BITS = ('0', '1')

ASCII\_BITS = 8

def bit\_list\_to\_string(b):

    """converts list of {0, 1}\* to string"""

    return ''.join([BITS[e] for e in b])

def seq\_to\_bits(seq):

    return [0 if b == '0' else 1 for b in seq]

def pad\_bits(bits, pad):

    """pads seq with leading 0s up to length pad"""

    assert len(bits) <= pad

    return [0] \* (pad - len(bits)) + bits

def convert\_to\_bits(n):

    """converts an integer `n` to bit array"""

    result = []

    if n == 0:

        return [0]

    while n > 0:

        result = [(n % 2)] + result

        n = n / 2

    return result

def string\_to\_bits(s):

    def chr\_to\_bit(c):

        return pad\_bits(convert\_to\_bits(ord(c)), ASCII\_BITS)

    return [b for group in

            map(chr\_to\_bit, s)

            for b in group]

def bits\_to\_char(b):

    assert len(b) == ASCII\_BITS

    value = 0

    for e in b:

        value = (value \* 2) + e

    return chr(value)

def list\_to\_string(p):

    return ''.join(p)

def bits\_to\_string(b):

    return ''.join([bits\_to\_char(b[i:i + ASCII\_BITS])

        for i in range(0, len(b), ASCII\_BITS)])

inputString = "attack at dawn"

numberOutput = int(bit\_list\_to\_string(string\_to\_bits(inputString)),2) #1976620216402300889624482718775150

bitSeq = seq\_to\_bits(bin(numberOutput)[2:]) #[2:] is needed to get rid of 0b in front

paddedString = pad\_bits(bitSeq,len(bitSeq) + (8 - (len(bitSeq) % 8))) #Need to pad because conversion from dec to bin throws away MSB's

outputString = bits\_to\_string(paddedString) #attack at dawn