hashlib — Cryptographic Hashing

> **Purpose:** Cryptographic hashes and message digests

The hashlib module defines an API for accessing different cryptographic hashing algorithms. To work with a specific hash algorithm, use the appropriate constructor function or new() to create a hash object. From there, the objects use the same API, no matter what algorithm is being used.

Hash Algorithms

Since hashlib is "backed" by OpenSSL, all of the algorithms provided by that library are available, including:

- md5
- sha1
- sha224
- sha256
- sha384
- sha512

Some algorithms are available on all platforms, and some depend on the underlying libraries. For lists of each, look at algorithms_guaranteed and algorithms_available respectively.

```
# hashlib_algorithms.py

import hashlib


print('Guaranteed:\n{}\n'.format(
    ', '.join(sorted(hashlib.algorithms_guaranteed))))
print('Available:\n{}'.format(
    ', '.join(sorted(hashlib.algorithms_available))))
```

```
$ python3 hashlib_algorithms.py

Guaranteed:
blake2b, blake2s, md5, sha1, sha224, sha256, sha384, sha3_224,
sha3_256, sha3_384, sha3_512, sha512, shake_128, shake_256

Available:
BLAKE2b512, BLAKE2s256, MD4, MD5, MD5-SHA1, RIPEMD160, SHA1,
SHA224, SHA256, SHA384, SHA512, blake2b, blake2b512, blake2s,
blake2s256, md4, md5, md5-sha1, ripemd160, sha1, sha224, sha256,
sha384, sha3_224, sha3_256, sha3_384, sha3_512, sha512,
shake_128, shake_256, whirlpool
```

Sample Data

All of the examples in this section use the same sample data:

```
# hashlib_data.py

import hashlib

lorem = '''Lorem ipsum dolor sit amet, consectetur adipisicing
elit, sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis
aute irure dolor in reprehenderit in voluptate velit esse cillum
dolore eu fugiat nulla pariatur. Excepteur sint occaecat
cupidatat non proident, sunt in culpa qui officia deserunt
mollit anim id est laborum.'''
```

MD5 Example

To calculate the MD5 hash, or *digest*, for a block of data (here a unicode string converted to a byte string), first create the hash object, then add the data and call `digest()` or `hexdigest()`.

```
# hashlib_md5.py

import hashlib

from hashlib_data import lorem

h = hashlib.md5()
h.update(lorem.encode('utf-8'))
print(h.hexdigest())
```

This example uses the `hexdigest()` method instead of `digest()` because the output is formatted so it can be printed clearly. If a binary digest value is acceptable, use `digest()`.

```
$ python3 hashlib_md5.py

3f2fd2c9e25d60fb0fa5d593b802b7a8
```

SHA1 Example

A SHA1 digest is calculated in the same way.

```
# hashlib_sha1.py

import hashlib

from hashlib_data import lorem

h = hashlib.sha1()
h.update(lorem.encode('utf-8'))
print(h.hexdigest())
```

The digest value is different in this example because the algorithm changed from MD5 to SHA1.

```
$ python3 hashlib_sha1.py

ea360b288b3dd178fe2625f55b2959bf1dba6eef
```

Creating a Hash by Name

Sometimes it is more convenient to refer to the algorithm by name in a string rather than by using the constructor function directly. It is useful, for example, to be able to store the hash type in a configuration file. In those cases, use `new()` to create a hash calculator.

```
# hashlib_new.py

import argparse
import hashlib
import sys

from hashlib_data import lorem


parser = argparse.ArgumentParser('hashlib demo')
parser.add_argument(
    'hash_name',
    choices=hashlib.algorithms_available,
    help='the name of the hash algorithm to use',
)
parser.add_argument(
    'data',
    nargs='?',
    nargs='?',
    default=lorem,
    help='the input data to hash, defaults to lorem ipsum',
)
args = parser.parse_args()

h = hashlib.new(args.hash_name)
h.update(args.data.encode('utf-8'))
print(h.hexdigest())
```

When run with a variety of arguments:

```
$ python3 hashlib_new.py sha1

ea360b288b3dd178fe2625f55b2959bf1dba6eef

$ python3 hashlib_new.py sha256

3c887cc71c67949df29568119cc646f46b9cd2c2b39d456065646bc2fc09ffd8

$ python3 hashlib_new.py sha512

a7e53384eb9bb4251a19571450465d51809e0b7046101b87c4faef96b9bc904cf7f90
035f444952dfd9f6084eeee2457433f3ade614712f42f80960b2fca43ff

$ python3 hashlib_new.py md5

3f2fd2c9e25d60fb0fa5d593b802b7a8
```

Incremental Updates

The `update()` method of the hash calculators can be called repeatedly. Each time, the digest is updated based on the additional text fed in. Updating incrementally is more efficient than reading an entire file into memory, and produces the same results.

```
# hashlib_update.py

import hashlib

from hashlib_data import lorem

h = hashlib.md5()
h.update(lorem.encode('utf-8'))
all_at_once = h.hexdigest()


def chunkize(size, text):
    "Return parts of the text in size-based increments."
    start = 0
    while start < len(text):
        chunk = text[start:start + size]
        yield chunk
        start += size
    return


h = hashlib.md5()
for chunk in chunkize(64, lorem.encode('utf-8')):
    h.update(chunk)
line_by_line = h.hexdigest()

print('All at once :', all_at_once)
print('Line by line:', line_by_line)
print('Same        :', (all_at_once == line_by_line))
```

This example demonstrates how to update a digest incrementally as data is read or otherwise produced.

```
$ python3 hashlib_update.py

All at once : 3f2fd2c9e25d60fb0fa5d593b802b7a8
All at once : 3f2fd2c9e25d60fb0fa5d593b802b7a8
Line by line: 3f2fd2c9e25d60fb0fa5d593b802b7a8
Same        : True
```

See also

- Standard library documentation for hashlib
- hmac – The hmac module.
- OpenSSL – An open source encryption toolkit.
- Cryptography module – A Python package that provides cryptographic recipes and primitives.
- Voidspace: IronPython and hashlib – A wrapper for `hashlib` that works with IronPython.

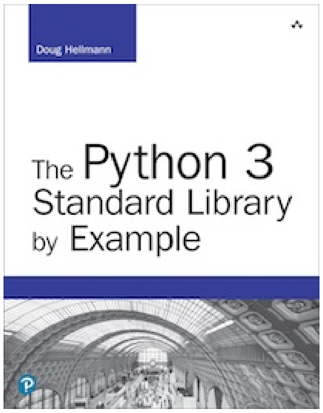
Cryptography

hmac — Cryptographic Message Signing and Verification

This page was last updated 2018-12-09.

Navigation

Cryptography
hmac — Cryptographic Message Signing and Verification



The output from all the example programs from PyMOTW-3 has been generated with Python 3.7.1, unless otherwise noted. Some of the features described here may not be available in earlier versions of Python.

Looking for examples for Python 2?

This Site

Module Index
Index

Other Writing

Blog
The Python Standard Library By Example