Fortifying Vermouth with Python

Dan Sharp

November 6, 2019

One of the many benefits of learning Python is developing the ability to automate solutions to everyday problems.

Recently some friends were experimenting with making vermouth and asked for help with the fortifying calculations. They wanted to bring a wine of 12.5 percent alcohol up to 16 percent by adding some spirit of 72.3 percent. This is more than a simple percentage calculation because by adding spirit not only does the percentage of alcohol rise, but also the total volume.

This type of calculation pops up all over the place in many industries and can be solved by hand using a Pearson Square or algebraically as shown below. There are also many calculators online that do this.

However, it's more interesting to build our own. We'll do the math first and then look at how we can automate it with Python.

The percentage equation is: $\frac{x}{y}(100)$ = the percentage of x in y where $y \neq 0$. We can use this as the basis of our calculator.

We'll start by listing the variables we know and don't know. We have;

1. an initial volume of wine to be fortified, v_1 with an initial percentage of alcohol which we'll call p_1.

2. an unknown volume of fortifying spirit, v_x with a known percentage of alcohol which we'll call p_2.

3. a desired final percentage which we will call p_f.

By inspection we can see that our x value will be $(\frac{P_1}{100})v_1 + (\frac{p_2}{100})v_x$ Our y value will just be $v_1 + v_x$.

Substituting these variables into the percentage equation:

$$\frac{(\frac{p_1}{100})v_1 + (\frac{p_2}{100})v_x}{v_1 + v_x}(100) = p_f \tag{1}$$

Rearranging (1) to solve for V_x gives:

$$\frac{(\frac{p_f}{100})v_1 - (\frac{p_1}{100})v_1}{(\frac{p_2}{100}) - (\frac{p_f}{100})} = v_x \tag{2}$$

So we know we can solve for v_x algebraically, but what about with Python? The great thing is we don't even need to know the algebraic method to solve this problem.

We'll start by writing a module called `fortify.py` that contains a function to calculate a value of v_x, and a second function to make sure that value makes sense.

Our first function `return v x` starts with a zero value for v_x and solves the percentage equation multiple times, incrementing v_x by 0.1 each iteration until the answer equals p_f.

This would be a very tedious way to solve a problem by hand, but it's a commonly utilised solution in computer science. It's not perfect, and if v_1 is too large we could be in for a long wait while our laptop does the numbers.

However, for our purposes, i.e volumes less than five litres it works just fine. We can always adjust the units or increments if we need to calculate larger volumes later on.

Our second function `test v x` inserts the `return v x` value into the percentage calculation and checks if it equals the final percentage we specified. If all is well then it returns the volume v_x. If, for example, a percentage value greater than 100 has been entered the user is prompted to review the inputs. This approach to testing doesn't catch all errors, a zero value for p_2 sends the program into an endless loop.

```
def return_v_x(v_1, p_1, p_2, p_f):
    """Computes v_x using percentage equation.
    All arguments should be positive floats. p_1, p_2 & p_f are percentages.
    v_1 and v_x are volumes in millilitres.
    """
    v_x = 0
    while((((((p_1/100) * v_1) + ((p_2/100) * v_x))/(v_1 + v_x)) * 100) < p_f):
        v_x += .1
        v_x = round(v_x,1)
        continue
    return v_x

def test_v_x(v_1, p_1, p_2, v_x, p_f):
    """Tests return_v_x value with percentage equation and compares with desired final percentage.
    """
    a = (((((p_1/100) * v_1) + ((p_2/100) * v_x)))/(v_1 + v_x)) * 100
    a = round(a,1)
    if a == p_f:
        print('For a final abv of %.1f percent add %.0f millilitres of the fortifying spirit for a
            total volume of %.0f millilitres.' % (a, v_x, v_x + v_1))
    else:
        print('Test failed, please review your inputs')
```

Now we've got our calculation module all that needs to be done is write the main program. This asks the user to input values for v_1, p_1, p_2 and p_f. Then we import our `fortify` module and write some explanatory comments as below.

```
"""
Created on Sat Oct 26 08:51:55 2019

@author: Dan Sharp
"""

#This is a program to calculate the volume of fortifying spirit
#to add to wine to raise it to a desired final percentage of alcohol.

import fortify

v_1 = float(input('Enter the initial volume to be fortified in millilitres:'))

p_1 = float(input('Enter the abv percentage of the inital volume:'))

p_2 = float(input('Enter the abv percentage of the fortifying spirit:'))

p_f = float(input('Enter the desired abv percentage of the final liquid:'))

v_x = fortify.return_v_x(v_1, p_1, p_2, p_f)

fortify.test_v_x(v_1, p_1, p_2, v_x, p_f)

"""
Sample run:
fortification_calculator.py
Enter the initial volume to be fortified in millilitres:750

Enter the abv percentage of the inital volume:12.5

Enter the abv percentage of the fortifying spirit:73.2

Enter the desired abv percentage of the final liquid:16.5
For a final abv of 16.5 percent add 53 millilitres of the fortifying spirit
for a total volume of 803 millilitres.
"""
```

Ten minutes work and we have our calculator. It's not idiot proof and we could have done more work to restrict the user inputs to only valid values. But the idea of using Python is to make tasks quicker and easier so we won't waste time and effort on the minor details.

So to answer our vermouth problem in the opening paragraph, as we can see in the sample run above, 53 millilitres of 73.2 percent spirit is needed to fortify 750 millilitres of vermouth to 16.5 percent.

The code in this article is based on methods explained in 'Effective Computation in Physics: Field Guide to Research with Python' by Anthony Scopatz, Kathryn D. Huff (O'Reilly, 2015) and 'A Primer on Scientific Programming with Python' by Hans Petter Langtangen (4th Ed).