## Day 1

Part 1 : Elves expense report of numbers, of which we have to find two that add to 2020, then multiple them together for the answer.

Part 2 : the same exercise but finding 3 numbers that add to 2020.

Learnt: how to [split](https://www.w3schools.com/python/ref_string_split.asp) an input (as “””string”””) up into a list {raw\_input.split(“\n”)} and treat it {expenses = [int(string) for string in split\_input]}

In functions the return result can be true or false (or any data), with the code written after the return call.

[Filter](https://www.w3schools.com/python/ref_func_filter.asp) function : filter(function, iterable)

### Day 1 improvements

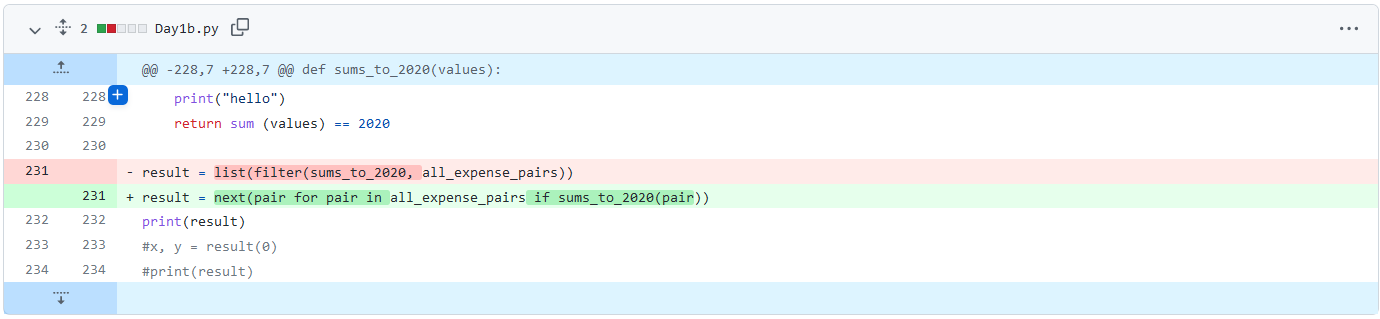
Use [**List Comprehension**](https://www.w3schools.com/python/python_lists_comprehension.asp) instead of filter()

List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list.

Example:

Based on a list of fruits, you want a new list, containing only the fruits with the letter "a" in the name.

Without list comprehension you will have to write a for statement with a conditional test inside:



newlist = [expression for item in iterable if condition == True]

**Instead of:**

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]  
newlist = []  
for x in fruits:  
  if "a" in x:  
    newlist.append(x)  
print(newlist)

**Becomes:**

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]  
newlist = [x for x in fruits if "a" in x]  
print(newlist)

eg

The condition is like a filter that only accepts the items that valuate to True.

newlist = [x for x in fruits if x != "apple"]

newlist = [x for x in fruits] (The condition is optional and can be omitted:)

newlist = [x for x in range(10)] (The iterable can be any iterable object, like a list, tuple, set etc.)

newlist = [x for x in range(10) if x < 5]

The expression is the current item in the iteration, but it is also the outcome, which you can manipulate before it ends up like a list item in the new list:

newlist = [x.upper() for x in fruits] (Makes words uppercase)

newlist = ['hello' for x in fruits]

The expression can also contain conditions, not like a filter, but as a way to manipulate the outcome:

newlist = [x if x != "banana" else "orange" for x in fruits]

The expression in the example above says:

"Return the item if it is not banana, if it is banana return orange".

## Day 2

Part 1 : Password database where a certain letter is given a minimum and maximum number of times to be a correct password.

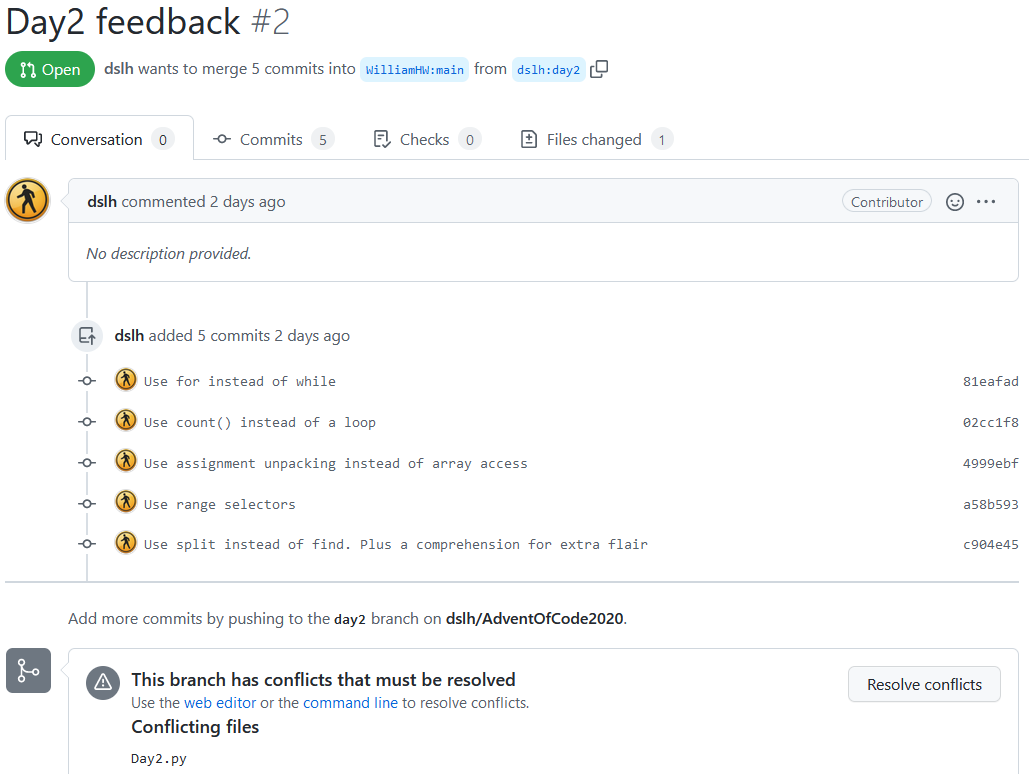
Part 2 : the instructions state 2 positions where a given letter must occur only once to be a valid password.

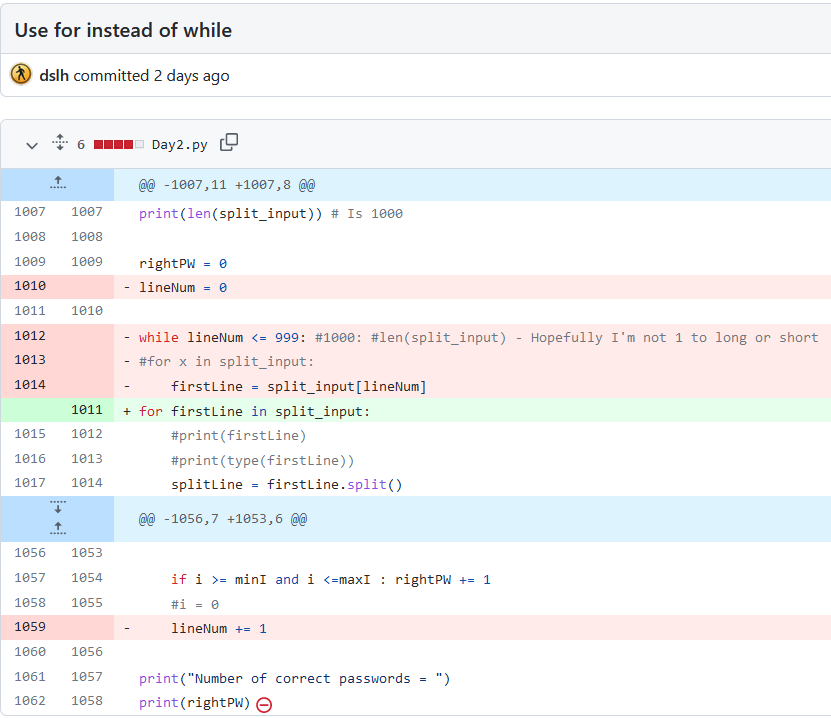
Learnt: [while](https://www.w3schools.com/python/python_while_loops.asp) function (apparently I should have chosen for)

With the while loop we can execute a set of statements as long as a condition is true.

i = 1  
while i < 6:  
  print(i)  
  i += 1  
  if i == 3:  
    continue  
else:  
  print("i is no longer less than 6")

### Day 2 improvements





[**For**](https://www.w3schools.com/python/python_for_loops.asp) **Loops**

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

The for loop does not require an indexing variable to set beforehand.

fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
  if x == "apple":  
    continue   
  if x == "banana":  
    break  
  print(x)   
else:  
  print("Finally finished!")

With the continue statement we can stop the current iteration of the loop, and continue with the next.

With the break statement we can stop the loop before it has looped through all the items.

The else keyword in a for loop specifies a block of code to be executed when the loop is finished.

**Note:** The else block will NOT be executed if the loop is stopped by a break statement.

for x in range(6):  
  if x == 3: break  
  print(x)  
else:  
  print("Finally finished!") (This line won’t show)

------------------

for x in range(2, 30, 3):  
  print(x)

----------------

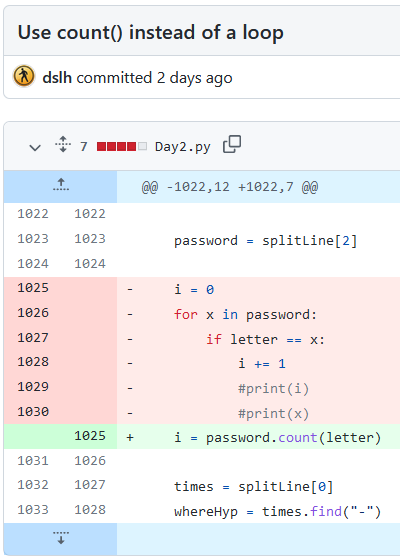
A nested loop is a loop inside a loop.

The "inner loop" will be executed one time for each iteration of the "outer loop":

adj = ["red", "big", "tasty"]  
fruits = ["apple", "banana", "cherry"]  
for x in adj:  
  for y in fruits:  
    print(x, y)

------

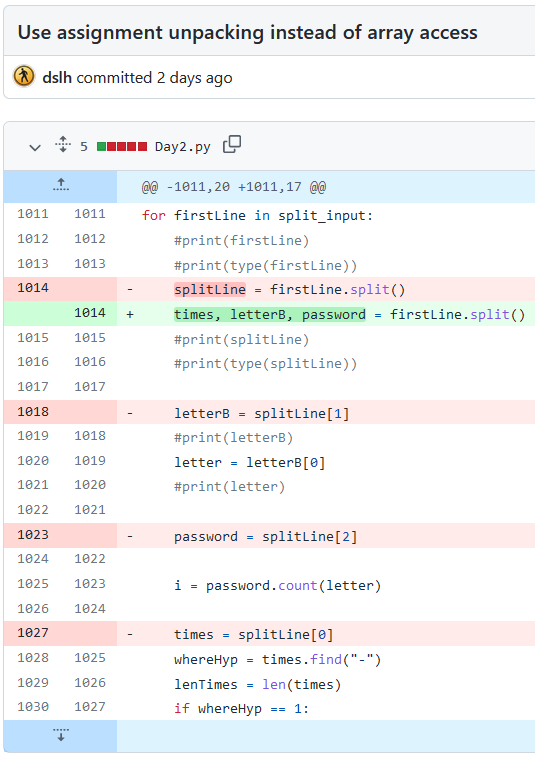
for x in [0, 1, 2]:  
  pass



Python String [count()](https://www.w3schools.com/python/ref_string_count.asp) Method

The count() method returns the number of times a specified value appears in the string.

string.count(value, start, end)



Python String [split()](https://www.w3schools.com/python/ref_string_split.asp) Method

Split a string into a list where each word is a list item:

string.split(separator, maxsplit)

But assigning the values during unpacking. See also:

Python Variables - [Assign Multiple Values](https://www.w3schools.com/python/python_variables_multiple.asp) - Python allows you to assign values to multiple variables in one line:

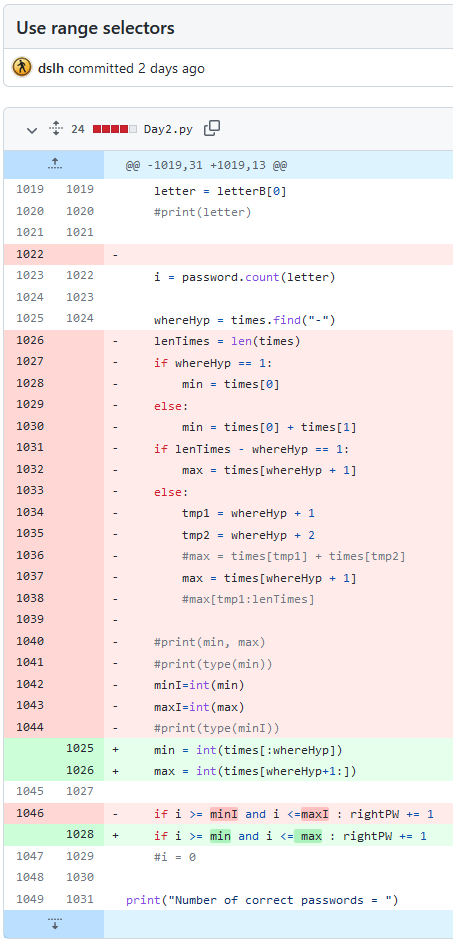
x, y, z = "Orange", "Banana", "Cherry"  
print(x)  
print(y)  
print(z)

Python - Unpack Tuples - We are also allowed to extract the values back into variables. This is called "unpacking":

fruits = ("apple", "banana", "cherry")  
(green, yellow, red) = fruits  
print(green)  
print(yellow)  
print(red)

Assign the rest of the values as a list called "red":

fruits = ("apple", "mango", "papaya", "pineapple", "cherry")  
(green, \*tropic, red) = fruits  
print(green)  
print(tropic)  
print(red) (Gives one string for apple, then a tuple (I guess it’s a tuple) of three, then a string of cherry.



**After finding the hyphen in the string with Python String** [**find()**](https://www.w3schools.com/python/ref_string_find.asp) **Method:**

The find() method finds the first occurrence of the specified value.

The find() method returns -1 if the value is not found.

The find() method is almost the same as the [index()](https://www.w3schools.com/python/ref_string_index.asp) method, the only difference is that the index() method raises an exception if the value is not found. (See example below)

string.find(value, start, end)

**Then get certain characters out of the string, cause Strings are Arrays**

Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.

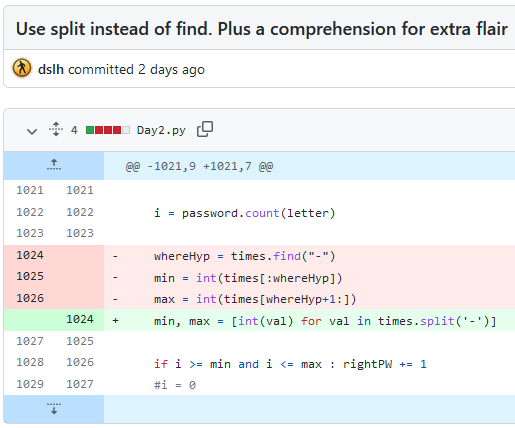
However, Python does not have a character data type, a single character is simply a string with a length of 1.

Square brackets can be used to access elements of the string.

a = "Hello, World!"

print(a[1:6]) 🡪 ello,

**Then convert the string to an integer** (as seen on Day 1 and badly in my original code).



Using [**List Comprehension**](https://www.w3schools.com/python/python_lists_comprehension.asp) and assignment unpacking in form of:newlist = [expression for item in iterable ~~if condition == True~~]

Splitting my string around the hyphen character, gives a list, which can by cycled, and directing modifying the expression to force it to be an integer.