:mod:`dbm` --- Interfaces to Unix "databases"

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.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)dbm.rst`, **line 4**)
>
> Unknown directive type "module".
>
> ```
> .. module:: dbm
>    :synopsis: Interfaces to various Unix "database" formats.
> ```

Source code: :source:`Lib/dbm/__init__.py`

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, **line 7**); *backlink*
>
> Unknown interpreted text role "source".

:mod:`dbm` is a generic interface to variants of the DBM database --- :mod:`dbm.gnu` or :mod:`dbm.ndbm`. If none of these modules is installed, the slow-but-simple implementation in module :mod:`dbm.dumb` will be used. There is a third party interface to the Oracle Berkeley DB.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, **line 11**); *backlink*
>
> Unknown interpreted text role "mod".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, **line 11**); *backlink*
>
> Unknown interpreted text role "mod".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, **line 11**); *backlink*
>
> Unknown interpreted text role "mod".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, **line 11**); *backlink*
>
> Unknown interpreted text role "mod".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, **line 18**)
>
> Unknown directive type "exception".
>
> ```
> .. exception:: error
>
>    A tuple containing the exceptions that can be raised by each of the supported
>    modules, with a unique exception also named :exc:`dbm.error` as the first
>    item --- the latter is used when :exc:`dbm.error` is raised.
> ```

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, **line 25**)
>
> Unknown directive type "function".

```
.. function:: whichdb(filename)

   This function attempts to guess which of the several simple database modules
   available --- :mod:`dbm.gnu`, :mod:`dbm.ndbm` or :mod:`dbm.dumb` --- should
   be used to open a given file.

   Returns one of the following values: ``None`` if the file can't be opened
   because it's unreadable or doesn't exist; the empty string (``''``) if the
   file's format can't be guessed; or a string containing the required module
   name, such as ``'dbm.ndbm'`` or ``'dbm.gnu'``.
```

The object returned by :func:`.open` supports the same basic functionality as dictionaries; keys and their corresponding values can be stored, retrieved, and deleted, and the :keyword:`in` operator and the :meth:`keys` method are available, as well as :meth:`get` and :meth:`setdefault`.

Unknown interpreted text role "meth".

Key and values are always stored as bytes. This means that when strings are used they are implicitly converted to the default encoding before being stored.

These objects also support being used in a :keyword:`with` statement, which will automatically close them when done.

The following example records some hostnames and a corresponding title, and then prints out the contents of the database:

```
import dbm

# Open database, creating it if necessary.
with dbm.open('cache', 'c') as db:

    # Record some values
    db[b'hello'] = b'there'
    db['www.python.org'] = 'Python Website'
    db['www.cnn.com'] = 'Cable News Network'
```

```
    # Note that the keys are considered bytes now.
    assert db[b'www.python.org'] == b'Python Website'
    # Notice how the value is now in bytes.
    assert db['www.cnn.com'] == b'Cable News Network'

    # Often-used methods of the dict interface work too.
    print(db.get('python.org', b'not present'))

    # Storing a non-string key or value will raise an exception (most
    # likely a TypeError).
    db['www.yahoo.com'] = 4

# db is automatically closed when leaving the with statement.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 124)

Unknown directive type "seealso".

```
    .. seealso::

        Module :mod:`shelve`
            Persistence module which stores non-string data.
```

The individual submodules are described in the following sections.

:mod:`dbm.gnu` --- GNU's reinterpretation of dbm

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 133); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 136)

Unknown directive type "module".

```
    .. module:: dbm.gnu
       :platform: Unix
       :synopsis: GNU's reinterpretation of dbm.
```

Source code: :source:`Lib/dbm/gnu.py`

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 140); *backlink*

Unknown interpreted text role "source".

This module is quite similar to the :mod:`dbm` module, but uses the GNU library `gdbm` instead to provide some additional functionality. Please note that the file formats created by :mod:`dbm.gnu` and :mod:`dbm.ndbm` are incompatible.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 144); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 144); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 144); *backlink*

Unknown interpreted text role "mod".

The :mod:`dbm.gnu` module provides an interface to the GNU DBM library. `dbm.gnu.gdbm` objects behave like mappings (dictionaries), except that keys and values are always converted to bytes before storing. Printing a `gdbm` object doesn't print the keys and values, and the :meth:`items` and :meth:`values` methods are not supported.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 148); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 148); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 148); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 155)

Unknown directive type "exception".

```
.. exception:: error

   Raised on :mod:`dbm.gnu`-specific errors, such as I/O errors. :exc:`KeyError` is
   raised for general mapping errors like specifying an incorrect key.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 161)

Unknown directive type "function".

```
.. function:: open(filename[, flag[, mode]])

   Open a ``gdbm`` database and return a :class:`gdbm` object.  The *filename*
   argument is the name of the database file.

   The optional *flag* argument can be:

   +---------+-------------------------------------------+
   | Value   | Meaning                                   |
   +=========+===========================================+
   | ``'r'`` | Open existing database for reading only   |
   |         | (default)                                 |
   +---------+-------------------------------------------+
   | ``'w'`` | Open existing database for reading and    |
   |         | writing                                   |
   +---------+-------------------------------------------+
   | ``'c'`` | Open database for reading and writing,    |
   |         | creating it if it doesn't exist           |
   +---------+-------------------------------------------+
   | ``'n'`` | Always create a new, empty database, open |
   |         | for reading and writing                   |
   +---------+-------------------------------------------+

   The following additional characters may be appended to the flag to control
   how the database is opened:

   +---------+--------------------------------------------+
   | Value   | Meaning                                    |
   +=========+============================================+
   | ``'f'`` | Open the database in fast mode.  Writes    |
   |         | to the database will not be synchronized.  |
   +---------+--------------------------------------------+
   | ``'s'`` | Synchronized mode. This will cause changes |
   |         | to the database to be immediately written  |
   |         | to the file.                               |
   +---------+--------------------------------------------+
   | ``'u'`` | Do not lock database.                      |
   +---------+--------------------------------------------+
```

```
   Not all flags are valid for all versions of ``gdbm``.  The module constant
   :const:`open_flags` is a string of supported flag characters.  The exception
   :exc:`error` is raised if an invalid flag is specified.

   The optional *mode* argument is the Unix mode of the file, used only when the
   database has to be created.  It defaults to octal ``0o666``.

   In addition to the dictionary-like methods, ``gdbm`` objects have the
   following methods:

   .. versionchanged:: 3.11
      Accepts :term:`path-like object` for filename.

   .. method:: gdbm.firstkey()

      It's possible to loop over every key in the database using this method  and the
      :meth:`nextkey` method.  The traversal is ordered by ``gdbm``'s internal
      hash values, and won't be sorted by the key values.  This method returns
      the starting key.

   .. method:: gdbm.nextkey(key)

      Returns the key that follows *key* in the traversal.  The following code prints
      every key in the database ``db``, without having to create a list in memory that
      contains them all::

         k = db.firstkey()
         while k is not None:
             print(k)
             k = db.nextkey(k)

   .. method:: gdbm.reorganize()

      If you have carried out a lot of deletions and would like to shrink the space
      used by the ``gdbm`` file, this routine will reorganize the database.  ``gdbm``
      objects will not shorten the length of a database file except by using this
      reorganization; otherwise, deleted file space will be kept and reused as new
      (key, value) pairs are added.

   .. method:: gdbm.sync()

      When the database has been opened in fast mode, this method forces any
      unwritten data to be written to the disk.

   .. method:: gdbm.close()

      Close the ``gdbm`` database.
```

:mod:`dbm.ndbm` --- Interface based on ndbm

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 248); *backlink***
>
> Unknown interpreted text role "mod".

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 251)**
>
> Unknown directive type "module".
>
> ```
>    .. module:: dbm.ndbm
>       :platform: Unix
>       :synopsis: The standard "database" interface, based on ndbm.
> ```

Source code: :source:`Lib/dbm/ndbm.py`

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 255); *backlink***
>
> Unknown interpreted text role "source".

The :mod:`dbm.ndbm` module provides an interface to the Unix "(n)dbm" library. Dbm objects behave like mappings (dictionaries),

except that keys and values are always stored as bytes. Printing a `dbm` object doesn't print the keys and values, and the :meth:`items` and :meth:`values` methods are not supported.

This module can be used with the "classic" ndbm interface or the GNU GDBM compatibility interface. On Unix, the :program:`configure` script will attempt to locate the appropriate header file to simplify building this module.

```
+---------+-----------------------------------------+
| ``'n'`` | Always create a new, empty database, open |
|         | for reading and writing                 |
+---------+-----------------------------------------+

The optional *mode* argument is the Unix mode of the file, used only when the
database has to be created.  It defaults to octal ``0o666`` (and will be
modified by the prevailing umask).

In addition to the dictionary-like methods, ``ndbm`` objects
provide the following method:

.. versionchanged:: 3.11
   Accepts :term:`path-like object` for filename.

.. method:: ndbm.close()

   Close the ``ndbm`` database.
```

:mod:`dbm.dumb` --- Portable DBM implementation

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 317); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 320)

Unknown directive type "module".

```
.. module:: dbm.dumb
   :synopsis: Portable implementation of the simple DBM interface.
```

Source code: :source:`Lib/dbm/dumb.py`

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 323); *backlink*

Unknown interpreted text role "source".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 325)

Unknown directive type "index".

```
.. index:: single: databases
```

Note

The :mod:`dbm.dumb` module is intended as a last resort fallback for the :mod:`dbm` module when a more robust module is not available. The :mod:`dbm.dumb` module is not written for speed and is not nearly as heavily used as the other database modules.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 329); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)dbm.rst`, line 329); *backlink*

Unknown interpreted text role "mod".

The :mod:`dbm.dumb` module provides a persistent dictionary-like interface which is written entirely in Python. Unlike other modules such as :mod:`dbm.gnu` no external library is required. As with other persistent mappings, the keys and values are always stored as bytes.

The module defines the following:

```
.. versionchanged:: 3.5
   :func:`.open` always creates a new database when the flag has the value
   ``'n'``.

.. versionchanged:: 3.8
   A database opened with flags ``'r'`` is now read-only.  Opening with
   flags ``'r'`` and ``'w'`` no longer creates a database if it does not
   exist.

.. versionchanged:: 3.11
   Accepts :term:`path-like object` for filename.

In addition to the methods provided by the
:class:`collections.abc.MutableMapping` class, :class:`dumbdbm` objects
provide the following methods:

.. method:: dumbdbm.sync()

   Synchronize the on-disk directory and data files.  This method is called
   by the :meth:`Shelve.sync` method.

.. method:: dumbdbm.close()

   Close the ``dumbdbm`` database.
```