# :mod:`crypt` --- Function to check Unix passwords

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 1); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 4)

Unknown directive type "module".

```
.. module:: crypt
   :platform: Unix
   :synopsis: The crypt() function used to check Unix passwords.
   :deprecated:
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 9)

Unknown directive type "moduleauthor".

```
.. moduleauthor:: Steven D. Majewski <sdm7g@virginia.edu>
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 10)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Steven D. Majewski <sdm7g@virginia.edu>
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 11)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Peter Funk <pf@artcom-gmbh.de>
```

**Source code:** :source:`Lib/crypt.py`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 13); *backlink*

Unknown interpreted text role "source".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 15)

Unknown directive type "index".

```
.. index::
   single: crypt(3)
   pair: cipher; DES
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 19)

Unknown directive type "deprecated".

```
.. deprecated:: 3.11
   The :mod:`crypt` module is deprecated (see :pep:`594` for details).
```

---

This module implements an interface to the :manpage:`crypt(3)` routine, which is a one-way hash function based upon a modified DES algorithm; see the Unix man page for further details. Possible uses include storing hashed passwords so you can check passwords without storing the actual password, or attempting to crack Unix passwords with a dictionary.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 24); *backlink***

Unknown interpreted text role "manpage".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 30)**

Unknown directive type "index".

```
.. index:: single: crypt(3)
```

Notice that the behavior of this module depends on the actual implementation of the :manpage:`crypt(3)` routine in the running system. Therefore, any extensions available on the current implementation will also be available on this module.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 32); *backlink***

Unknown interpreted text role "manpage".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 37)**

Unknown directive type "availability".

```
.. availability:: Unix. Not available on VxWorks.
```

## Hashing Methods

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 42)**

Unknown directive type "versionadded".

```
.. versionadded:: 3.3
```

The :mod:`crypt` module defines the list of hashing methods (not all methods are available on all platforms):

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 44); *backlink***

Unknown interpreted text role "mod".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 47)**

Unknown directive type "data".

```
.. data:: METHOD_SHA512

   A Modular Crypt Format method with 16 character salt and 86 character
   hash based on the SHA-512 hash function.  This is the strongest method.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 52)**

Unknown directive type "data".

```
.. data:: METHOD_SHA256

   Another Modular Crypt Format method with 16 character salt and 43
   character hash based on the SHA-256 hash function.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 57)

Unknown directive type "data".

```
.. data:: METHOD_BLOWFISH

   Another Modular Crypt Format method with 22 character salt and 31
   character hash based on the Blowfish cipher.

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 64)

Unknown directive type "data".

```
.. data:: METHOD_MD5

   Another Modular Crypt Format method with 8 character salt and 22
   character hash based on the MD5 hash function.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 69)

Unknown directive type "data".

```
.. data:: METHOD_CRYPT

   The traditional method with a 2 character salt and 13 characters of
   hash.  This is the weakest method.
```

## Module Attributes

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 78)

Unknown directive type "versionadded".

```
.. versionadded:: 3.3
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 80)

Unknown directive type "attribute".

```
.. attribute:: methods

   A list of available password hashing algorithms, as
   ``crypt.METHOD_*`` objects.  This list is sorted from strongest to
   weakest.
```

## Module Functions

The :mod:`crypt` module defines the following functions:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 90); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 92)

Unknown directive type "function".

```
.. function:: crypt(word, salt=None)

   *word* will usually be a user's password as typed at a prompt or  in a graphical
   interface.  The optional *salt* is either a string as returned from
   :func:`mksalt`, one of the ``crypt.METHOD_*`` values (though not all
   may be available on all platforms), or a full encrypted password
   including salt, as returned by this function.  If *salt* is not
   provided, the strongest method available in :attr:`methods` will be used.

   Checking a password is usually done by passing the plain-text password
   as *word* and the full results of a previous :func:`crypt` call,
   which should be the same as the results of this call.

   *salt* (either a random 2 or 16 character string, possibly prefixed with
   ``$digit$`` to indicate the method) which will be used to perturb the
   encryption algorithm.  The characters in *salt* must be in the set
   ``[./a-zA-Z0-9]``, with the exception of Modular Crypt Format which
   prefixes a ``$digit$``.

   Returns the hashed password as a string, which will be composed of
   characters from the same alphabet as the salt.

   .. index:: single: crypt(3)

   Since a few :manpage:`crypt(3)` extensions allow different values, with
   different sizes in the *salt*, it is recommended to use  the full crypted
   password as salt when checking for a password.

   .. versionchanged:: 3.3
      Accept ``crypt.METHOD_*`` values in addition to strings for *salt*.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 124)**

Unknown directive type "function".

```
.. function:: mksalt(method=None, *, rounds=None)

   Return a randomly generated salt of the specified method.  If no
   *method* is given, the strongest method available in :attr:`methods` is
   used.

   The return value is a string suitable for passing as the *salt* argument
   to :func:`crypt`.

   *rounds* specifies the number of rounds for ``METHOD_SHA256``,
   ``METHOD_SHA512`` and ``METHOD_BLOWFISH``.
   For ``METHOD_SHA256`` and ``METHOD_SHA512`` it must be an integer between
   ``1000`` and ``999_999_999``, the default is ``5000``.  For
   ``METHOD_BLOWFISH`` it must be a power of two between ``16`` (2\ :sup:`4`)
   and ``2_147_483_648`` (2\ :sup:`31`), the default is ``4096``
   (2\ :sup:`12`).

   .. versionadded:: 3.3

   .. versionchanged:: 3.7
      Added the *rounds* parameter.
```

## Examples

A simple example illustrating typical use (a constant-time comparison operation is needed to limit exposure to timing attacks. :func:`hmac.compare_digest` is suitable for this purpose):

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]crypt.rst, line 150); *backlink***

Unknown interpreted text role "func".

```
import pwd
import crypt
import getpass
```

```
from hmac import compare_digest as compare_hash

def login():
    username = input('Python login: ')
    cryptedpasswd = pwd.getpwnam(username)[1]
    if cryptedpasswd:
        if cryptedpasswd == 'x' or cryptedpasswd == '*':
            raise ValueError('no support for shadow passwords')
        cleartext = getpass.getpass()
        return compare_hash(crypt.crypt(cleartext, cryptedpasswd), cryptedpasswd)
    else:
        return True
```

To generate a hash of a password using the strongest available method and check it against the original:

```
import crypt
from hmac import compare_digest as compare_hash

hashed = crypt.crypt(plaintext)
if not compare_hash(hashed, crypt.crypt(plaintext, hashed)):
    raise ValueError("hashed version doesn't validate against original")
```