19.8. `binascii` — Convert between binary and ASCII

The `binascii` module contains a number of methods to convert between binary and various ASCII-encoded binary representations. Normally, you will not use these functions directly but use wrapper modules like `uu`, `base64`, or `binhex` instead. The `binascii` module contains low-level functions written in C for greater speed that are used by the higher-level modules.

> **Note:** `a2b_*` functions accept Unicode strings containing only ASCII characters. Other functions only accept bytes-like objects (such as `bytes`, `bytearray` and other objects that support the buffer protocol).
>
> *Changed in version 3.3:* ASCII-only unicode strings are now accepted by the `a2b_*` functions.

The `binascii` module defines the following functions:

`binascii.`**`a2b_uu`**(*string*)

Convert a single line of uuencoded data back to binary and return the binary data. Lines normally contain 45 (binary) bytes, except for the last line. Line data may be followed by whitespace.

`binascii.`**`b2a_uu`**(*data*)

Convert binary data to a line of ASCII characters, the return value is the converted line, including a newline char. The length of *data* should be at most 45.

`binascii.`**`a2b_base64`**(*string*)

Convert a block of base64 data back to binary and return the binary data. More than one line may be passed at a time.

`binascii.`**`b2a_base64`**(*data*, *, *newline=True*)

Convert binary data to a line of ASCII characters in base64 coding. The return value is the converted line, including a newline char if *newline* is true. The output of this function conforms to **RFC 3548**.

Changed in version 3.6: Added the *newline* parameter.

`binascii.`**`a2b_qp`**(*data*, *header=False*)

Convert a block of quoted-printable data back to binary and return the binary data. More than one line may be passed at a time. If the optional argument *header* is present and true, underscores will be decoded as spaces.

`binascii.`**`b2a_qp`**(*data*, *quotetabs=False*, *istext=True*, *header=False*)

Convert binary data to a line(s) of ASCII characters in quoted-printable encoding. The return value is the converted line(s). If the optional argument *quotetabs* is present and true, all tabs and spaces will be encoded. If the optional argument *istext* is present and true, newlines are not encoded but trailing whitespace will be encoded. If the optional argument *header* is present and true, spaces will be encoded as underscores per **RFC 1522**. If the optional argument *header* is present and false, newline characters will be encoded as well; otherwise linefeed conversion might corrupt the binary data stream.

`binascii.`**`a2b_hqx`**(*string*)

Convert binhex4 formatted ASCII data to binary, without doing RLE-decompression. The string should contain a complete number of binary bytes, or (in case of the last portion of the binhex4 data) have the remaining bits zero.

`binascii.`**`rledecode_hqx`**(*data*)

Perform RLE-decompression on the data, as per the binhex4 standard. The algorithm uses `0x90` after a byte as a repeat indicator, followed by a count. A count of `0` specifies a byte value of `0x90`. The routine returns the decompressed data, unless data input data ends in an orphaned repeat indicator, in which case the `Incomplete` exception is raised.

Changed in version 3.2: Accept only bytestring or bytearray objects as input.

`binascii.`**`rlecode_hqx`**(*data*)

Perform binhex4 style RLE-compression on *data* and return the result.

`binascii.`**`b2a_hqx`**(*data*)

Perform hexbin4 binary-to-ASCII translation and return the resulting string. The argument should already be RLE-coded, and have a length divisible by 3 (except possibly the last fragment).

`binascii.`**`crc_hqx`**(*data*, *value*)

Compute a 16-bit CRC value of *data*, starting with *value* as the initial CRC, and return the result. This uses the CRC-CCITT polynomial $x^{16} + x^{12} + x^5 + 1$, often represented as 0x1021. This CRC is used in the binhex4 format.

`binascii.`**`crc32`**(*data*[, *value*])

Compute CRC-32, the 32-bit checksum of *data*, starting with an initial CRC of *value*. The default initial CRC is zero. The algorithm is consistent with the ZIP file checksum. Since the algorithm is designed for use as a checksum algorithm, it is not suitable for use as a general hash algorithm. Use as follows:

```
print(binascii.crc32(b"hello world"))
# Or, in two pieces:
crc = binascii.crc32(b"hello")
crc = binascii.crc32(b" world", crc)
print('crc32 = {:#010x}'.format(crc))
```

Changed in version 3.0: The result is always unsigned. To generate the same numeric value across all Python versions and platforms, use `crc32(data) & 0xffffffff`.

`binascii.`**`b2a_hex`**(*data*)
`binascii.`**`hexlify`**(*data*)

Return the hexadecimal representation of the binary *data*. Every byte of *data* is converted into the corresponding 2-digit hex representation. The returned bytes object is therefore twice as long as the length of *data*.

`binascii.`**`a2b_hex`**(*hexstr*)
`binascii.`**`unhexlify`**(*hexstr*)

Return the binary data represented by the hexadecimal string *hexstr*. This function is the inverse of `b2a_hex()`. *hexstr* must contain an even number of hexadecimal digits (which can be upper or lower case), otherwise an `Error` exception is raised.

exception `binascii.`**`Error`**

Exception raised on errors. These are usually programming errors.

exception `binascii.`**`Incomplete`**

Exception raised on incomplete data. These are usually not programming errors, but may be handled by reading a little more data and trying again.

See also:

Module `base64`
Support for RFC compliant base64-style encoding in base 16, 32, 64, and 85.

Module `binhex`
Support for the binhex format used on the Macintosh.

Module `uu`
Support for UU encoding used on Unix.

Module `quopri`
Support for quoted-printable encoding used in MIME email messages.