# xorbitant

How encryption is done (enc.py)
The program reads a plaintext file (plaintext.txt).
It XORs every byte with a repeating key, which is the FLAG environment variable (or "CTF{example_flag}" by default).
The key repeats every len(key) bytes.
The result is written to out.bin.
Key length is known to be 69.

solver.py

```
from collections import Counter
import argparse
from typing import ByteString

SPACE = 0x20
DEFAULT_KEYLEN = 69

def recover_key(ciphertext: ByteString, keylen: int) -> bytearray:
    key = bytearray(keylen)
    for r in range(keylen):
        # slice every keylen-th byte beginning at offset r
        slice_bytes = ciphertext[r::keylen]
        if not slice_bytes:
            raise ValueError(f"No bytes found for key position {r} (keylen
too large?)")
        most_common_byte, _ = Counter(slice_bytes).most_common(1)[0]
        key[r] = most_common_byte ^ SPACE
    return key

def main():
    parser = argparse.ArgumentParser(description="Recover repeating-key XOR
assuming SPACE is most common.")
    parser.add_argument("-i", "--input", default="out.bin", help="ciphertext
file (default: out.bin)")
    parser.add_argument("-k", "--keylen", type=int, default=DEFAULT_KEYLEN,
help=f"key length (default: {DEFAULT_KEYLEN})")
    args = parser.parse_args()

    try:
        with open(args.input, "rb") as f:
            ct = f.read()
    except FileNotFoundError:
        print(f"Error: cannot open file '{args.input}'")
        return
```

```
    try:
        key = recover_key(ct, args.keylen)
        flag = key.decode("ascii", errors="strict")
    except Exception as e:
        print("Failed to recover key:", e)
        return

    print(flag)

if __name__ == "__main__":
    main()
```