Cf.: https://code.tutsplus.com/tutorials/base64-encoding-and-decoding-using-python--cms-25588



Code › Python

# Base64 Encoding and Decoding Using Python

Abder-Rahman Ali Last updated Apr 11, 2022 | Read Time: 9 min

Python | Programming Fundamentals | OOP

Say you have a binary image file you wanted to transfer across a network.

You're amazed that the file wasn't received properly on the other side—the file just contained strange characters!

Well, it seems that you attempted to send your file in its raw bits and bytes format, while the media used was designed for streaming text.

What would be the workaround to avoid such an issue? The answer is Base64 encoding. In this article, I will show you how we can use Python to encode and decode a binary image. The program is illustrated as a standalone local program, but you can apply the concept to different applications like sending your encoded image from your mobile device to a server, and many other applications.

## What Is Base64?

Before moving more deeper in the article, let's define what we mean by Base64.

Base64 is a way in which 8-bit binary data is encoded into a format that can be represented in 6 bits. This is done using only the characters `A-Z`, `a-z`, `0-9`, `+`, and `/` in order to represent data, with `=` used to pad data. For instance, using this encoding, three 8-bit bytes are converted into four 6-bit groups.

The term Base64 is taken from the Multipurpose Internet Mail Extensions (MIME) standard, which is widely used for HTTP and XML, and was originally developed for encoding email attachments for transmission.

## Why Do We Use Base64?

Base64 is very important for binary data representation, such that it allows binary data to be represented in a way that looks and acts as plain text, which makes it more reliable to be stored in databases, sent in emails, or used in text-based format such as XML. Base64 is basically used for representing data in an ASCII string format.

As mentioned in the introduction of this article, without Base64 sometimes data will not be readable at all.

# Base64 Encoding

Base64 encoding is the process of converting binary data into a limited character set of 64 characters. As shown in the first section, those characters are `A-Z`, `a-z`, `0-9`, `+`, and `/` (count them, did you notice they add up to 64?). This character set is considered the most common character set, and is referred to as MIME's Base64. It uses `A-Z`, `a-z`, and `0-9` for the first 62 values, and `+` and `/` for the last two values.

The Base64 encoded data ends up being longer than the original data, so that as mentioned above, for every 3 bytes of binary data, there are at least 4 bytes of Base64 encoded data. This is due to the fact that we are squeezing the data into a smaller set of characters.

Have you ever seen part of a raw email file like the one shown below (which most likely originates from an email not being delivered)? If so, then you have seen Base64 encoding in action! (If you notice `=` at the end, you can conclude that this is Base64 encoding, since the equals sign is used in the encoding process for padding.)

```
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: base64

2KfZhNiz2YTYp9mFINi52YTZitmD2YUg2YjYsdit2YXYqSDYp9mE2YTZhyDZiNio2LHZ
h9iMDQoNCtij2YjYryDZgdmC2Lcg2KfZhNin2LPYqtmB2LPYp9ixINi52YYg2KfZhNmF2
2KfYqiDYp9mE2K/Ysdin2LPZitipINin2YTYqtmKINiq2YbYtdit2YjZhiDYqNmH2Kcg2
INmK2LHZitivINin2YTYqtmI2LPYuSDZgdmKDQrYt9mE2Kgg2KfZhNi52YTZhSDYp9mE2
2YrYjCDYudmE2YXYpyDYqNij2YbZiiDYutmK2LEg2YXYqtiu2LXYtSDYqNin2YTYudmE2
hNi02LHYudmKINmI2KPZgdiq2YLYryDZhNmE2YXZhtmH2Kwg2KfZhNi52YTZhdmKDQrZh
gy4NCg0K2KzYstin2YPZhSDYp9mE2YTZhyDYrtmK2LHYpyDYudmE2Ykg2YbYtdit2YPZh
INmH2LDYpyDYp9mE2LTYo9mGLg0KDQrYudio2K/Yp9mE2LHYrdmF2YYNCg==
```

```
--089e0141aa264e929a0514593016
Content-Type: text/html; charset=UTF-8
Content-Transfer-Encoding: base64
```

Base64 is carried out in multiple steps, as follows:

- The text to be encoded in converted into its respective decimal values, that is, into their ASCII equivalent (i.e. a:97, b:98, etc.). Here's the ASCII table.
- The decimal values obtained in the above step are converted into their binary equivalents (i.e. 97: 01100001).
- All the binary equivalents are concatenated, obtaining a large set of binary numbers.
- The large set of binary numbers is divided into equal sections, with each section containing only 6 bits.
- The equal sets of 6 bits are converted into their decimal equivalents.
- Finally, the decimal equivalents are converted into their Base64 values (i.e. 4: E). Here are the decimal values and their Base64 alphabet.

# Base64 Decoding

Base64 decoding is the opposite of Base64 encoding. In other words, it is carried out by reversing the steps described in the previous section.

So the steps of Base64 decoding can be described as follows:

- Each character in the string is changed to its Base64 decimal value.
- The decimal values obtained are converted into their binary equivalents.
- The first two bits of the binary numbers are truncated from each of the binary numbers obtained, and the sets of 6 bits are combined, forming one large string of binary digits.
- The large string of binary digits obtained in the previous step is split into groups of 8 bits.
- The 8-bit binary numbers are converted into their decimal equivalents.
- Finally, the decimal values obtained are converted into their ASCII

equivalent.



## Base64 Encoding and Decoding of a String

It will become easier for you to understand how all this works once you see what is going on behind the scenes. Let's try to encode and decode a simple three-letter word, `Hey`.

We begin by converting each letter of the word into its ASCII equivalent, and then converting the ASCII equivalent into binary. This gives us the following values:

| Letter | ASCII Index Value | 8-bit Binary Value |
|---|---|---|
| H | 72 | 01001000 |
| e | 101 | 01100101 |
| y | 121 | 01111001 |

In other words, we can write `Hey` in binary like this:

> `01001000 01100101 01111001`

There are a total of 24 bits, which when turned into groups of 6 bits, each result in four values:

> `010010 000110 010101 111001`

In a Base64 table, the characters `A` to `Z` are represented by the values **0** to **25**. The characters `a` to `z` are represented by the values **26** to **51**. The numbers `0` to `9` are represented by the values **52** to **61**. The characters `+` and `/` are represented by **62** and **63**. The character `=` is used for padding when the the bits can't be properly divided into groups of 6.

We will now convert our rearranged bits into numerical values and then get the character that represents those numerical values.

| 6-bit Binary Value | Base64 Index Value | Letter |
|---|---|---|
| 010010 | 18 | S |
| 000110 | 6 | G |
| 010101 | 21 | V |
| 111001 | 57 | 5 |

Based on our calculations above, the letter `Hey` will become `SGV5` when Base64 encoded. We can test if that is correct using the following code:

```
from base64 import b64encode

text_binary = b'Hey'

# SGV5
print(b64encode(text_binary))
```

The whole process is done in reverse to get back our original data after Base64 decoding.

Now, I will quickly show you the encoding of another word, `Heyo`, to explain the occurrence of `=` in the encoded string.

| Letter | ASCII Index Value | 8-bit Binary Value |
|---|---|---|
| H | 72 | 01001000 |
| e | 101 | 01100101 |
| y | 121 | 01111001 |
| o | 111 | 01101111 |

There are a total of 32 bits. This will give us five different groups of 6 bits, with two leftover bits: `11`. We pad them with `0000` to get a 6-bit group. Making groups of 6 bits from the above arrangement will give you the following:

> `010010 000110 010101 111001 011011 110000`

The rearranged bits will give you back the following characters based on the Base64 index values.

| 6-bit Binary Value | Base64 Index Value | Letter |
|---|---|---|
| 010010 | 18 | S |
| 000110 | 6 | G |
| 010101 | 21 | V |
| 111001 | 57 | 5 |
| 011011 | 27 | b |

| | | |
|---|---|---|
| 110000 | 48 | w |

This means that our Base64 encoded value for `Heyo` would be `SGV5bw==`. Each `=` represents one pair of `00`s that we added for padding the original bit-sequence.

```
from base64 import b64encode

text_binary = b'Heyo'

# SGV5bw==
print(b64encode(text_binary))
```

## Base64 Encoding an Image

Let's now get to the meat of this article. In this section, I'm going to show you how we can easily Base64 encode an image using Python.

I will be using the following binary image. Go ahead and download it, and let's get Python rolling! (I'm assuming that the name of the image is **deer.gif**.)



The first thing we have to do in order to use Base64 in Python is to import the base64 module:

```
import base64
```

In order to encode the image, we simply use the function `base64.b64encode(s)`. Python describes the function as follows:

> Encode the bytes-like object `s` using Base64 and return the encoded bytes.

Thus, we can do the following in order to Base64 encode our image:

```
import base64
image = open('deer.gif', 'rb') #open binary file in read mode
image_read = image.read()
image_64_encode = base64.b64encode(image_read)
```

If you want to see the output of the encoding process, type the following:

`print image_64_encode`

## Base64 Decoding an Image

To decode an image using Python, we simply use the `base64.b64decode(s)` function. Python mentions the following regarding this function:

> Decode the Base64 encoded bytes-like object or ASCII string s and return the decoded bytes.

So, in order to decode the image we encoded in the previous section, we do the following:

`base64.decode(image_64_encode)`

# Putting It All Together

Let's put the program for Base64 encoding and decoding an image together. The Python script that does that should look something like the following:

```
import base64
image = open('deer.gif', 'rb')
image_read = image.read()
image_64_encode = base64.b64encode(image_read)
image_64_decode = base64.b64decode(image_64_encode)
image_result = open('deer_decode.gif', 'wb') # create a writable image
image_result.write(image_64_decode)
```

If you open **deer_decode.gif**, which you have on your desktop, you will notice that you have the original image **deer.gif** we encoded in the first step.

As we have seen from this article, Python makes it very easy to perform what seems to be a complex task.

# URL-Safe Encoding and Decoding

As I mentioned earlier in the tutorial, Base64 encoding also uses the characters `+` and `/` besides regular alphanumeric values. However, these characters have a special meaning within URLs. This means that a Base64 encoded value that uses these characters can result in unexpected behavior if it is used inside URLs.

One solution for this problem is to use the `urlsafe_base64encode()` and `urlsafe_base64decode()` functions to encode and decode any data. These functions replace `+` with `-` and `/` with `_` during encoding.

Here is an example in Python that shows this difference:

```
import base64

image = open('dot.jpg', 'rb')
image_data = image.read()

unsafe_encode = base64.b64encode(image_data)
safe_encode = base64.urlsafe_b64encode(image_data)

# b'/9j/4QAYRXhpZgAASUkqAAgAAAAAAAAAAAAAP/sABFEdWNr....
print(unsafe_encode)

# b'_9j_4QAYRXhpZgAASUkqAAgAAAAAAAAAAAAAP_sABFEdWNr....
print(safe_encode)
```

## Learn Python

Learn Python with our complete Python tutorial guide, whether you're just getting started or you're a seasoned coder looking to learn new skills.

*This post has been updated with contributions from Nitish Kumar. Nitish is a web developer with experience in creating eCommerce websites on various platforms. He spends his free time working on personal projects that make his everyday life easier or taking long evening walks with friends.*

Sign up


## Abder-Rahman Ali

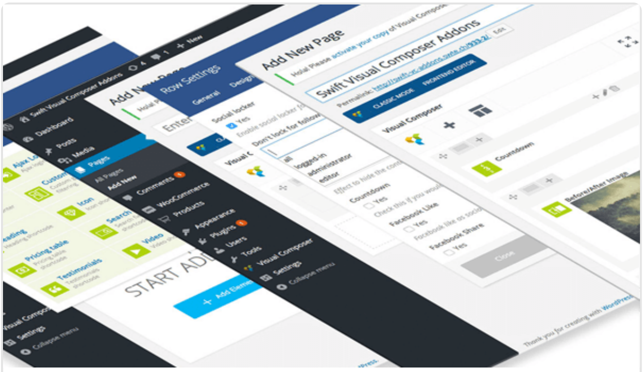
Dr. Aber-Rahman Ali is a researcher who uses machine learning and image processing in medical image analysis.

He also likes writing about Python!

abderhasan

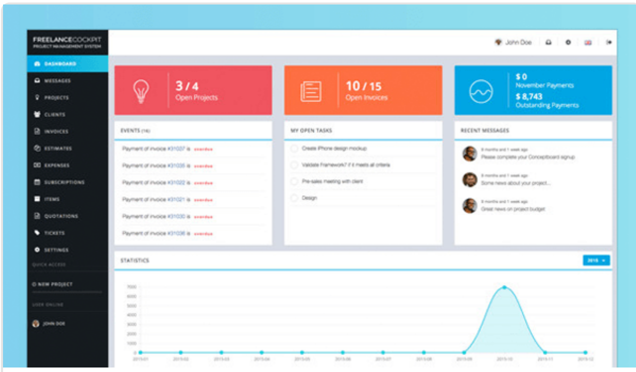
LOOKING FOR SOMETHING TO HELP KICK START YOUR NEXT PROJECT?

Envato Market has a range of items for sale to help get you started.



**WordPress Plugins**

From $5



**PHP Scripts**

From $5

**JavaScript**

From $3

---

**QUICK LINKS** - Explore popular categories