**Introduction**

Have you ever received a PDF or an image file from someone via email, only to see strange characters when you open it? This can happen if your email server was only designed to handle text data. Files with binary data, bytes that represent non-text information like images, can be easily corrupted when being transferred and processed to text-only systems.

Base64 encoding allows us to convert bytes containing binary or text data to [ASCII](https://en.wikipedia.org/wiki/ASCII) characters. By encoding our data, we improve the chances of it being processed correctly by various systems.

In this tutorial, we would learn how Base64 encoding and decoding works, and how it can be used. We will then use Python to Base64 encode and decode both text and binary data.

* [What Is Base64 Encoding?](https://stackabuse.com/encoding-and-decoding-base64-strings-in-python/#whatisbase64encoding)
* [How Does Base64 Encoding Work?](https://stackabuse.com/encoding-and-decoding-base64-strings-in-python/#howdoesbase64endcodingwork)
* [Why Use Base64 Encoding?](https://stackabuse.com/encoding-and-decoding-base64-strings-in-python/#whyusebase64encoding)
* [Encoding Strings with Python](https://stackabuse.com/encoding-and-decoding-base64-strings-in-python/#encodingstringswithpython)
* [Decoding Strings with Python](https://stackabuse.com/encoding-and-decoding-base64-strings-in-python/#decodingstringswithpython)
* [Encoding Binary Data with Python](https://stackabuse.com/encoding-and-decoding-base64-strings-in-python/#encodingbinarydatawithpython)
* [Decoding Binary Data with Python](https://stackabuse.com/encoding-and-decoding-base64-strings-in-python/#decodingbinarydatawithpython)
* [Conclusion](https://stackabuse.com/encoding-and-decoding-base64-strings-in-python/#conclusion)

**What is Base64 Encoding?**

Base64 encoding is a type of conversion of bytes into ASCII characters. In mathematics, the base of a number system refers to how many different characters represent numbers. The name of this encoding comes directly from the mathematical definition of bases - we have 64 characters that represent numbers.

The Base64 character set contains:

* 26 uppercase letters
* 26 lowercase letters
* 10 numbers
* + and / for new lines (some implementations may use different characters)

When the computer converts Base64 characters to binary, each Base64 character represents 6 bits of information.

**Note:** This is not an encryption algorithm, and should not be used for security purposes.

Now that we know what Base64 encoding and how it is represented on a computer, let's look deeper into how it works.

**How Does Base64 Encoding Work?**

We will illustrate how Base64 encoding works by converting text data, as it's more standard than the various binary formats to choose from. If we were to Base64 encode a string we would follow these steps:

1. Take the ASCII value of each character in the string
2. Calculate the 8-bit binary equivalent of the ASCII values
3. Convert the 8-bit chunks into chunks of 6 bits by simply re-grouping the digits
4. Convert the 6-bit binary groups to their respective decimal values.
5. Using a base64 encoding table, assign the respective base64 character for each decimal value.

Let's see how it works by converting the string "Python" to a Base64 string.

The ASCII values of the characters P, y, t, h, o, n are 15, 50, 45, 33, 40, 39 respectively. We can represent these ASCII values in 8-bit binary as follows:

01010000 01111001 01110100 01101000 01101111 01101110

Recall that Base64 characters only represent 6 bits of data. We now re-group the 8-bit binary sequences into chunks of 6 bits. The resultant binary will look like this:

010100 000111 100101 110100 011010 000110 111101 101110

**Note:** Sometimes we are not able to group the data into sequences of 6 bits. If that occurs, we have to [pad the sequence](https://en.wikipedia.org/wiki/Base64#Output_padding).

With our data in groups of 6 bits, we can obtain the decimal value for each group. Using our last result, we get the following decimal values:

20 7 37 52 26 6 61 46

Finally, we will convert these decimals into the appropriate Base64 character using the Base64 conversion table:

As you can see, the value 20 corresponds to the letter U. Then we look at 7 and observe it's mapped to H. Continuing this lookup for all decimal values, we can determine that "Python" is represented as UHl0aG9u when Base64 encoded. You can verify this result with an online converter.

To Base64 encode a string, we convert it to binary sequences, then to decimal sequences, and finally, use a lookup table to get a string of ASCII characters. With that deeper understanding of how it works, let's look at why would we Base64 encode our data.

**Why use Base64 Encoding?**

In computers, all data of different types are transmitted as 1s and 0s. However, some communication channels and applications are not able to understand all the bits it receives. This is because the meaning of a sequence of 1s and 0s is dependent on the type of data it represents. For example, 10110001 must be processed differently if it represents a letter or an image.

To work around this limitation, you can encode your data to text, improving the chances of it being transmitted and processed correctly. Base64 is a popular method to get binary data into ASCII characters, which is widely understood by the majority of networks and applications.

A common real-world scenario where Base64 encoding is heavily used are in mail servers. They were originally built to handle text data, but we also expect them to send images and other media with a message. In those cases, your media data would be Base64 encoded when it is being sent. It will then be Base64 decoded when it is received so an application can use it. So, for example, the image in the HTML might look like this:

<img src="...">

Understanding that data sometimes need to be sent as text so it won't be corrupted, let's look at how we can use Python to Base64 encoded and decode data.

**Encoding Strings with Python**

Python 3 provides a base64 module that allows us to easily encode and decode information. We first convert the string into a bytes-like object. Once converted, we can use the base64 module to encode it.

In a new file encoding\_text.py, enter the following:

import base64

message = "Python is fun"

message\_bytes = message.encode('ascii')

base64\_bytes = base64.b64encode(message\_bytes)

base64\_message = base64\_bytes.decode('ascii')

print(base64\_message)

In the code above, we first imported the base64 module. The message variable stores our input string to be encoded. We convert that to a bytes-like object using the string's encode method and store it in message\_bytes. We then Base64 encode message\_bytes and store the result in base64\_bytes using the base64.b64encode method. We finally get the string representation of the Base64 conversion by decoding the base64\_bytes as ASCII.

**Subscribe to our Newsletter**

Top of Form

Get occassional tutorials, guides, and jobs in your inbox. No spam ever. Unsubscribe at any time.

Newsletter Signup

Subscribe

Bottom of Form

**Note:** Be sure to use the same encoding format to when converting from string to bytes, and from bytes to string. This prevents data corruption.

Running this file would provide the following output:

$ python3 encoding\_text.py

UHl0aG9uIGlzIGZ1bg==

Now let's see how we can decode a Base64 string to its raw representation.

**Decoding Strings with Python**

Decoding a Base64 string is essentially a reverse of the encoding process. We decode the Base64 string into bytes of unencoded data. We then convert the bytes-like object into a string.

In a new file called decoding\_text.py, write the following code:

import base64

base64\_message = 'UHl0aG9uIGlzIGZ1bg=='

base64\_bytes = base64\_message.encode('ascii')

message\_bytes = base64.b64decode(base64\_bytes)

message = message\_bytes.decode('ascii')

print(message)

Once again, we need the base64 module imported. We then encode our message into a bytes-like object with encode('ASCII'). We continue by calling the base64.b64decode method to decode the base64\_bytes into our message\_bytes variable. Finally, we decode message\_bytes into a string object message, so it becomes human readable.

Run this file to see the following output:

$ python3 decoding\_text.py

Python is fun

Now that we can encode and decode string data, let's try to encode binary data.

**Encoding Binary Data with Python**

As we mentioned previously, Base64 encoding is primarily used to represent binary data as text. In Python, we need to read the binary file, and Base64 encode its bytes so we can generate its encoded string.

Let's see how we can encode this image:

Create a new file encoding\_binary.py and add the following:

import base64

with open('logo.png', 'rb') as binary\_file:

binary\_file\_data = binary\_file.read()

base64\_encoded\_data = base64.b64encode(binary\_file\_data)

base64\_message = base64\_encoded\_data.decode('utf-8')

print(base64\_message)

Let's go over the code snippet above. We open the file using open('my\_image.png', 'rb'). Note how we passed the 'rb' argument along with the file path - this tells Python that we are reading a binary file. Without using 'rb', Python would assume we are reading a text file.

We then use the read() method to get all the data in the file into the binary\_file\_data variable. Similar to how we treated strings, we Base64 encoded the bytes with base64.b64encode and then used the decode('utf-8') on base64\_encoded\_data to get the Base64 encoded data using human-readable characters.

Executing the code will produce similar output to:

$ python3 encoding\_binary.py



Your output may vary depending on the image you've chosen to encode.

Now that we know how to Bas64 encode binary data in Python, let's move on Base64 decoding binary data.

**Decoding Binary Data with Python**

Base64 decoding binary is similar to Base64 decoding text data. The key difference is that after we Base64 decode the string, we save the data as a binary file instead of a string.

Let's see how to Base64 decode binary data in practice by creating a new file called decoding\_binary.py. Type the following code into the Python file:

import base64

base64\_img = 'iVBORw0KGgoAAAANSUhEUgAAABAAAAAQCAYAAAAf8/9hAAAACXBIWXMAAAsTAAA' \

'LEwEAmpwYAAAB1klEQVQ4jY2TTUhUURTHf+fy/HrjhNEX2KRGiyIXg8xgSURuokX' \

'LxFW0qDTaSQupkHirthK0qF0WQQQR0UCbwCQyw8KCiDbShEYLJQdmpsk3895p4aS' \

'v92ass7pcfv/zP+fcc4U6kXKe2pTY3tjSUHjtnFgB0VqchC/SY8/293S23f+6VEj' \

'9KKwCoPDNIJdmr598GOZNJKNWTic7tqb27WwNuuwGvVWrAit84fsmMzE1P1+1TiK' \

'MVKvYUjdBvzPZXCwXzyhyWNBgVYkgrIow09VJMznpyebWE+Tdn9cEroBSc1JVPS+' \

'6moh5Xyjj65vEgBxafGzWetTh+rr1eE/c/TMYg8hlAOvI6JP4KmwLgJ4qD0TIbli' \

'TB+sunjkbeLekKsZ6Zc8V027aBRoBRHVoduDiSypmGFG7CrcBEyDHA0ZNfNphC0D' \

'6amYa6ANw3YbWD4Pn3oIc+EdL36V3od0A+MaMAXmA8x2Zyn+IQeQeBDfRcUw3B+2' \

'PxwZ/EdtTDpCPQLMh9TKx0k3pXipEVlknsf5KoNzGyOe1sz8nvYtTQT6yyvTjIax' \

'smHGB9pFx4n3jIEfDePQvCIrnn0J4B/gA5J4XcRfu4JZuRAw3C51OtOjM3l2bMb8' \

'Br5eXCsT/w/EAAAAASUVORK5CYII='

base64\_img\_bytes = base64\_img.encode('utf-8')

with open('decoded\_image.png', 'wb') as file\_to\_save:

decoded\_image\_data = base64.decodebytes(base64\_img\_bytes)

file\_to\_save.write(decoded\_image\_data)

In the above code, we first convert our Base64 string data into a bytes-like object that can be decoded. When you are base64 decoding a binary file, you **must know the type of data that is being decoded**. For example, this data is only valid as a PNG file and not a MP3 file as it encodes an image.

Once the destination file is open, we Base64 decode the data with base64.decodebytes, a different method from base64.b64decode that was used with strings. This method should be used to decode binary data. Finally, we write the decoded data to a file.

In the same directory that you executed decoding\_binary.py, you would now see a new decoded\_image.png file that contains the original image that was encoded earlier.

**Conclusion**

Base64 encoding is a popular technique to convert data in different binary formats to a string of ASCII characters. This is useful when transmitting data to networks or applications that cannot process raw binary data but would readily handle text.

With Python, we can use the base64 module to Base64 encode and decode text and binary data.

What applications would you use to encode and decode Base64 data?