The base64 Module

Python 2.x 2.4–2.7 , Python 3.x 3.0–3.6

Base 64 encoding represents a common scheme for encoding binary into ASCII string format using radix 64. The base64 module is part of the standard library, which means it installs along with Python. Understanding of bytes and strings is critical to this topic and can be reviewed here . This topic explains how to use the various features and number bases of the base64 module.

Examples

Encoding and Decoding ASCII85

Adobe created it's own encoding called **ASCII85** which is similar to Base85, but has its differences. This encoding is used frequently in Adobe PDF files. These functions were released in Python version 3.4. Otherwise, the functions base64.a85encode() and base64.a85encode() are similar to the previous:

```
import base64
# Creating a string
s = "Hello World!"
# Encoding the string into bytes
b = s.encode("UTF-8")
# ASCII85 Encode the bytes
e = base64.a85encode(b)
# Decoding the ASCII85 bytes to string
s1 = e.decode("UTF-8")
# Printing ASCII85 encoded string
print("ASCII85 Encoded:", s1)
# Encoding the ASCII85 encoded string into bytes
b1 = s1.encode("UTF-8")
# Decoding the ASCII85 bytes
d = base64.a85decode(b1)
# Decoding the bytes to string
s2 = d.decode("UTF-8")
print(s2)
```

This outputs the following:

```
ASCII85 Encoded: 87cURD]i,"Ebo80
Hello World!
```

Encoding and Decoding Base16

The base64 module also includes encoding and decoding functions for Base16. Base 16 is most commonly referred to as **hexadecimal** . These functions are very similar to the both the Base64 and Base32 functions:

```
import base64
# Creating a string
s = "Hello World!"
# Encoding the string into bytes
b = s.encode("UTF-8")
# Base16 Encode the bytes
e = base64.b16encode(b)
# Decoding the Base16 bytes to string
s1 = e.decode("UTF-8")
# Printing Base16 encoded string
print("Base16 Encoded:", s1)
# Encoding the Base16 encoded string into bytes
b1 = s1.encode("UTF-8")
# Decoding the Base16 bytes
d = base64.b16decode(b1)
# Decoding the bytes to string
s2 = d.decode("UTF-8")
print(s2)
```

This would produce the following output:

```
Base16 Encoded: 48656C6C6F20576F726C6421
Hello World!
```

Encoding and Decoding Base32

The base64 module also includes encoding and decoding functions for Base32. These functions are very similar to the Base64 functions:

similar to the Base64 functions:

```
import base64
# Creating a string
s = "Hello World!"
# Encoding the string into bytes
b = s.encode("UTF-8")
# Base32 Encode the bytes
e = base64.b32encode(b)
# Decoding the Base32 bytes to string
s1 = e.decode("UTF-8")
# Printing Base32 encoded string
print("Base32 Encoded:", s1)
# Encoding the Base32 encoded string into bytes
b1 = s1.encode("UTF-8")
# Decoding the Base32 bytes
d = base64.b32decode(b1)
# Decoding the bytes to string
s2 = d.decode("UTF-8")
print(s2)
```

This would produce the following output:

```
Base32 Encoded: JBSWY3DPEBLW64TMMQQQ====
Hello World!
```

Encoding and Decoding Base64

To include the base64 module in your script, you must import it first:

```
import base64
```

The base64 encode and decode functions both require a bytes-like object . To get our string into bytes, we must encode it using Python's built in encode function. Most commonly, the `UTF-8` encoding is used, however a full list of these standard encodings (including languages with different characters) can be found here in the official Python Documentation. Below is an example of encoding a string into bytes:

```
s = "Hello World!"
b = s.encode("UTF-8")
```

The output of the last line would be:

`b'Hello World!'`

The `b` prefix is used to denote the value is a bytes object.

To Base64 encode these bytes, we use the `base64.b64encode()` function:

```
import base64
s = "Hello World!"
b = s.encode("UTF-8")
e = base64.b64encode(b)
print(e)
```

That code would output the following:

`b'SGVsbG8gV29ybGQh'`

which is still in the bytes object. To get a string out of these bytes, we can use Python's `decode()` method with the `UTF-8` encoding:

```
import base64
s = "Hello World!"
b = s.encode("UTF-8")
e = base64.b64encode(b)
s1 = e.decode("UTF-8")
print(s1)
```

The output would then be:

`SGVsbG8gV29ybGQh`

If we wanted to encode the string and then decode we could use the `base64.b64decode()` method:

```
import base64
# Creating a string
s = "Hello World!"
# Encoding the string into bytes
b = s.encode("UTF-8")
# Base64 Encode the bytes
e = base64.b64encode(b)
```

```
# Decoding the Base64 bytes to string
s1 = e.decode("UTF-8")
# Printing Base64 encoded string
print("Base64 Encoded:", s1)
# Encoding the Base64 encoded string into bytes
b1 = s1.encode("UTF-8")
# Decoding the Base64 bytes
d = base64.b64decode(b1)
# Decoding the bytes to string
s2 = d.decode("UTF-8")
print(s2)
```

As you may have expected, the output would be the original string:

```
Base64 Encoded: SGVsbG8gV29ybGQh
Hello World!
```

Encoding and Decoding Base85

Just like the Base64, Base32, and Base16 functions, the Base85 encoding and decoding functions are `base64.b85encode()` and `base64.b85decode()` :

```
import base64
# Creating a string
s = "Hello World!"
# Encoding the string into bytes
b = s.encode("UTF-8")
# Base85 Encode the bytes
e = base64.b85encode(b)
# Decoding the Base85 bytes to string
s1 = e.decode("UTF-8")
# Printing Base85 encoded string
print("Base85 Encoded:", s1)
# Encoding the Base85 encoded string into bytes
b1 = s1.encode("UTF-8")
# Decoding the Base85 bytes
d = base64.b85decode(b1)
# Decoding the bytes to string
s2 = d.decode("UTF-8")
print(s2)
```

which outputs the following:

```
Base85 Encoded: NM&qnZy;B1a%^NF
Hello World!
```

Syntax

```
base64.b64encode(s, altchars=None)
base64.b64decode(s, altchars=None, validate=False)
base64.standard_b64encode(s)
base64.standard_b64decode(s)
base64.urlsafe_b64encode(s)
base64.urlsafe_b64decode(s)
base64.b32encode(s)
base64.b32decode(s)
base64.b16encode(s)
base64.b16decode(s)
base64.a85encode(b, foldspaces=False, wrapcol=0, pad=False, adobe=False)
base64.a85decode(b, foldpaces=False, adobe=False, ignorechars=b'\t\n\r\v')
base64.b85encode(b, pad=False)
```

```
base64.b85decode(b)
```

Parameters

Parameter	Description
base64.b64encode(s, altchars=None)	
s	A bytes-like object
altchars	A bytes-like object of length 2+ of characters to replace the '+' and '=' characters when creating the Base64 alphabet. Extra characters are ignored.
base64.b64decode(s, altchars=None, validate=False)	
s	A bytes-like object
altchars	A bytes-like object of length 2+ of characters to replace the '+' and '=' characters when creating the Base64 alphabet. Extra characters are ignored.
validate	If valide is True, the characters not in the normal Base64 alphabet or the alternative alphabet are **not** discarded before the padding check
base64.standard_b64encode(s)	
s	A bytes-like object
base64.standard_b64decode(s)	
s	A bytes-like object
base64.urlsafe_b64encode(s)	
s	A bytes-like object
base64.urlsafe_b64decode(s)	
s	A bytes-like object
b32encode(s)	
s	A bytes-like object
b32decode(s)	
s	A bytes-like object
base64.b16encode(s)	
s	A bytes-like object
base64.b16decode(s)	
s	A bytes-like object
base64.a85encode(b, foldspaces=False, wrapcol=0, pad=False, adobe=False)	
b	A bytes-like object
foldspaces	If foldspaces is True, the character 'y' will be used instead of 4 consecutive spaces.
wrapcol	The number characters before a newline (0 implies no newlines)
pad	If pad is True, the bytes are padded to a multiple of 4 before encoding
adobe	If adobe is True, the encoded sequened with be framed with '<~' and "~>' as used with Adobe products
base64.a85decode(b, foldspaces=False, adobe=False, ignorechars=b'\t\n\r\v)	
b	A bytes-like object
foldspaces	If foldspaces is True, the character 'y' will be used instead of 4 consecutive spaces.
adobe	If adobe is True, the encoded sequened with be framed with '<~' and "~>' as used with Adobe products
ignorechars	A bytes-like object of characters to ignore in the encoding process
base64.b85encode(b, pad=False)	
b	A bytes-like object
pad	If pad is True, the bytes are padded to a multiple of 4 before encoding

Parameters base64.b85decode(b)	Description
b	A bytes-like object

Remarks

Up until Python 3.4 came out, base64 encoding and decoding functions only worked with `bytes` or `bytearray` types. Now these functions accept any bytes-like object .