XOR Decryption

Paul Gribble

2024-03-25

The Exercise: XOR Decryption

Each character on a computer is assigned a unique code and the preferred standard is ASCII (American Standard Code for Information Interchange). For example, uppercase `A` = `65`, asterisk `*` = `42`, and lowercase `k` = `107`.

A modern encryption method is to take a text file, convert the bytes to ASCII, then XOR each byte with a given value, taken from a secret key. The advantage with the XOR function is that using the same encryption key on the cipher text, restores the plain text; for example, `65 XOR 42` = `107`, then `107 XOR 42` = `65`.

For unbreakable encryption, the key is the same length as the plain text message, and the key is made up of random bytes. The user would keep the encrypted message and the encryption key in different locations, and without both "halves", it is impossible to decrypt the message.

Unfortunately, this method is impractical for most users, so the modified method is to use a password as a key. If the password is shorter than the message, which is likely, the key is repeated cyclically throughout the message. The balance for this method is using a sufficiently long password key for security, but short enough to be memorable.

Your task has been made easy, as the encryption key consists of **three lower case characters**.

Using 0059_cipher.txt, a file containing the encrypted ASCII codes, and the knowledge that the plain text must contain common English words, decrypt the message. Then find the sum of the ASCII values in the decrypted text.

Your program should output the following:

```
The encryption key is: XXX
The sum of the ASCII values in the original text is: YYY
```

where `XXX` is the 3-character encryption key and `YYY` is the sum of the ASCII values in the original text.

—from Project Euler

Hints

- in Python the ^ operator can be used to perform bitwise XOR on integers
- in Python the `ord()` function can be used to get the ASCII value of a character, e.g. `ord('A')` returns `65` and `ord('a')` returns `97`
- in Python the `chr()` function can be used to get the character from an integer ASCII value, e.g. `chr(65)` returns `'A'` and `chr(97)` returns `'a'`

A Sample Solution

The Code

```
import time

# open the file and load in all integers into a list
#
with open("0059_cipher.txt","r") as fid:
    ciphertext = fid.readlines()[0].split(',')

print(f"ciphertext is {len(ciphertext)} chars long")
print(f"the first 10 are {ciphertext[:10]}")

# define function to encrypt/decrypt a message using a key with XOR
#
def crypt(message, key):
    """
    crypt(message, key) applies XOR bitwise to each char in message (int)
                        using the chars in key (str) and returns the (de/en)crypted
                        message (str); assumes key is shorter than message
    """
    message2 = "" # string starts out empty
    for i in range(len(message)):
        message2 += chr( int(message[i]) ^ ord(key[i % len(key)]) )
    return message2
```

```
# loop through all possible keys and decrypt
#
letters = "abcdefghijklmnopqrstuvwxyz" # we know the key is lower-case letters only
n = 1
nn = 26*26*26
solutions = []
t1 = time.time()
for c1 in letters:
    for c2 in letters:
        for c3 in letters:
            print(f"\r{n}/{nn}", end=" ")
            n = n + 1
            key = c1 + c2 + c3
            message2 = crypt(ciphertext, key)
            ok = message2.find(' the ') >= 0
            if ok:
                solutions.append({"n": n, "key": key, "message": message2})
t2 = time.time()
print(f"\nelapsed time: {t2-t1:.3f} sec")
print(f"found {len(solutions)} potential solution(s)")

# look at all the potential solutions
#
for i in range(len(solutions)):
    n = solutions[i]["n"]
    key = solutions[i]["key"]
    message = solutions[i]["message"]
    print(f"n: {n}")
    print(f"key: {key}")
    print(f"message: {message}")
    print("---")
```

The Output:

```
ciphertext is 1455 chars long
the first 10 are ['36', '22', '80', '0', '0', '4', '23', '25', '19', '17']
17576/17576
elapsed time: 5.276 sec
found 1 potential solution(s)
n: 3319
```

```
key: exp
message: An extract taken from the introduction of one of Euler's most celebrated ...
```

The decrypted message in full is:

```
An extract taken from the introduction of one of Euler's most celebrated papers, "De summis serierum reciprocarum" [On the sums of series of reciprocals]: I have recently found, quite unexpectedly, an elegant expression for the entire sum of this series 1 + 1/4 + 1/9 + 1/16 + etc., which depends on the quadrature of the circle, so that if the true sum of this series is obtained, from it at once the quadrature of the circle follows. Namely, I have found that the sum of this series is a sixth part of the square of the perimeter of the circle whose diameter is 1; or by putting the sum of this series equal to s, it has the ratio sqrt(6) multiplied by s to 1 of the perimeter to the diameter. I will soon show that the sum of this series to be approximately 1.644934066842264364; and from multiplying this number by six, and then taking the square root, the number 3.141592653589793238 is indeed produced, which expresses the perimeter of a circle whose diameter is 1. Following again the same steps by which I had arrived at this sum, I have discovered that the sum of the series 1 + 1/16 + 1/81 + 1/256 + 1/625 + etc. also depends on the quadrature of the circle. Namely, the sum of this multiplied by 90 gives the biquadrate (fourth power) of the circumference of the perimeter of a circle whose diameter is 1. And by similar reasoning I have likewise been able to determine the sums of the subsequent series in which the exponents are even numbers.
```

And the 3-character key found to decrypt it was: `exp`