at a time, one per line. Here is an example of how to read in two numbers and just add them:

```
x = float(input("Enter a value for x: "))
print("x=",x)
y = float(input("Enter a value for y: "))
print("y=",y)
print("x+y=", x+y)
```

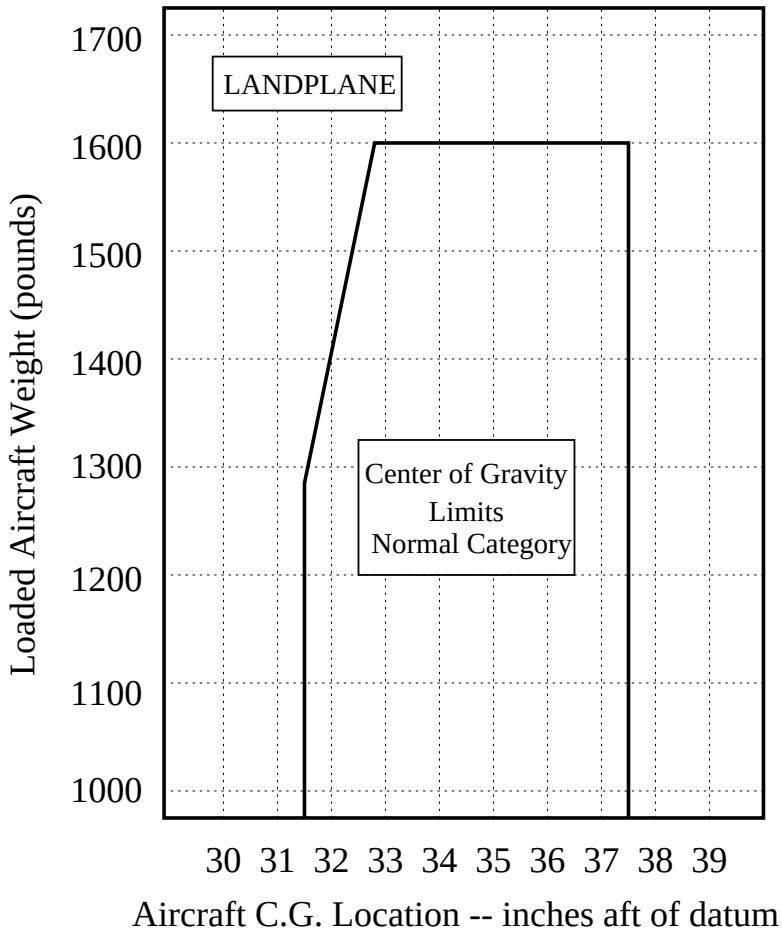

Figure 1: Weight and CG limits for a typical small aircraft.

The `input()` function prints a prompt and then waits for the user to type something (and then press the Enter key). The `input()` function then returns as its result whatever the user typed, and in this example that becomes the input to the `float()` function.

The `float()` function turns whatever input it is given into a floating-point number (a number with a possible decimal part, as opposed to an integer). If you don't use the `float()` function here, the result from the `input()` function is of a data type called a "character string," which cannot be used for mathematical calculations.

Whenever you read in numerical data values it is a good idea to print them back again, for confirmation that the computer read what you typed correctly. It is also a good idea to include text which explains what is being output. Just printing numbers with nothing else can be too cryptic.

Assignment

Your goal for this exercise is to write a Python script which will accept numerical data from the keyboard and use it to compute the gross weight and center of gravity (CG) position for a small aircraft. The weight and CG range limits for a typical small aircraft are shown in Fig. 1.

To compete this exercise you must do the following:

1. Write a new Python script which will read in exactly four arm and weight values (one value per line, first the arm, then the weight, repeated four times). Once the total weight and CG position have been computed and printed, the program should warn the user if the weight is too high, or if the CG is not in the proper range:
    a. If the weight is at *or above* the maximum allowed (1600 pounds) then print an appropriate warning message.
    b. If the CG is not *within* the proper range (31.5 inches to 37.5 inches) print an appropriate warning message.
2. Demonstrate that your code works using the data from the previous assignment. (You should of course get the same numerical values.)
3. Demonstrate that your code works by using data for which the gross weight is too high. Then demonstrate your code with data for which the CG is too far forward or too far aft.
4. Submit both your script and the demonstration output to your instructor.

Optional Improvements:

You do not have to make these improvements to your program (yet), but you can do so if you want to make it work nicely:

- Print the total weight and CG position after each pair of data points is entered. You should still print the total weight and CG position after the last data points are entered.
- Print a special warning if the weight is on *or above* the diagonal part at the left of the CG limits shown in Fig. 1. You may assume that this line passes through the points (31.5,1280) and (32.9,1600). One way to do this is to take the value computed for the CG and then calculate the weight limit given by a line which passes through those points. If the actual weight is above this value then print the warning message.