

In [1]:

```

import math
key = "megabuck"           }= chave
sorted_key = sorted(key)    }= chave ordenada
texto = "pleasetransferonemilliondollarstomyswissbankaccountsixtwotwo" }= plaintext
colunas = len(key)          } A quantidade de colunas é dado pelo tamanho da KEY.
linhas = math.ceil(len(texto)/len(sorted_key)) } A quantidade de linhas é o arredondamento
alfabeto = "abcdefghijklmnopqrstuvwxyz"      } superior da divisão do tamanho do texto
                                         } pelo tamanho da chave.

```

*Adicionando characteren  
caso necessario:*

In [2]:

```

temp = 0
while len(texto) != (linhas*colunas):
    texto = texto + alfabeto[temp]
    temp += 1

matriz = [["0"] * colunas for _ in range(linhas)] } Inicialização
temp = 0
x = 0
y = 0
for letra in texto:
    if y == colunas:
        x += 1
        y = 0
    matriz[x][y] = letra
    y += 1
} Alocando os characteren nas suas  
respectivas posições.

```

In [3]:

```

for lista in matriz:
    print(lista)

```

In [4]:

```

temp = 0
texto_criptografado = ""
for i in range(colunas):
    for j in range(linhas):
        pos = key.index(sorted_key[temp]) } A posição das colunas é dado
        texto_criptografado += matriz[j][pos] } pelo index das letras do sorted_key.
        print(matriz[j][pos], end = '')
    temp += 1

```

*Criptografia*

Em virtude de eu  
não saber como ocorria  
a criptografia, cheguei  
a conclusão que para  
a criptografia ser  
o tamanho da  
chave é um  
múltiplo do tamanho do  
texto criptografado.

```

texto_criptografado = "aflksoselawaiatoossctlnmomantesilyntwrnntsowdpaedobuoeriricxb" } resultado
multiplos = []
for i in range(2, math.ceil(len(texto_criptografado)/2) + 1, 1):
    if len(texto_criptografado) % i == 0:
        multiplos.append(i)

print(multiplos)

```

[2, 4, 8, 16, 32]

```
In [6]: from itertools import permutations

def gerar_permutacoes(lista):
    return list(permutations(lista))
```

} gerador de permutações

```
In [8]: from langdetect import detect
for i in multiplos:
    pedacos = []
    x = 0
    y = i
    for j in range(int(len(texto_criptografado)/i)):
        pedacos.append(texto_criptografado[x:y])
        x += i
        y += i
    print(f"Há {len(pedacos)} pedaços => {pedacos}\n =====")
    matriz = [[0] * len(pedacos[0]) for _ in range(int(len(texto_criptografado)/i))]
    linha = 0
    coluna = 0
    """
    if len(pedacos) <= 8:
        permutacoes = gerar_permutacoes(pedacos)
        print(len(permutoes))
    """
    =====
```

Há 32 pedaços => ['af', 'll', 'sk', 'so', 'se', 'la', 'wa', 'ia', 'to', 'os', 'sc', 'tc', 'ln', 'mo', 'ma', 'nt', 'es', 'il', 'yn', 'tw', 'rn', 'nt', 'so', 'wd', 'pa', 'ed', 'ob', 'uo', 'er', 'ir', 'ic', 'xb']

Há 16 pedaços => ['afll', 'skso', 'sel', 'waia', 'toos', 'sctc', 'lnmo', 'mant', 'esil', 'yntw', 'rnnt', 'sowd', 'paed', 'obuo', 'erir', 'icxb']

Há 8 pedaços => ['afllskso', 'selawaia', 'toossctc', 'lnmomant', 'esilyntw', 'rnntso', 'wd', 'paedobuo', 'eriricxb']

Há 4 pedaços => ['afllskso selawaia', 'toossctc lnmomant', 'esilyntw rnntsowd', 'paedob uoeriricxb']

Há 2 pedaços => ['afllskso selawaia toossctc lnmomant', 'esilyntw rnntsowd paedob uoeriricxb']

Considerações Finais: Não consegui quebrar a cifra, tentei por método exaustivo gerando todas as permutações, porém logo percebi que para chaves grandes o problema já ficaria inviável por ter  $n!$ . Limitei o tamanho da chave para apenas 8 caracteres, caso do exemplo, e analisei apenas os 8 primeiros caracteres da permutação e verificava se a concatenação destes 8 caracteres formava uma palavra válida, e foi formada a palavra chave: "planet", mas ela ocorria  $1/8!$  e também poderia ter gerado outras palavras válidas como "tear", "ate", "ect", o que confundia a biblioteca que identificava a língua falada no texto.