100 - Simple cipher - Crypto

> *I got an encrypted message and a file I used for encryption. However I do not know what to do, so I want you to solve it instead.*

enc_text.txt = `0c157e2b7f7b515e075b391f143200080a00050316322b272e0d525017562e73183e3a0d564f6718`

And encryption.py looks like:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

mes = "*****secret*****"
key = "J2msBeG8"

# padding with spaces
if len(mes) % len(key) != 0:
    n = len(key) - len(mes) % len(key)
    for i in range(n):
        mes += " "

m = []
for a in range(len(key)):
    i = a
    for b in range(len(mes)/len(key)):
        m.append(ord(mes[i]) ^ ord(key[a]))
        i += len(key)

enc_mes = ""
for j in range(len(m)):
    enc_mes += "%02x" % m[j]

print enc_mes
```

It's a non linear xoring.

Example: message length is 16, key length is 8 (so 2 key loops).

Normal xoring gives: m[0] ^ k[0], m[1] ^ k[1], m[2] ^ k[2], m[3] ^ k[3], m[4] ^ k[4], m[5] ^ k[5], m[6] ^ k[6], m[7] ^ k[7], m[8] ^ k[0], m[9] ^ k[1], m[10] ^ k[2], m[11] ^ k[3], m[12] ^ k[4], m[13] ^ k[5], m[14] ^ k[6], m[15] ^ k[7]

The modified xoring gives: m[0] ^ k[0], m[8] ^ k[0], m[1] ^ k[1], m[9] ^ k[1], m[2] ^ k[2], m[10] ^ k[2], m[3] ^ k[3], m[11] ^ k[3], m[4] ^ k[4], m[12] ^ k[4], m[5] ^ k[5], m[13] ^ k[5], m[6] ^ k[6], m[14] ^ k[6], m[7] ^ k[7], m[15] ^ k[7]

In our case: message length is 40, key length is 8 (so 5 key loops).

So I wrote the python lines that does exactly the reverse process, we can test with the default message:

```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

key = "J2msBeG8"
mes = "*****secret*****"

# padding with spaces
if len(mes) % len(key) != 0:
    n = len(key) - len(mes) % len(key)
    for i in range(n):
        mes += " "

m = []
for a in range(len(key)):
    i = a
    for b in range(len(mes)/len(key)):
        m.append(ord(mes[i]) ^ ord(key[a]))
        i += len(key)

enc_mes = ""
for j in range(len(m)):
    enc_mes += "%02x" % m[j]

print enc_mes

#enc_mes = "0c157e2b7f7b515e075b391f143200080a00050316322b272e0d525017562e73183e3a0d564f6718"

enc_mes_splited = [enc_mes[i:i + 2] for i in range(0, len(enc_mes), 2)]

for j in range(len(enc_mes_splited)):
    enc_mes_splited[j] = int(enc_mes_splited[j], 16)

m = enc_mes_splited
mes = ""

for b in range(len(enc_mes_splited)/len(key)):
    for a in range(len(key)):
        mes += str(chr((m[len(enc_mes_splited)/len(key)*a+b]) ^ ord(key[a])))
print mes
```

And we find the original message:

```
$ python2 encryption.py
60381857471959596868164f226d5b12
*****secret*****
```

So now let's replace `enc_mess` with the challenge value and we find `FIT{Thi5_cryp74n4lysi5_wa5_very_5impl3}` .