Project 3:
The One-Time Pad

CSCI 360

March 22, 2017

Due on Monday, April 3.

1 The Main Idea

The goal of this project is to implement the One-Time Pad. Recall that the One-Time Pad is carried out as followed:

- `KeyGen`: The key generation algorithm randomly generates a key K as a binary string of length n.
- `Enc`: Represent the message m as a binary string of length n. Encrypt via the bitwise XOR operation,
$$c = K \oplus m.$$
- `Dec` Decrypt by computing $m = K \oplus c$.

You are provided with functions which convert from text to binary, and vice-versa.

2 Challenge 1: KeyGen

Write a function which generates the key for the OTP. This key should be randomly generated. For this, you must use the `random` module. The function should have input n, the desired key length, and output an n-bit random binary key.

```
INPUT: integer n
OUTPUT: n-bit binary string

Example: Input 3,  output 110
         Input 5,  output 10110
         Input 10, output 0011000101
```

Following is optional psuedocode

```
Import the random module at the beginning of your code.

def KeyGen(n):
    Initialize a key to an empty string

    For each item in range(n):
        Append a random bit to the key

    Return the key
```

3 Challenge 2: Encrypt

Next write a function which performs encryption. The input should be a bitstring of length n and a key (also a bitstring of length n), and the output should be an bitstring of length n. Encryption is carried out by performing bitwise XOR.

```
INPUT: key (n-bit string), plaintext (n-bit string)
OUTPUT: ciphertext (n-bit string)

Example:
Key        1011101
Plaintext  0010100

Output     1001001
```

And the following optional psuedocode:

```
def encrypt(key, plaintext):
    Initialize the ciphertext as an empty string

    for i in range(len(plaintext)):
        Compute plaintext[i] XOR key[i]
        Append this value to the ciphertext

    return the ciphertext
```

3.1 How to Compute XOR

Note that computing XOR is equivalent to performing computations modulo 2.

0 XOR 0 = 0 corresponds to $0 + 0 \equiv 0 \pmod{2}$, or (0 + 0)%2

0 XOR 1 = 1 corresponds to $0 + 1 \equiv 1 \pmod{2}$, or (0 + 1)%2

1 XOR 0 = 1 corresponds to $1 + 0 \equiv 1 \pmod{2}$, or (1 + 0)%2

1 XOR 1 = 0 corresponds to $1 + 1 \equiv 0 \pmod{2}$, or (1 + 1)%2

4 Challenge 3: Decrypt

The decryption process works the same as the encryption process, except now you XOR the ciphertext with the key to retrieve the plaintext.

5 Challenge 4: Application

Write an application in the main function which performs the following:

(1) Ask the user for a plaintext message.

(2) Convert the plaintext message to binary.

(3) Generate a random key with the same length as the binary plaintext message.

(4) Encrypt.

(5) Convert the ciphertext to text and output.

An example run of the program is as follows:

```
What would you like to encrypt? ABC
Your ciphertext is 7!x
```

6 Challenge 5: Decryption

Extend your application. Ask the user whether they would like to encrypt or decrypt, and perform the requested function. You receive 5 points on take home exam 1.