Stolen Sheets - Alan Yan

We assume the cryptographic scheme is an affine map. That is, for every character, we do the maps

$$Char \rightarrow f \rightarrow Ax + B \text{ in } Z/26Z \rightarrow f^{-1} \rightarrow new\ Char$$

where f is the map $a \rightarrow 0,\ b \rightarrow 1,\ c \rightarrow 2,\ \ldots$

My code is below.

```
hiddentext = "mrkcdpmsuimzstqrlg"
alphabet = 'abcdefghijklmnopqrstuvwxyz'

LIST = ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight',
'nine', 'ten']
cipher = ['qzc', 'xwq', 'xbpcc', 'tqop', 'tsfc', 'gsn', 'gcfcz', 'cskbx',
'zszc', 'xcz']

# converts array of integer to array of string
def A(n):
      ret = ""
      for i in n:
            ret += alphabet[i]
      return ret

# convert array of string to array of integer
def B(n):
      ret = []
      for i in n:
            ret.append(alphabet.index(i))
      return ret

# affine transform array of integers
def C(n, a, b):
      ret = []
      for i in n:
            ret.append((a * i + b) % 26)
      return ret;

#decode assuming choiceA and choiceB
def decode(cipherArray, bestArray, choiceA, choiceB):
      plaintext = []
      for i in cipherArray:
            plaintext.append(A(C(B(i), choiceA, choiceB)))
      if (plaintext == bestArray):
            return True
      return False

# output the correct flag
def solve(word, a, b):
      print(A(C(B(word), a, b)))
```

```
#brute force search
for i in range(0, 26):
      for j in range(0, 26):
            if(decode(cipher, LIST, i, j)):
                  a = i
                  b = j

solve(hiddentext, a, b)
```