:mod:`difflib` --- Helpers for computing deltas

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

Unknown directive type "module".

```
.. module:: difflib
   :synopsis: Helpers for computing differences between objects.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 7)

Unknown directive type "moduleauthor".

```
.. moduleauthor:: Tim Peters <tim_one@users.sourceforge.net>
```

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

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Tim Peters <tim_one@users.sourceforge.net>
```

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

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

Unknown interpreted text role "source".

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

Unknown directive type "testsetup".

```
.. testsetup::

   import sys
   from difflib import *
```

This module provides classes and functions for comparing sequences. It can be used for example, for comparing files, and can produce information about file differences in various formats, including HTML and context and unified diffs. For comparing directories and files, see also, the :mod:`filecmp` module.

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

Unknown interpreted text role "mod".

This is a flexible class for comparing pairs of sequences of any type, so long as the sequence elements are :term:`hashable`. The basic algorithm predates, and is a little fancier than, an algorithm published in the late 1980's by Ratcliff and Obershelp under the hyperbolic name "gestalt pattern matching." The idea is to find the longest contiguous matching subsequence that contains no "junk" elements; these "junk" elements are ones that are uninteresting in some sense, such as blank lines or whitespace. (Handling junk is an extension to the Ratcliff and Obershelp algorithm.) The same idea is then applied recursively to the pieces of the sequences to the left and to the right of the matching subsequence. This does not yield minimal edit sequences, but does tend to yield matches that "look right" to people.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 29); *backlink*

Unknown interpreted text role "term".

Timing: The basic Ratcliff-Obershelp algorithm is cubic time in the worst case and quadratic time in the expected case. :class:`SequenceMatcher` is quadratic time for the worst case and has expected-case behavior dependent in a complicated way on how many elements the sequences have in common; best case time is linear.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 41); *backlink*

Unknown interpreted text role "class".

Automatic junk heuristic: :class:`SequenceMatcher` supports a heuristic that automatically treats certain sequence items as junk. The heuristic counts how many times each individual item appears in the sequence. If an item's duplicates (after the first one) account for more than 1% of the sequence and the sequence is at least 200 items long, this item is marked as "popular" and is treated as junk for the purpose of sequence matching. This heuristic can be turned off by setting the `autojunk` argument to `False` when creating the :class:`SequenceMatcher`.

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 55)

Unknown directive type "versionadded".

```
.. versionadded:: 3.2
   The *autojunk* parameter.
```

This is a class for comparing sequences of lines of text, and producing human-readable differences or deltas. Differ uses :class:`SequenceMatcher` both to compare sequences of lines, and to compare sequences of characters within similar (near-matching) lines.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 61); *backlink*

Unknown interpreted text role "class".

Each line of a :class:`Differ` delta begins with a two-letter code:

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

Unknown interpreted text role "class".

Code	Meaning
`'- '`	line unique to sequence 1
`'+ '`	line unique to sequence 2
`'  '`	line common to both sequences
`'? '`	line not present in either input sequence

Lines beginning with '`?`' attempt to guide the eye to intraline differences, and were not present in either input sequence. These lines can be confusing if the sequences contain tab characters.

This class can be used to create an HTML table (or a complete HTML file containing the table) showing a side by side, line by line comparison of text with inter-line and intra-line change highlights. The table can be generated in either full or contextual difference mode.

The constructor for this class is:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 95)

Unknown directive type "method".

```
.. method:: __init__(tabsize=8, wrapcolumn=None, linejunk=None, charjunk=IS_CHARACTER_JUNK)

   Initializes instance of :class:`HtmlDiff`.

   *tabsize* is an optional keyword argument to specify tab stop spacing and
   defaults to ``8``.

   *wrapcolumn* is an optional keyword to specify column number where lines are
   broken and wrapped, defaults to ``None`` where lines are not wrapped.

   *linejunk* and *charjunk* are optional keyword arguments passed into :func:`ndiff`
   (used by :class:`HtmlDiff` to generate the side by side HTML differences).  See
   :func:`ndiff` documentation for argument default values and descriptions.
```

The following methods are public:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 111)

Unknown directive type "method".

```
.. method:: make_file(fromlines, tolines, fromdesc='', todesc='', context=False, \
                      numlines=5, *, charset='utf-8')

   Compares *fromlines* and *tolines* (lists of strings) and returns a string which
   is a complete HTML file containing a table showing line by line differences with
   inter-line and intra-line changes highlighted.

   *fromdesc* and *todesc* are optional keyword arguments to specify from/to file
   column header strings (both default to an empty string).
```

```
      *context* and *numlines* are both optional keyword arguments. Set *context* to
      ``True`` when contextual differences are to be shown, else the default is
      ``False`` to show the full files. *numlines* defaults to ``5``.  When *context*
      is ``True`` *numlines* controls the number of context lines which surround the
      difference highlights.  When *context* is ``False`` *numlines* controls the
      number of lines which are shown before a difference highlight when using the
      "next" hyperlinks (setting to zero would cause the "next" hyperlinks to place
      the next difference highlight at the top of the browser without any leading
      context).

      .. note::
         *fromdesc* and *todesc* are interpreted as unescaped HTML and should be
         properly escaped while receiving input from untrusted sources.

      .. versionchanged:: 3.5
         *charset* keyword-only argument was added.  The default charset of
         HTML document changed from ``'ISO-8859-1'`` to ``'utf-8'``.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 139)

Unknown directive type "method".

```
   .. method:: make_table(fromlines, tolines, fromdesc='', todesc='', context=False, numlines=5)

      Compares *fromlines* and *tolines* (lists of strings) and returns a string which
      is a complete HTML table showing line by line differences with inter-line and
      intra-line changes highlighted.

      The arguments for this method are the same as those for the :meth:`make_file`
      method.
```

:file:`Tools/scripts/diff.py` is a command-line front-end to this class and contains a good example of its use.

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

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 152)

Unknown directive type "function".

```
   .. function:: context_diff(a, b, fromfile='', tofile='', fromfiledate='', tofiledate='', n=3, lineterm='\\n')

      Compare *a* and *b* (lists of strings); return a delta (a :term:`generator`
      generating the delta lines) in context diff format.

      Context diffs are a compact way of showing just the lines that have changed plus
      a few lines of context.  The changes are shown in a before/after style.  The
      number of context lines is set by *n* which defaults to three.

      By default, the diff control lines (those with ``***`` or ``---``) are created
      with a trailing newline.  This is helpful so that inputs created from
      :func:`io.IOBase.readlines` result in diffs that are suitable for use with
      :func:`io.IOBase.writelines` since both the inputs and outputs have trailing
      newlines.

      For inputs that do not have trailing newlines, set the *lineterm* argument to
      ``""`` so that the output will be uniformly newline free.

      The context diff format normally has a header for filenames and modification
      times.  Any or all of these may be specified using strings for *fromfile*,
      *tofile*, *fromfiledate*, and *tofiledate*.  The modification times are normally
      expressed in the ISO 8601 format. If not specified, the
      strings default to blanks.

         >>> s1 = ['bacon\n', 'eggs\n', 'ham\n', 'guido\n']
         >>> s2 = ['python\n', 'eggy\n', 'hamster\n', 'guido\n']
         >>> sys.stdout.writelines(context_diff(s1, s2, fromfile='before.py', tofile='after.py'))
         *** before.py
         --- after.py
         ***************
         *** 1,4 ****
         ! bacon
         ! eggs
         ! ham
           guido
         --- 1,4 ----
         ! python
         ! eggy
         ! hamster
           guido

      See :ref:`difflib-interface` for a more detailed example.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 196)

Unknown directive type "function".

```
   .. function:: get_close_matches(word, possibilities, n=3, cutoff=0.6)
```

```
    Return a list of the best "good enough" matches.  *word* is a sequence for which
    close matches are desired (typically a string), and *possibilities* is a list of
    sequences against which to match *word* (typically a list of strings).

    Optional argument *n* (default ``3``) is the maximum number of close matches to
    return; *n* must be greater than ``0``.

    Optional argument *cutoff* (default ``0.6``) is a float in the range [0, 1].
    Possibilities that don't score at least that similar to *word* are ignored.

    The best (no more than *n*) matches among the possibilities are returned in a
    list, sorted by similarity score, most similar first.

       >>> get_close_matches('appel', ['ape', 'apple', 'peach', 'puppy'])
       ['apple', 'ape']
       >>> import keyword
       >>> get_close_matches('wheel', keyword.kwlist)
       ['while']
       >>> get_close_matches('pineapple', keyword.kwlist)
       []
       >>> get_close_matches('accept', keyword.kwlist)
       ['except']
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 222)

Unknown directive type "function".

```
.. function:: ndiff(a, b, linejunk=None, charjunk=IS_CHARACTER_JUNK)

   Compare *a* and *b* (lists of strings); return a :class:`Differ`\ -style
   delta (a :term:`generator` generating the delta lines).

   Optional keyword parameters *linejunk* and *charjunk* are filtering functions
   (or ``None``):

   *linejunk*: A function that accepts a single string argument, and returns
   true if the string is junk, or false if not. The default is ``None``. There
   is also a module-level function :func:`IS_LINE_JUNK`, which filters out lines
   without visible characters, except for at most one pound character (``'#'``)
   -- however the underlying :class:`SequenceMatcher` class does a dynamic
   analysis of which lines are so frequent as to constitute noise, and this
   usually works better than using this function.

   *charjunk*: A function that accepts a character (a string of length 1), and
   returns if the character is junk, or false if not. The default is module-level
   function :func:`IS_CHARACTER_JUNK`, which filters out whitespace characters (a
   blank or tab; it's a bad idea to include newline in this!).

   :file:`Tools/scripts/ndiff.py` is a command-line front-end to this function.

      >>> diff = ndiff('one\ntwo\nthree\n'.splitlines(keepends=True),
      ...              'ore\ntree\nemu\n'.splitlines(keepends=True))
      >>> print(''.join(diff), end="")
      - one
      ?  ^
      + ore
      ?  ^
      - two
      - three
      ?  -
      + tree
      + emu
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 259)

Unknown directive type "function".

```
.. function:: restore(sequence, which)

   Return one of the two sequences that generated a delta.

   Given a *sequence* produced by :meth:`Differ.compare` or :func:`ndiff`, extract
   lines originating from file 1 or 2 (parameter *which*), stripping off line
   prefixes.

   Example:

      >>> diff = ndiff('one\ntwo\nthree\n'.splitlines(keepends=True),
      ...              'ore\ntree\nemu\n'.splitlines(keepends=True))
      >>> diff = list(diff) # materialize the generated delta into a list
      >>> print(''.join(restore(diff, 1)), end="")
      one
      two
      three
      >>> print(''.join(restore(diff, 2)), end="")
      ore
      tree
      emu
```

**System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-`

main\Doc\library\[cpython-main][Doc][library]difflib.rst, line 282)

Unknown directive type "function".

```
.. function:: unified_diff(a, b, fromfile='', tofile='', fromfiledate='', tofiledate='', n=3, lineterm='\\n')

   Compare *a* and *b* (lists of strings); return a delta (a :term:`generator`
   generating the delta lines) in unified diff format.

   Unified diffs are a compact way of showing just the lines that have changed plus
   a few lines of context.  The changes are shown in an inline style (instead of
   separate before/after blocks).  The number of context lines is set by *n* which
   defaults to three.

   By default, the diff control lines (those with ``---``, ``+++``, or ``@@``) are
   created with a trailing newline.  This is helpful so that inputs created from
   :func:`io.IOBase.readlines` result in diffs that are suitable for use with
   :func:`io.IOBase.writelines` since both the inputs and outputs have trailing
   newlines.

   For inputs that do not have trailing newlines, set the *lineterm* argument to
   ``""`` so that the output will be uniformly newline free.

   The context diff format normally has a header for filenames and modification
   times.  Any or all of these may be specified using strings for *fromfile*,
   *tofile*, *fromfiledate*, and *tofiledate*.  The modification times are normally
   expressed in the ISO 8601 format. If not specified, the
   strings default to blanks.


      >>> s1 = ['bacon\n', 'eggs\n', 'ham\n', 'guido\n']
      >>> s2 = ['python\n', 'eggy\n', 'hamster\n', 'guido\n']
      >>> sys.stdout.writelines(unified_diff(s1, s2, fromfile='before.py', tofile='after.py'))
      --- before.py
      +++ after.py
      @@ -1,4 +1,4 @@
      -bacon
      -eggs
      -ham
      +python
      +eggy
      +hamster
       guido

   See :ref:`difflib-interface` for a more detailed example.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst, line 324)

Unknown directive type "function".

```
.. function:: diff_bytes(dfunc, a, b, fromfile=b'', tofile=b'', fromfiledate=b'', tofiledate=b'', n=3, lineterm=

   Compare *a* and *b* (lists of bytes objects) using *dfunc*; yield a
   sequence of delta lines (also bytes) in the format returned by *dfunc*.
   *dfunc* must be a callable, typically either :func:`unified_diff` or
   :func:`context_diff`.

   Allows you to compare data with unknown or inconsistent encoding. All
   inputs except *n* must be bytes objects, not str. Works by losslessly
   converting all inputs (except *n*) to str, and calling ``dfunc(a, b,
   fromfile, tofile, fromfiledate, tofiledate, n, lineterm)``. The output of
   *dfunc* is then converted back to bytes, so the delta lines that you
   receive have the same unknown/inconsistent encodings as *a* and *b*.

   .. versionadded:: 3.5
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst, line 340)

Unknown directive type "function".

```
.. function:: IS_LINE_JUNK(line)

   Return ``True`` for ignorable lines.  The line *line* is ignorable if *line* is
   blank or contains a single ``'#'``, otherwise it is not ignorable.  Used as a
   default for parameter *linejunk* in :func:`ndiff` in older versions.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst, line 347)

Unknown directive type "function".

```
.. function:: IS_CHARACTER_JUNK(ch)

   Return ``True`` for ignorable characters.  The character *ch* is ignorable if *ch*
   is a space or tab, otherwise it is not ignorable.  Used as a default for
   parameter *charjunk* in :func:`ndiff`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst, line 354)

Unknown directive type "seealso".

```
.. seealso::

   `Pattern Matching: The Gestalt Approach <http://www.drdobbs.com/database/pattern-matching-the-gestalt-approac
      Discussion of a similar algorithm by John W. Ratcliff and D. E. Metzener. This
      was published in `Dr. Dobb's Journal <http://www.drdobbs.com/>`_ in July, 1988.
```

SequenceMatcher Objects

The :class:`SequenceMatcher` class has this constructor:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 366); *backlink*

Unknown interpreted text role "class".

Optional argument *isjunk* must be `None` (the default) or a one-argument function that takes a sequence element and returns true if and only if the element is "junk" and should be ignored. Passing `None` for *isjunk* is equivalent to passing `lambda x: False`; in other words, no elements are ignored. For example, pass:

```
lambda x: x in " \t"
```

if you're comparing lines as sequences of characters, and don't want to synch up on blanks or hard tabs.

The optional arguments *a* and *b* are sequences to be compared; both default to empty strings. The elements of both sequences must be :term:`hashable`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 382); *backlink*

Unknown interpreted text role "term".

The optional argument *autojunk* can be used to disable the automatic junk heuristic.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 388)

Unknown directive type "versionadded".

```
.. versionadded:: 3.2
   The *autojunk* parameter.
```

SequenceMatcher objects get three data attributes: *bjunk* is the set of elements of *b* for which *isjunk* is `True`; *bpopular* is the set of non-junk elements considered popular by the heuristic (if it is not disabled); *b2j* is a dict mapping the remaining elements of *b* to a list of positions where they occur. All three are reset whenever *b* is reset with :meth:`set_seqs` or :meth:`set_seq2`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 391); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 391); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 398)

Unknown directive type "versionadded".

```
.. versionadded:: 3.2
   The *bjunk* and *bpopular* attributes.
```

:class:`SequenceMatcher` objects have the following methods:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 401); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 403)

Unknown directive type "method".

```
.. method:: set_seqs(a, b)

   Set the two sequences to be compared.
```

:class:`SequenceMatcher` computes and caches detailed information about the second sequence, so if you want to compare one sequence against many sequences, use :meth:`set_seq2` to set the commonly used sequence once and call :meth:`set_seq1` repeatedly, once for each of the other sequences.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst, line 407); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst, line 407); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst, line 407); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst, line 413)

Unknown directive type "method".

```
.. method:: set_seq1(a)

   Set the first sequence to be compared.  The second sequence to be compared
   is not changed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst, line 419)

Unknown directive type "method".

```
.. method:: set_seq2(b)

   Set the second sequence to be compared.  The first sequence to be compared
   is not changed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst, line 425)

Unknown directive type "method".

```
.. method:: find_longest_match(alo=0, ahi=None, blo=0, bhi=None)

   Find longest matching block in ``a[alo:ahi]`` and ``b[blo:bhi]``.

   If *isjunk* was omitted or ``None``, :meth:`find_longest_match` returns
   ``(i, j, k)`` such that ``a[i:i+k]`` is equal to ``b[j:j+k]``, where ``alo
   <= i <= i+k <= ahi`` and ``blo <= j <= j+k <= bhi``. For all ``(i', j',
   k')`` meeting those conditions, the additional conditions ``k >= k'``, ``i
   <= i'``, and if ``i == i'``, ``j <= j'`` are also met. In other words, of
   all maximal matching blocks, return one that starts earliest in *a*, and
   of all those maximal matching blocks that start earliest in *a*, return
   the one that starts earliest in *b*.

      >>> s = SequenceMatcher(None, " abcd", "abcd abcd")
      >>> s.find_longest_match(0, 5, 0, 9)
      Match(a=0, b=4, size=5)

   If *isjunk* was provided, first the longest matching block is determined
   as above, but with the additional restriction that no junk element appears
   in the block.  Then that block is extended as far as possible by matching
   (only) junk elements on both sides. So the resulting block never matches
   on junk except as identical junk happens to be adjacent to an interesting
   match.

   Here's the same example as before, but considering blanks to be junk. That
   prevents ``' abcd'`` from matching the ``' abcd'`` at the tail end of the
   second sequence directly.  Instead only the ``'abcd'`` can match, and
   matches the leftmost ``'abcd'`` in the second sequence:

      >>> s = SequenceMatcher(lambda x: x==" ", " abcd", "abcd abcd")
      >>> s.find_longest_match(0, 5, 0, 9)
      Match(a=1, b=0, size=4)

   If no blocks match, this returns ``(alo, blo, 0)``.

   This method returns a :term:`named tuple` ``Match(a, b, size)``.

   .. versionchanged:: 3.9
      Added default arguments.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst, line 466)

Unknown directive type "method".

```
.. method:: get_matching_blocks()

   Return list of triples describing non-overlapping matching subsequences.
   Each triple is of the form ``(i, j, n)``,
   and means that ``a[i:i+n] == b[j:j+n]``.  The
   triples are monotonically increasing in *i* and *j*.

   The last triple is a dummy, and has the value ``(len(a), len(b), 0)``.  It
   is the only triple with ``n == 0``.  If ``(i, j, n)`` and ``(i', j', n')``
   are adjacent triples in the list, and the second is not the last triple in
   the list, then ``i+n < i'`` or ``j+n < j'``; in other words, adjacent
   triples always describe non-adjacent equal blocks.

   .. XXX Explain why a dummy is used!

   .. doctest::

      >>> s = SequenceMatcher(None, "abxcd", "abcd")
      >>> s.get_matching_blocks()
      [Match(a=0, b=0, size=2), Match(a=3, b=2, size=2), Match(a=5, b=4, size=0)]
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 488)

Unknown directive type "method".

```
   .. method:: get_opcodes()

      Return list of 5-tuples describing how to turn *a* into *b*. Each tuple is
      of the form ``(tag, i1, i2, j1, j2)``.  The first tuple has ``i1 == j1 ==
      0``, and remaining tuples have *i1* equal to the *i2* from the preceding
      tuple, and, likewise, *j1* equal to the previous *j2*.

      The *tag* values are strings, with these meanings:

      +---------------+---------------------------------------------+
      | Value         | Meaning                                     |
      +===============+=============================================+
      | ``'replace'`` | ``a[i1:i2]`` should be replaced by          |
      |               | ``b[j1:j2]``.                               |
      +---------------+---------------------------------------------+
      | ``'delete'``  | ``a[i1:i2]`` should be deleted.  Note that  |
      |               | ``j1 == j2`` in this case.                  |
      +---------------+---------------------------------------------+
      | ``'insert'``  | ``b[j1:j2]`` should be inserted at          |
      |               | ``a[i1:i1]``. Note that ``i1 == i2`` in     |
      |               | this case.                                  |
      +---------------+---------------------------------------------+
      | ``'equal'``   | ``a[i1:i2] == b[j1:j2]`` (the sub-sequences |
      |               | are equal).                                 |
      +---------------+---------------------------------------------+

      For example::

        >>> a = "qabxcd"
        >>> b = "abycdf"
        >>> s = SequenceMatcher(None, a, b)
        >>> for tag, i1, i2, j1, j2 in s.get_opcodes():
        ...     print('{:7}   a[{}:{}] --> b[{}:{}] {!r:>8} --> {!r}'.format(
        ...         tag, i1, i2, j1, j2, a[i1:i2], b[j1:j2]))
        delete    a[0:1] --> b[0:0]      'q' --> ''
        equal     a[1:3] --> b[0:2]     'ab' --> 'ab'
        replace   a[3:4] --> b[2:3]      'x' --> 'y'
        equal     a[4:6] --> b[3:5]     'cd' --> 'cd'
        insert    a[6:6] --> b[5:6]       '' --> 'f'
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 529)

Unknown directive type "method".

```
   .. method:: get_grouped_opcodes(n=3)

      Return a :term:`generator` of groups with up to *n* lines of context.

      Starting with the groups returned by :meth:`get_opcodes`, this method
      splits out smaller change clusters and eliminates intervening ranges which
      have no changes.

      The groups are returned in the same format as :meth:`get_opcodes`.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 540)

Unknown directive type "method".

```
   .. method:: ratio()

      Return a measure of the sequences' similarity as a float in the range [0,
      1].

      Where T is the total number of elements in both sequences, and M is the
      number of matches, this is 2.0\*M / T. Note that this is ``1.0`` if the
      sequences are identical, and ``0.0`` if they have nothing in common.

      This is expensive to compute if :meth:`get_matching_blocks` or
      :meth:`get_opcodes` hasn't already been called, in which case you may want
      to try :meth:`quick_ratio` or :meth:`real_quick_ratio` first to get an
      upper bound.

      .. note::

         Caution: The result of a :meth:`ratio` call may depend on the order of
         the arguments. For instance::

            >>> SequenceMatcher(None, 'tide', 'diet').ratio()
            0.25
            >>> SequenceMatcher(None, 'diet', 'tide').ratio()
            0.5
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 565)

Unknown directive type "method".

```
.. method:: quick_ratio()

   Return an upper bound on :meth:`ratio` relatively quickly.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 570)

Unknown directive type "method".

```
.. method:: real_quick_ratio()

   Return an upper bound on :meth:`ratio` very quickly.
```

The three methods that return the ratio of matching to total characters can give different results due to differing levels of approximation, although :meth:`quick_ratio` and :meth:`real_quick_ratio` are always at least as large as :meth:`ratio`:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 575); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 575); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 575); *backlink*

Unknown interpreted text role "meth".

```
>>> s = SequenceMatcher(None, "abcd", "bcde")
>>> s.ratio()
0.75
>>> s.quick_ratio()
0.75
>>> s.real_quick_ratio()
1.0
```

SequenceMatcher Examples

This example compares two strings, considering blanks to be "junk":

```
>>> s = SequenceMatcher(lambda x: x == " ",
...                     "private Thread currentThread;",
...                     "private volatile Thread currentThread;")
```

:meth:`ratio` returns a float in [0, 1], measuring the similarity of the sequences. As a rule of thumb, a :meth:`ratio` value over 0.6 means the sequences are close matches:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 600); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 600); *backlink*

Unknown interpreted text role "meth".

```
>>> print(round(s.ratio(), 3))
0.866
```

If you're only interested in where the sequences match, :meth:`get_matching_blocks` is handy:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 607); *backlink*

Unknown interpreted text role "meth".

```
>>> for block in s.get_matching_blocks():
...     print("a[%d] and b[%d] match for %d elements" % block)
a[0] and b[0] match for 8 elements
a[8] and b[17] match for 21 elements
a[29] and b[38] match for 0 elements
```

Note that the last tuple returned by :meth:`get_matching_blocks` is always a dummy, `(len(a), len(b), 0)`, and this is the only case in which the last tuple element (number of elements matched) is `0`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, **line 616**); ***backlink***

Unknown interpreted text role "meth".

If you want to know how to change the first sequence into the second, use :meth:`get_opcodes`:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, **line 620**); ***backlink***

Unknown interpreted text role "meth".

```
>>> for opcode in s.get_opcodes():
...     print("%6s a[%d:%d] b[%d:%d]" % opcode)
 equal a[0:8] b[0:8]
insert a[8:8] b[8:17]
 equal a[8:29] b[17:38]
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, **line 629**)

Unknown directive type "seealso".

```
.. seealso::

   * The :func:`get_close_matches` function in this module which shows how
     simple code building on :class:`SequenceMatcher` can be used to do useful
     work.

   * `Simple version control recipe
     <https://code.activestate.com/recipes/576729/>`_ for a small application
     built with :class:`SequenceMatcher`.
```

Differ Objects

Note that :class:`Differ`-generated deltas make no claim to be **minimal** diffs. To the contrary, minimal diffs are often counter-intuitive, because they synch up anywhere possible, sometimes accidental matches 100 pages apart. Restricting synch points to contiguous matches preserves some notion of locality, at the occasional cost of producing a longer diff.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, **line 645**); ***backlink***

Unknown interpreted text role "class".

The :class:`Differ` class has this constructor:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, **line 651**); ***backlink***

Unknown interpreted text role "class".

Optional keyword parameters *linejunk* and *charjunk* are for filter functions (or `None`):

linejunk: A function that accepts a single string argument, and returns true if the string is junk. The default is `None`, meaning that no line is considered junk.

charjunk: A function that accepts a single character argument (a string of length 1), and returns true if the character is junk. The default is `None`, meaning that no character is considered junk.

These junk-filtering functions speed up matching to find differences and do not cause any differing lines or characters to be ignored. Read the description of the :meth:`~SequenceMatcher.find_longest_match` method's *isjunk* parameter for an explanation.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, **line 668**); ***backlink***

Unknown interpreted text role "meth".

:class:`Differ` objects are used (deltas generated) via a single method:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, **line 674**); ***backlink***

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-

main\Doc\library\[cpython-main][Doc][library]difflib.rst, line 677)

Unknown directive type "method".

```
.. method:: Differ.compare(a, b)

   Compare two sequences of lines, and generate the delta (a sequence of lines).

   Each sequence must contain individual single-line strings ending with
   newlines.  Such sequences can be obtained from the
   :meth:`~io.IOBase.readlines` method of file-like objects.  The delta
   generated also consists of newline-terminated strings, ready to be
   printed as-is via the :meth:`~io.IOBase.writelines` method of a
   file-like object.
```

Differ Example

This example compares two texts. First we set up the texts, sequences of individual single-line strings ending with newlines (such sequences can also be obtained from the :meth:`~io.BaseIO.readlines` method of file-like objects):

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst, line 694); *backlink*

Unknown interpreted text role "meth".

```
>>> text1 = '''  1. Beautiful is better than ugly.
...   2. Explicit is better than implicit.
...   3. Simple is better than complex.
...   4. Complex is better than complicated.
... '''.splitlines(keepends=True)
>>> len(text1)
4
>>> text1[0][-1]
'\n'
>>> text2 = '''  1. Beautiful is better than ugly.
...   3.   Simple is better than complex.
...   4. Complicated is better than complex.
...   5. Flat is better than nested.
... '''.splitlines(keepends=True)
```

Next we instantiate a Differ object:

```
>>> d = Differ()
```

Note that when instantiating a :class:`Differ` object we may pass functions to filter out line and character "junk." See the :meth:`Differ` constructor for details.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst, line 717); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst, line 717); *backlink*

Unknown interpreted text role "meth".

Finally, we compare the two:

```
>>> result = list(d.compare(text1, text2))
```

`result` is a list of strings, so let's pretty-print it:

```
>>> from pprint import pprint
>>> pprint(result)
['    1. Beautiful is better than ugly.\n',
 '-   2. Explicit is better than implicit.\n',
 '-   3. Simple is better than complex.\n',
 '+   3.   Simple is better than complex.\n',
 '?     ++\n',
 '-   4. Complex is better than complicated.\n',
 '?            ^                     ---- ^\n',
 '+   4. Complicated is better than complex.\n',
 '?           ++++ ^                      ^\n',
 '+   5. Flat is better than nested.\n']
```

As a single multi-line string it looks like this:

```
>>> import sys
>>> sys.stdout.writelines(result)
    1. Beautiful is better than ugly.
-   2. Explicit is better than implicit.
-   3. Simple is better than complex.
+   3.   Simple is better than complex.
?     ++
-   4. Complex is better than complicated.
?            ^                     ---- ^
+   4. Complicated is better than complex.
?           ++++ ^                      ^
+   5. Flat is better than nested.
```

A command-line interface to difflib

This example shows how to use difflib to create a `diff`-like utility. It is also contained in the Python source distribution, as :file:`Tools/scripts/diff.py`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 761); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]difflib.rst`, line 765)

Unknown directive type "literalinclude".

```
.. literalinclude:: ../../Tools/scripts/diff.py
```