Today's Exercise: security and privacy and building an encryption module!

- First understand the problem and design a solution

- Next implement an "application" in Python
  - Application: You want your photograph(s) to be viewable only by authorized people

Security& Privacy Exercise: Encryption

- Encryption – coding your mesages
  - Sending secrets
  - Safeguard your private information!
- Caeser's Cipher – a simple 'substitution cipher' algorithm
  - History: used by Julius Caeser to send military secrets
- Original Form: Shift each alphabet by 3
  - A replaced by D, B replaced by E,…..Y replaced by B
  - Circular shift
  - "FRIDAY" encrypted as " IULGDB "

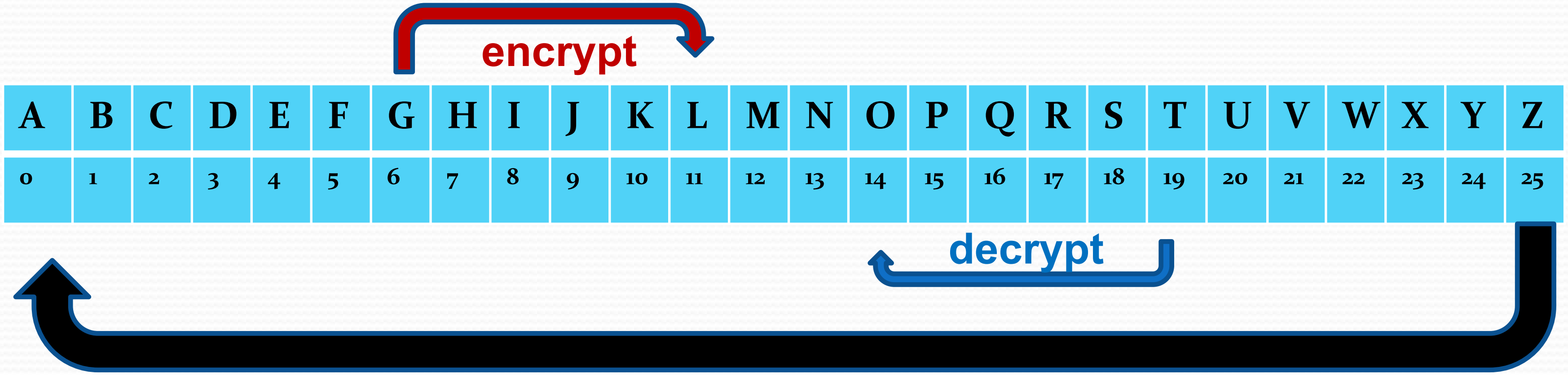
Generalized Shift(Caeser) Cipher.

- Instead of shifting by 3, shifted by some secret value K
  - K is between 0 and 25
    - Why ? Because there are 26 letters in the alphabet
- The value K is your secret "Key" (like a password)
- Encryption "algorithm" : Shift each letter by K
- To "decrypt" the message: Shift 'left' each letter by K
- Some math: we can assign a number from 0 to 25 to each letter in the alphabet starting with A
  - Shifting by K means adding K to that number
  - But circular addition…more in a bit

Example:

Message: **GOODBYE**
Key: K = 5

Encrypted message:
LTTIGDJ
(G replaced by L, O by T, …)



To decrypt the encrypted message, move letter left 5 places

So what's the "math" behind this..

- Algorithms need to be shown to be "correct"….
- This is where the math comes in !

Some Math…the CS "discrete" math:
Circular Addition uses *Modulo arithmetic*:

(A+K) mod N = remainder of (A+K) divided by N

Ex: (6+5) mod 26 =11 *(letter L)*,

(24+5) mod 26 = 3 *(which is letter D)*

To decrypt: (B –K) mod N

If (B-K) is negative it adds N to get result.

(3 -5) mod 26 = -2+26 = 24 = letter Y

Modulo arithmetic in Python

- Circular addition
  - Circular addition…..known as Modulo
    - A Mod N = remainder of A divided by N
- Good news: Python provides the Modulo operation
  - **B = a % N**

- To encrypt value a with key K: B = (a+K) mod N
  - For alphabet N=26 (we have 26 different values)

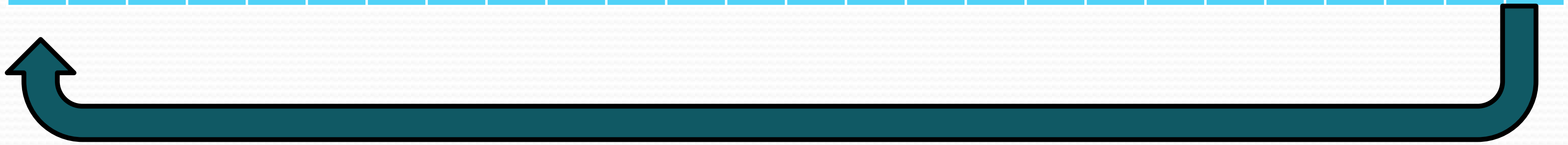
Weak encryption vs Strong encryption

- Strength of encryption = How easy is it to decipher your secret (i.e., encryption)
- In Caeser’s cipher we use the same key for each character in our message
  - Shift each alphabet by 5
- Another method: version of One-Time-Pad (OTP)
- Encrypt each position in message with a separate key
  - Message = BYE
  - Shift B by 3, shift Y by 7, shift E by 5 to get EFJ

Your first exercise….in breakout groups

- Encryption 1 using Caeser's cipher (circular shift):
  - Each of you chooses a key
  - Each chooses a day of the week (Monday through Sunday) and encrypts the day.
  - ONE of you shares your encrypted message (day of the week)
  - Can others in the group guess the message ?
- Encryption 2: using One time pad
  - Choose one of these words: Monday, Sunday, Friday
  - One of you encrypts the word they chose using on-time pad (a unique key for each position in the 6 letter word)
  - Share your encrypted message
  - Can the others guess (in one guess) what exactly the word is ?

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25



An application using Encryption & implementation in Python today….

- You want to send a picture (your selfie) to a friend
  - Or better yet, post it on a website
- To restrict who can see it, you want to encrypt it and only those with the correct key will be able to see the picture
- Steps:
  1. Take your selfie
  2. Import into your Python code and enter a secret Key
  3. Write (and run) python code to encrypt the selfie
     - Implement the encryption algorithm we discussed
  4. Decrypt with the key – a wrong key will lead to a jumbled image
- Checking your encryption: Look at the encrypted image and see how similar it looks to the original image
  - The less similar it looks the "better" the encryption!

Getting Started…some preliminaries

- An image (i.e., your selfie) is a matrix of pixels
- To simplify our algorithm (for purpose of demonstration!) we convert your image to a grayscale image
- input image is a N by M matrix A[i,j] of pixels and key=K
  - Each pixel A[i,j] has a greyscale value between 0 and 255
    - i.e., 256 different values – analogy with 26 letters in alphabet
- To encrypt image, for each pixel add K to A[i,j] to get B[i,j]
  - Important: Circular addition with 256 different values
  - Python operator: %
    - B[i,j] = (A[i,j] + K) % 256

A better encryption "system"

- We saw how the one-time pad (OTP) is a better technique
- Applying the concept to this system: each pixel A[i,j] has its own key K[i,j]
  - And then algorithm is B[i,j] = (A[i,j] + K[i,j]) % 256
- Here is a cool trick: instead of entering gazilion values of K[i,j], how about using a 'secret' image as your key ?!!
  - Key image K, represented as a matrix K[i,j]
- Change to algorithm:
  - import the key image as K[i,j] convert to greyscale
  - B[i,j] = (A[i,j] + K[i,j]) % 256