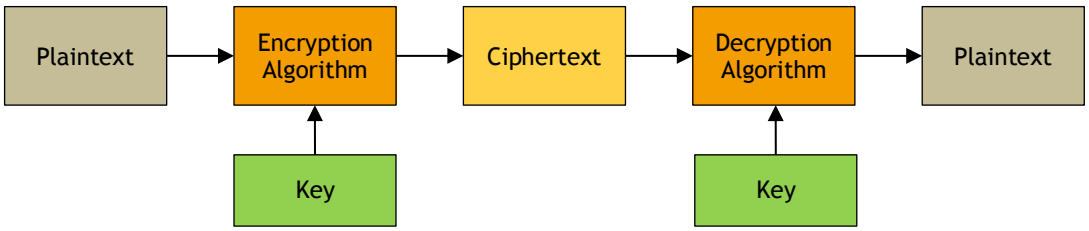
Section 3.5, Miller 3rd ed. Substitution Cipher

Encryption: Substitute each character with a different character

Decryption: Reverse the substitution

Use a **key** to tell which letter is substituted for which.

Alphabet: All possible chars the plaintext can have
Key: Must be a ***permutation*** of the alphabet



Small Toy Example

Alphabet: `'abcdef'`
Key: `'dabfce'`
Plaintext: `'bed'`

Alphabet	a	b	c	d	e	f
key	d	a	b	f	c	e
	0	1	2	3	4	5

Encryption:

Plaintext	Index i in alphabet	Ciphertext: key[i]
b	1	a
e	4	c
d	3	f

Ciphertext: `'acf'`

Decryption:

Ciphertext	Index i in key	Decrypted: Alphabet[i]
a	1	b
c	4	e
f	3	d

Decrypted: `'bed'`

See 03-04-substitution-cipher-small-demo.py

Turn these ideas into functions

See 03-05-substitution-cipher-small-with-functions.py

Creating a Random Key

Generate random permutation of alphabet with length n:

```
key = ''
for i in range(n):
     select char ch at random from the alphabet
     add ch to the key
     remove ch from the alphabet
```

See 03-06-substitution-cipher-small-with-random-key.py

Creating a Key from a Password

Password can only have characters from the alphabet

1. Remove all duplicate chars from password
2. `before` = all alphabet chars **before** last char of password
3. `after` = all alphabet chars **after** last char of password
4. remove password chars from `before` & **`after`**
5. Concatenate the following:
    - `key = password + after + before`

Example

```
alphabet = 'abcdef'
pw = 'bedebceed'
```

1. Remove dups from PW
   `pw = 'bedc'`

2. Make before (alphabet letters before 'c')
   `before = 'ab'`

3. Make after (alphabet letters after 'c')
   `after = 'def'`

4. Remove PW chars from before and after
   ```
   before = 'a'
   after = 'f'
   ```

5. Concatenate pw + after + before
   ```
   key  =      pw      +    after    +   before
        =    'bedc'    +     'f'     +    'a'
        =   'bedcfa'
   ```