:mod:`getopt` --- C-style parser for command line options

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

Unknown directive type "module".

```
.. module:: getopt
   :synopsis: Portable parser for command line options; support both short and
              long option names.
```

Source code: :source:`Lib/getopt.py`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]getopt.rst, line 8); *backlink*

Unknown interpreted text role "source".

Note

The :mod:`getopt` module is a parser for command line options whose API is designed to be familiar to users of the C :c:func:`getopt` function. Users who are unfamiliar with the C :c:func:`getopt` function or who would like to write less code and get better help and error messages should consider using the :mod:`argparse` module instead.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]getopt.rst, line 12); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]getopt.rst, line 12); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]getopt.rst, line 12); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]getopt.rst, line 12); *backlink*

Unknown interpreted text role "mod".

This module helps scripts to parse the command line arguments in `sys.argv`. It supports the same conventions as the Unix :c:func:`getopt` function (including the special meanings of arguments of the form '-' and '--'). Long options similar to those supported by GNU software may be used as well via an optional third argument.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]getopt.rst, line 20); *backlink*

Unknown interpreted text role "c:func".

This module provides two functions and an exception:

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

Unknown directive type "function".

```
.. function:: getopt(args, shortopts, longopts=[])

   Parses command line options and parameter list.  *args* is the argument list to
   be parsed, without the leading reference to the running program. Typically, this
   means ``sys.argv[1:]``. *shortopts* is the string of option letters that the
   script wants to recognize, with options that require an argument followed by a
   colon (``':'``; i.e., the same format that Unix :c:func:`getopt` uses).

   .. note::

      Unlike GNU :c:func:`getopt`, after a non-option argument, all further
      arguments are considered also non-options. This is similar to the way
      non-GNU Unix systems work.

   *longopts*, if specified, must be a list of strings with the names of the
   long options which should be supported.  The leading ``'--'`` characters
   should not be included in the option name.  Long options which require an
   argument should be followed by an equal sign (``'='``).  Optional arguments
   are not supported.  To accept only long options, *shortopts* should be an
   empty string.  Long options on the command line can be recognized so long as
   they provide a prefix of the option name that matches exactly one of the
   accepted options.  For example, if *longopts* is ``['foo', 'frob']``, the
   option ``--fo`` will match as ``--foo``, but ``--f`` will
   not match uniquely, so :exc:`GetoptError` will be raised.

   The return value consists of two elements: the first is a list of ``(option,
   value)`` pairs; the second is the list of program arguments left after the
   option list was stripped (this is a trailing slice of *args*).  Each
   option-and-value pair returned has the option as its first element, prefixed
   with a hyphen for short options (e.g., ``'-x'``) or two hyphens for long
   options (e.g., ``'--long-option'``), and the option argument as its
   second element, or an empty string if the option has no argument.  The
   options occur in the list in the same order in which they were found, thus
   allowing multiple occurrences.  Long and short options may be mixed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]getopt.rst, line 66)

Unknown directive type "function".

```
.. function:: gnu_getopt(args, shortopts, longopts=[])

   This function works like :func:`getopt`, except that GNU style scanning mode is
   used by default. This means that option and non-option arguments may be
   intermixed. The :func:`getopt` function stops processing options as soon as a
   non-option argument is encountered.

   If the first character of the option string is ``'+'``, or if the environment
   variable :envvar:`POSIXLY_CORRECT` is set, then option processing stops as
   soon as a non-option argument is encountered.
```

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

Unknown directive type "exception".

```
.. exception:: GetoptError

   This is raised when an unrecognized option is found in the argument list or when
   an option requiring an argument is given none. The argument to the exception is
   a string indicating the cause of the error.  For long options, an argument given
   to an option which does not require one will also cause this exception to be
   raised.  The attributes :attr:`msg` and :attr:`opt` give the error message and
   related option; if there is no specific option to which the exception relates,
   :attr:`opt` is an empty string.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]getopt.rst, line 89)

Unknown directive type "exception".

```
.. exception:: error

   Alias for :exc:`GetoptError`; for backward compatibility.
```

An example using only Unix style options:

```
>>> import getopt
>>> args = '-a -b -cfoo -d bar a1 a2'.split()
>>> args
['-a', '-b', '-cfoo', '-d', 'bar', 'a1', 'a2']
>>> optlist, args = getopt.getopt(args, 'abc:d:')
>>> optlist
[('-a', ''), ('-b', ''), ('-c', 'foo'), ('-d', 'bar')]
>>> args
['a1', 'a2']
```

Using long option names is equally easy:

```
>>> s = '--condition=foo --testing --output-file abc.def -x a1 a2'
>>> args = s.split()
>>> args
['--condition=foo', '--testing', '--output-file', 'abc.def', '-x', 'a1', 'a2']
>>> optlist, args = getopt.getopt(args, 'x', [
...     'condition=', 'output-file=', 'testing'])
>>> optlist
[('--condition', 'foo'), ('--testing', ''), ('--output-file', 'abc.def'), ('-x', '')]
>>> args
['a1', 'a2']
```

In a script, typical usage is something like this:

```
import getopt, sys

def main():
    try:
        opts, args = getopt.getopt(sys.argv[1:], "ho:v", ["help", "output="])
    except getopt.GetoptError as err:
        # print help information and exit:
        print(err)  # will print something like "option -a not recognized"
        usage()
        sys.exit(2)
    output = None
    verbose = False
    for o, a in opts:
        if o == "-v":
            verbose = True
        elif o in ("-h", "--help"):
            usage()
            sys.exit()
        elif o in ("-o", "--output"):
            output = a
        else:
            assert False, "unhandled option"
    # ...

if __name__ == "__main__":
    main()
```

Note that an equivalent command line interface could be produced with less code and more informative help and error messages by using the :mod:`argparse` module:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]getopt.rst, line 147); *backlink*

Unknown interpreted text role "mod".

```
import argparse

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('-o', '--output')
    parser.add_argument('-v', dest='verbose', action='store_true')
```

```
args = parser.parse_args()
# ... do something with args.output ...
# ... do something with args.verbose ..
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]getopt.rst, line 160)

Unknown directive type "seealso".

```
.. seealso::

   Module :mod:`argparse`
      Alternative command line option and argument parsing library.
```