Python Cryptography Tutorial

Contents

Time required: 60 minutes

Khan Academy Computer Science Theory Videos

- What is cryptography?
- Ciphers vs. Codes
- Shift cipher
- The Caesar cipher
- Caesar Cipher Exploration
- Frequency Fingerprint Exploration
- Polyalphabetic cipher

You can continue through the rest of the videos in these lessons if you wish. They are short and interesting.

Substitution Cipher

Substitution of single letters separately—simple substitution—can be demonstrated by writing out the alphabet in some order to represent the substitution. This is termed a substitution alphabet. The cipher alphabet may be shifted or reversed (creating the Caesar and Atbash ciphers, respectively) or scrambled in a more complex fashion, in which case it

is called a mixed alphabet or deranged alphabet. Traditionally, mixed alphabets may be created by first writing out a keyword, removing repeated letters in it, then writing all the remaining letters in the alphabet in the usual order.

Using this system, the keyword "zebras" gives us the following alphabets:

Plaintext alphabet	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ciphertext alphabet	ZEBRASCDFGHIJKLMNOPQTUVWXY

The most common method of substitution replaces the 26 letters of the alphabet (one letter matches only one other)

A simple and very old method of sending secret messages is the substitution cipher. Each letter of the alphabet gets replaced by another letter of the alphabet, say every **a** gets replaced with an **x**, and every **b** gets replaced by a **z**, etc.

Tutorial 1: Substitution Cipher

Create a Python program named **substitution_cipher.py**

You can copy and paste the following code to save some time.

```
# The normal alphabet representing the characters typed in
# A space is included at the end of each string
alphabet = "abcdefghijklmnopqrstuvwxyz "
# The cipher key used to scramble the message
cipher_key = "xznlwebgjhqdyvtkfuompciasr "
```

```
"""
    Name: substitution_cipher.py
    Author:
    Created:
    Substitute letters in a key for the alphabet
    to create a secret message
"""


def main():

    # The normal alphabet representing the characters typed in
    # there is a space at the end of each for converting spaces
    alphabet = "abcdefghijklmnopqrstuvwxyz "

    # The cipher key used to encrypt and decrypt the message
    cipher_key = "xznlwebgjhqdyvtkfuompciasr "

    # Get the message from the user, convert it to lower case
    secret_message = input("Enter your secret message: ")
    secret_message = secret_message.lower()
```

```
#---------------------- ENCRYPT MESSAGE -----------------------------#
    encrypted_message = ""
    # Go through the string one character at a time
    for character in secret_message:

        if character.isalpha() or character.isspace():
            # If the character is a letter or a space
            # Find the index number of the character
            index = alphabet.index(character)

            # Use the index number to find the corresponding
            # cipher key character and append it to the
            # encrypted_message string , scrambling the message
            encrypted_message = encrypted_message + cipher_key[index]
        else:
            # If the character is not a letter, leave it alone
            # Add it to the string as is
            encrypted_message = encrypted_message + cipher_key[index]

    print(f"Encrypted message: {encrypted_message}")
    # Time to decode our secret message
    input("\nPress Enter to decode your secret message ")

#---------------------- DECRYPT MESSAGE -----------------------------#
    original_message = ""
    # Go through the string one character at a time
    for character in encrypted_message:

        if character.isalpha() or character.isspace():
            # If the character is a letter or a space
            # Find the index number of the character
            index = cipher_key.index(character)

            # Use the index number to get the corresponding
            # alphabet key character, unscrambling the message
            original_message = original_message + alphabet[index]
        else:
            # If the character is not a letter, leave it alone
            original_message = original_message + alphabet[index]
    print(f"Decrypted message: {original_message}")


# Call the main function
if __name__ == "__main__":
    main()
```

Example run:

```
Enter your secret message: This is a secret message
Encrypted message: mgjo jo x ownuwm ywooxbw

Press Enter to decode your secret message
Decrypted message: this is a secret message
```

Explanation

The string **cipher_key** is a random reordering of the alphabet.

The tricky part of the program is the for loop.

1. The loop goes through the message one character at a time. For every letter it finds, it replaces it with the corresponding letter from the key.
2. This substitution is accomplished by using the **index()** method to find the position in the alphabet of the current letter and replacing that letter with the letter from the key at that position.
3. All non-letter characters are copied as is. The program uses the **isalpha()** method to tell whether the current character is a letter or not. It uses the **isapace()** method to find spaces.
4. The code to decipher a message is nearly the same. Change the key[alphabet.index(c)] to alphabet[key.index(c)].

Assignment 1: Add spaces

A challenge. Your boss at the NSA wants you to change the spaces in the messages to a different character.

Challenge accepted!

Tutorial 2: Shared Key Cryptography

Cryptography is a Python module that helps encrypt and decrypt data.

1. **pip install cryptography**
2. Create a Python program named **cyptography_shared_key.py**
3. Enter the following code:

```python
"""
    Name: cryptography_shared_key.py
    Author:
    Created:
    Shared Key Cryptography
"""
# pip install cryptography
from cryptography.fernet import Fernet

# Generate binary shared key
key = Fernet.generate_key()

print(f"Shared Key: {key}")

# Value of key is assigned to a Fernet variable
cipher_suite = Fernet(key)

plain_text = "This example demonstrates the cryptography module"
print(plain_text)

# Plaintext is encoded to binary
# It is then encrypted using the shared encryption key
cipher_text = cipher_suite.encrypt(str.encode(plain_text))

# Display the ciphertext
print(f"Encrypted ciphertext: {cipher_text.decode()}")

# Decrypt the ciphertext
decrypted_text = cipher_suite.decrypt(cipher_text)

# Cisplay the plaintext and the decode() method
# Converts from byte to string
print(decrypted_text.decode())
```

Example run:

```
Shared Key: b'5lSF-xI8MphMzyGjklynz7McphzXnInG9eHInC_UbRA='
This example demonstrates the cryptography module
Encrypted ciphertext: gAAAAABiZgGtdF66UmRJlJCURc-PlkfoKNBp2sQteyLDUGD00z-wkD0aDD
uAgu8WIlGVixRiIV0TVmfX33mbe_6vvtdN_QrdlVaSyxjZ2JSOef0S_sYypq0n22huIan3WzHe8WNLGJ
XKDxqTMvhOW2mWTYr8oDVLhg==
This example demonstrates the cryptography module
```

Assignment Submission

1. Attach the code.
2. Attach a screenshot showing a successful run of the program.
3. Submit the assignment in Blackboard.