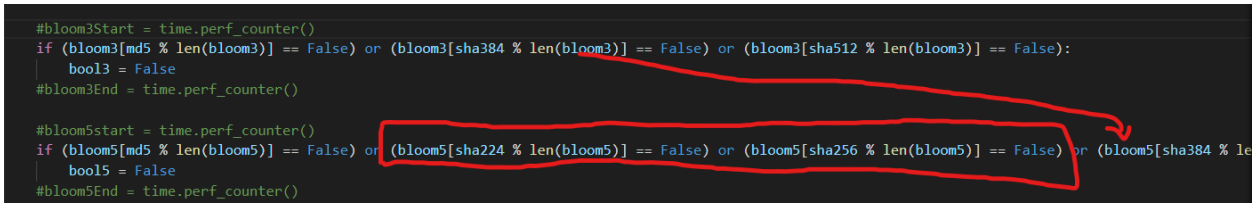
BLOOM FILTER WRITE-UP TASK - CHANCE GIGUIERE

1. For this program, I chose 5 hash functions from the python hashlib library.
   These hash functions are **MD5, SHA224, SHA256, SHA384,** and **SHA512**.
   I chose these functions because they were easy to implement into my program and because they are cryptographic.

2. The output ranges of the hash functions used are:
   MD5: 2^128 bits
   SHA-224: 2^224 bits
   SHA-256: 2^256 bits
   SHA-384: 2^384 bits
   SHA-512: 2^512 bits

3. The size of the bloom filters are calculated by the formula
   ```
   int(dictSize * log(2) / (log(2) ** hashCount * falsePositiveRate))
   ```
   Where ***dictSize*** is the number items in the dictionary used to train the bloom filter, ***hashCount*** is the number hashing operations to be used by the filter (3 or 5), and ***falsePositiveRate*** is a global variable initialized to 0.0001 (or 1 in 10,000)

4. When checking a single password, Bloom3 takes roughly .000002 seconds to decide if a password is in the bloom filter (give or take .0000003 seconds). Bloom5 is moderately faster at roughly 0.0000008 seconds (give or take .0000001 seconds). This was a surprising result to discover, because I assumed that it would take slightly longer to check a password with bloom5 compared to bloom3 since there are two additional hash values that must be checked with bloom5. One potential reason that this could be is that Bloom5 uses two additional hashing algorithms before it uses the ones present in Bloom3, so for example if sha224 returns a false result, then the result will be generated for bloom5 before bloom3.



5. The probability of false positives in my bloom filters for each case can be determined by using the formula $P = (1-[1-1/m]^{kn})^k$
   Where **P** is the probability of a false positive
   n is the number of items already in the bloom filter (i.e the size of the dictionary)
   m is the size of the Bit Array
   And **k** is the number of hashing operations.
   For both filters, **n** is 623518 (number of passwords in dictionary.txt)

For Bloom3, the probability is **2.9937925118914937e-12, or roughly 0.00000192903.** (**M** = 12977710242, **n** = 623518, **k** = 3)
For Bloom5, this is reduced to **2.047543104704747e-20**, which is significantly smaller. (**M** = 27011403544, **n** = 623518, **k** = 5)

6. I can reduce the rate of false positives by expanding the output range of my hashing functions as well as by expanding the size of my bloom filter(s)

7. If I set the size of my 5-hash bloom filter to the same size as my 3-hash filter, the False positive rate for the 5 hash filter will become **7.995437855998362e-19**
(Using the formula from #5 above, **K** = 5, **N** = 623518, and **M** = 12977710242 (The size of Bloom3)

8. **The probability of a False Negative in my Bloom Filters is 0%**. This is because it is impossible for a Bloom Filter to report a false negative (saying something isn't in the bloom filter when it actually is), but it is possible for them to give false positives, which is why we output "Maybe" and "No" rather than "Yes" and "No" when outputting the results.