Encryption - Algorithms

What is our GOAL for this CLASS?

In this class, we solved an encryption problem that would have taken up to 1 year to execute, and with logic building and algorithms, we brought down it's computation time to less than 10 seconds.

What did we ACHIEVE in the class TODAY?

- We understood the importance of logic building and algorithms
- We decrypted a secret message

Which CONCEPTS/ CODING BLOCKS did we cover today?

- Algorithms
- MD5 Hashes
- Decryption with brute force
- Permutations

How did we DO the activities?

1. Understand the boiler plate code:
    - We have the important libraries that we are going to use imported.

```
import hashlib
from itertools import permutations
```

    - We declare our encrypted message and make the **find_hash()** function call -

```
hash = '13b382e1a2f8e22535b4730d78bc8591'
answer = find_hash(hash)
print(f"Collision!  The word corresponding to the given hash is '{answer}'")
```

    - Inside the **find_hash()** function, we open the word list, declare our anagram - **“who outlay thieves”** and count the total number of words our answer should have. We also took out all the list of unique characters our anagram has with the help of **list()** and **set()** functions.

```
def find_hash(original_hash):
    word_file = open("words.txt","r")
    word_file = list(word_file)

    anagram = "who outlay thieves"
    words = anagram.count(' ')
    words += 1

    char_list = list(set(anagram))

    if ' ' in char_list:
        char_list.remove(' ')

    final_words = []

    #Student Activity
```


- We then have a loop that uses the **permutations()** function that takes the unique combinations of 3 words, joins them together with a space, encrypts the phrase with md5 encryption and finally checks if the encryption matches our secret message.

```
for elem in permutations(final_words, words):
    hash_elem = " ".join(elem)

    #Student Activity

    m = hashlib.md5()
    m.update(hash_elem.encode('utf-8'))
    word_hash = m.hexdigest()

    if word_hash == original_hash:
        return hash_elem
```

2. With 7,779 words in the dictionary, there are a total of 470.5 billion possibilities out of which, one is the answer. The total computation time to solve that could be up to 1 year, so to reduce the time, we remove all the unnecessary words by checking if the word has any character that the anagram doesn't. For this -

    - We iterate over all the words
    - Create a flag that will be **True** if there is any character in the word that is not in the anagram, and remain **False** if all the characters of the words are in the anagram as well.
    - Take out a list of all the characters in that word with **list()** and **set()** functions.
    - See if there is any character that is not in the anagram.
    - Change the flag to **True** if there is any such character
    - Append the word into the final list accordingly

```
for i in word_file:
    flag = False
    temp_word = i.replace('\n', '')
    temp_char = list(set(temp_word))
    for i in temp_char:
        if i not in char_list:
            flag = True
            break
    if flag == False:
        final_words.append(temp_word)

print(len(final_words))
```

3. With this, we should be left with only 194 words. Now, we can apply another algorithm that would check the length of the sentence generated by the **permutations()** function and compare it with the anagram's length. After all, the length of the **permutations()** generated phrase should be exactly the same as that of the anagram for it to be a potential answer -

```
for elem in permutations(final_words, words):
    hash_elem = " ".join(elem)

    if len(hash_elem) != len(anagram):
        continue

    m = hashlib.md5()
    m.update(hash_elem.encode('utf-8'))
    word_hash = m.hexdigest()

    if word_hash == original_hash:
        return hash_elem
```

4. Run the program with the following command in terminal/command prompt and see the message -
    a. **python main.py**

```
Collision!  The word corresponding to the given hash is 'whitehat loves you'
```

What's NEXT?

In the next class, we will learn about some basics of how viruses are built and how they work.

Expand Your Knowledge

To know more about click here