Aircraft Weight and Balance

In this exercise you will write a simple program to perform a weight and balance calculation for a small aircraft, while learning to use the "assignment" statement.

An aircraft will not fly correctly if it is over its maximum gross weight, or if the center of gravity (CG) is not within a prescribed range. It is the responsibility of the pilot in command to determine before each flight that the aircraft weight and center of gravity are within acceptable limits. Figure 1 illustrates the idea of center of gravity position for a small aircraft.

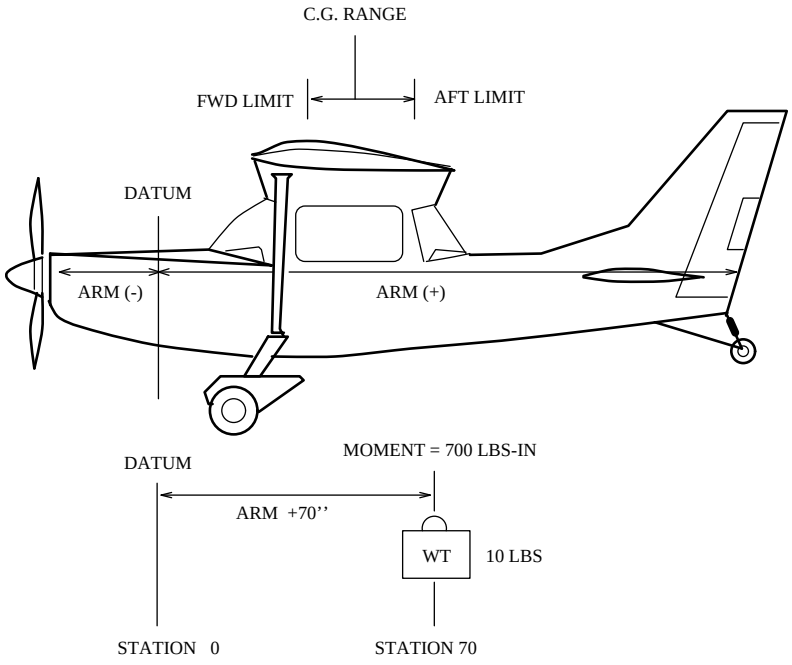

Figure 1: Weight and balance illustrated.

The gross (total) weight is simple to compute: add the empty weight of the aircraft, the weight of the pilot and passengers, the weight of any baggage, and the weight of the fuel. If w_i is the ith item out of these four things then the total weight W is

$$W = \sum_{i=1}^{4} w_i \tag{1}$$

The center of gravity is almost as simple to calculate. Each of the items mentioned above has a "moment," which is the product of the weight of the item and it's "arm," which is the distance of the object fore (-) or aft (+) from some reference point (often the firewall between the engine and the cabin). If d_i is the distance of the ith object (or "station") from the reference point, then the position of the center of gravity, d_{cg}, is the total moment divided by the total weight:

$$d_{cg} = \frac{1}{W} \sum_{i=1}^{4} (w_i \times d_i) \ . \tag{2}$$

Variables and Assignments

Calculations in most computer systems are based on some kind of "assignment" statement. The basic idea is that you calculate a result or specify a value, and then you "assign" that value or result to a variable. Just as on a chalk board, a "variable" acts as a named place-holder for the value. But on a chalk board variables usually have single letter names, like x or y or z. Most computer systems allow you to expand variable names to multiple letters, and possibly numbers and even the underscore character.

In Python variable names can be of any length, and can consist of letters, numbers, and the underscore character, provided the name does not begin with a digit. Variable names cannot have spaces, but the underscore can be used for separation. Variable names in Python are case-sensitive, which means uppercase and lowercase are different. The names `Force` and `force` are treated as separate variables. You can use mixed capitalization to make variable names more readable. For example: `NumberOfElements`.

The form of the assignment statement is that you name the variable to hold the result, then an equal sign, then a value or mathematical expression to specify what is to be stored in that variable. For example:

```
Sum = 2 + 2
rsq = x*x + y*y + z*z
```

One curious form of the assignment statement is

```
N = N + 1
```

On a chalk board or on paper this makes no sense; there is no value of N which is a solution if this is treated as an algebraic equation. But this is not an equation to be solved, it is an assignment statement. This statement means "take the value in the variable named 'N', add one to it, and then store the result back into the variable named 'N'."

Each assignment statement "assigns" a value to a variable. Another way to think of it is that variables are like lockers or cubbies which can hold your stuff. The name of the variable is the name of the locker or cubby holding something. Whenever a variable name is used in a mathematical expression, the computer uses whatever is currently stored in that location. An assignment statement puts something else in that location (and the previous contents disappear.)

Mathematical Operations

In Python, as well as many other computer languages, the mathematical operations of addition, subtraction, multiplication, and division, are specified by the characters `+`, `-`, `*`, and `/`. In Python the `**` operator is used for exponentiation, to raise the value in a variable or expression to a power. So an assignment statement to implement the famous equation $E = mc^2$ would be

```
E = m * c**2
```

Spaces in equations are ignored, but can improve readability, so the spaces around the `*` character here are not required, but may be helpful.

You can use parentheses to control the order of operation, just as you do on a chalk board. Without parentheses, the exponentiation operator `**` is applied first, then multiplication and division, then addition and subtraction.

Editing Scripts with IDLE or Thonny

To edit an existing Python script (a `.py` file) you can first open IDLE or Thonny and then pull down the "File" menu to "Open". But there is also a useful shortcut, using a right-click on the file icon. It's the same in principle on any kind of computer, but slightly different for each:

Windows: Right-click on the file icon, and pull down the menu to either "Edit with IDLE" or "Edit with Thonny". (If neither is available, try "Open with" and select one or the other.)

MacOS: Control-click on the file icon, pull down to "Open With", and select IDLE or Thonny.

Raspberry Pi: Right-click on the file icon, pull down to "Open With >" and then to "Open With...." Open the "Programming" node and select "Python 3 (IDLE)" or "Thonny". After you do this once that option will appear in the list presented whenever you right-click on a Python file and pull down to "Open With >".

Launching Python Scripts with the Mouse

The reason you have to right-click on the icon (or control-click on a Mac) to edit a Python script is that double-clicking is reserved for running the script, at least if you are using IDLE.* Though if you are using Thonny then double-clicking on the icon will simply open the file for editing, and you then press the "Play" button to run your script.

* On a Raspberry Pi you will have to configure this behavior, but it's easy to set up.

If you are using IDLE then double-clicking on the file icon for any `.py` file will run immediately; a window will pop up to show the output, and then the window will disappear when the script is done. Unfortunately the window may be gone before you can read it. To deal with this, you can add the following instruction to the very end of your scripts:

```
input("Press Enter to exit...")
```

Then the output window will pop up when you double-click on the icon, but won't disappear until you press the Enter key.

Assignment

Your goal for this exercise is to write a Python script which computes both the gross (total) weight and the location of the center of gravity (CG) of a small aircraft having four stations.

To complete this exercise you must do the following:

1. Use the following data for a small plane (a typical Cessna 150) to compute the gross weight and the position of the center of gravity:

Item	Weight (lbs)	Arm (inches)
Aircraft	1107	34.15
Pilot and passenger	340	39.12
Baggage	10	63.77
Fuel	135	42.22

2. Assign each of the numerical values in the table to a variable with a meaningful name. The names do not have to be long or use complete words, but they should be distinguishable and recognizable. Remember, underscore is an allowed character in Python variable names (though you may not need to know that for this program).
3. Then use those variables to compute the gross weight of the aircraft and the position of the center of gravity, storing each into other variables, which also have meaningful names. Use the simplest arithmetic you can. Some people find it useful to store intermediate results (like sums) in additional variables.
4. Finally, output the results. Be sure to label the two quantities correctly, and include their proper units.
5. Remember to include proper comments in your code, and remember that blank lines can be included for readability.
6. Print or save both your program and a sample of the output and give both to your instructor.