**[Python标准库]difflib——比较序列**

原创 2015年12月18日 19:50:01

* 6433

**[Python标准库]difflib——比较序列**  
        作用：比较序列（特别是文本行）。  
        Python 版本：2.1 及以后版本  
        difflib 模块包含一些用来计算和处理序列之间差异的工具。它对于比较文本尤其有用，其中包含的函数可以使用多种常用差异格式生成报告。  
        本节中的例子都会使用 difflib\_data.py 模块中以下这个常用的测试数据：

**[python]** [view plain](https://blog.csdn.net/dapeng0802/article/details/50347485) [copy](https://blog.csdn.net/dapeng0802/article/details/50347485)

1. text1 = """Lorem ipsum dolor sit amet, consectetuer adipiscing
2. elit. Integer eu lacus accumsan arcu fermentum euismod. Donec
3. pulvinar porttitor tellus. Aliquam venenatis. Donec facilisis
4. pharetra tortor.  In nec mauris eget megna consequat
5. convallis. Nam sed sem vitae odio pellentesque interdum. Sed
6. consequat viverra nisl. Suspendisse arcu metus, blandit quis,
7. rhoncus ac, pharetra eget, velit. Mauris urna. Morbi nonummy
8. molestie orci. Praesent nisi elit, fringilla ac, suscipit non,
9. tristique vel, mauris. Curabitur vel lorem id nisl porta
10. adioiscing. Suspendisse eu lectus. In nunc. Duis vulputate
11. tristique enim. Donec quis lectus a justo imperdiet tempus."""
12. text1\_lines = text1.splitlines()
14. text2 = """Lorem ipsum dolor sit amet, consectetuer adipiscing
15. elit. Integer eu lacus accumsan arcu fermentum euismod. Donec
16. pulvinar, porttitor tellus. Aliquam venenatis. Donec facilisis
17. pharetra tortor. In nec mauris eget megna consequat
18. convallis. Nam cras vitae mi vitae odio pellentesque interdum. Sed
19. consequat viverra nisl. Suspendisse arcu metus, blandit quis,
20. rhoncus ac, pharetra eget, velit. Mauris urna. Morbi nonummy
21. molestie orci. Praesent nisi elit, fringilla ac, suscipit non,
22. tristique vel, mauris. Curabitur vel lorem id nisl porta
23. adioiscing. Duis vulputate tristique enim. Donec quis lectus a justo imperdiet tempus.  Suspendisse eu lectus. In nunc."""
24. text2\_lines = text2.splitlines()

**比较文本体**  
        Differ 类用于处理文本序列，并生成人类可读的差异（deltas）或更改指令，包括各行中的差异。Differ 生成的默认输入与 UNIX 下的 diff 命令行工具类似，包括两个列表的原始输入值（包含共同的值），以及指示做了哪些更改的标记数据。

* 有 “-” 前缀的行指示这些行在第一个序列中，但不包含在第二个序列中。
* 有 “+” 前缀的行在第二个序列中，但不包含在第一个序列中。
* 如果某一行在不同版本之间存在增量差异，会使用一个以 “？” 为前缀的额外的行强调新版本中的变更。
* 如果一行未改变，会输出该行，而且其左边有一个额外的空格，使它与其他可能有差异的输出对齐。

        将文本传入 compare() 之前先分解为由单个文本行构成的序列，与传入大字符串相比，这样可以生成更可读的输出。

**[python]** [view plain](https://blog.csdn.net/dapeng0802/article/details/50347485) [copy](https://blog.csdn.net/dapeng0802/article/details/50347485)

1. **import** difflib
2. **from** difflib\_data **import** \*
4. d = difflib.Differ()
5. diff = d.compare(text1\_lines, text2\_lines)
6. **print** '\n'.join(diff)

        示例数据中两个文本段的开始部分是一样的，所以第一行会直接输出而没有任何额外标注。

**[python]** [view plain](https://blog.csdn.net/dapeng0802/article/details/50347485) [copy](https://blog.csdn.net/dapeng0802/article/details/50347485)

1. Lorem ipsum dolor sit amet, consectetuer adipiscing
2. elit. Integer eu lacus accumsan arcu fermentum euismod. Donec

        数据的第三行存在变化，修改后的文本中包含有一个逗号。这两个版本的数据行都会输出，而且第五行上的额外信息会显示文本中哪一列有修改，这里显示出增加了 “,” 字符。

**[python]** [view plain](https://blog.csdn.net/dapeng0802/article/details/50347485) [copy](https://blog.csdn.net/dapeng0802/article/details/50347485)

1. - pulvinar porttitor tellus. Aliquam venenatis. Donec facilisis
2. + pulvinar, porttitor tellus. Aliquam venenatis. Donec facilisis
3. ?         +

        输出中接下来几行显示删除了一个多余的空格。

**[python]** [view plain](https://blog.csdn.net/dapeng0802/article/details/50347485) [copy](https://blog.csdn.net/dapeng0802/article/details/50347485)

1. - pharetra tortor.  In nec mauris eget megna consequat
2. ?                 -
4. + pharetra tortor. In nec mauris eget megna consequat

        接下来有一个更为复杂的变更，替换了一个短语中的多个单词。

**[python]** [view plain](https://blog.csdn.net/dapeng0802/article/details/50347485) [copy](https://blog.csdn.net/dapeng0802/article/details/50347485)

1. - convallis. Nam sed sem vitae odio pellentesque interdum. Sed
2. ?                  - --
4. + convallis. Nam cras vitae mi vitae odio pellentesque interdum. Sed
5. ?                +++ +++++   +

        段落中下一句变化很大，所以表示差异时完全删除了老版本，而增加了新版本。

**[python]** [view plain](https://blog.csdn.net/dapeng0802/article/details/50347485) [copy](https://blog.csdn.net/dapeng0802/article/details/50347485)

1. consequat viverra nisl. Suspendisse arcu metus, blandit quis,
2. rhoncus ac, pharetra eget, velit. Mauris urna. Morbi nonummy
3. molestie orci. Praesent nisi elit, fringilla ac, suscipit non,
4. tristique vel, mauris. Curabitur vel lorem id nisl porta
5. + adioiscing. Duis vulputate tristique enim. Donec quis lectus a justo imperdiet tempus.  Suspendisse eu lectus. In nunc.
6. - adioiscing. Suspendisse eu lectus. In nunc. Duis vulputate
7. - tristique enim. Donec quis lectus a justo imperdiet tempus.

        ndiff() 函数生成的输出基本上相同，会特别“加工”来处理文本数据，并删除输入中的“噪声”。

**其他输出格式**  
        Differ 类会显示所有输入行，统一差异格式（unified diff）则不同，它只包含已修改的文本行和一些上下文。Python 2.3 中增加了 unified\_diff() 函数来生成这种输出。

**[python]** [view plain](https://blog.csdn.net/dapeng0802/article/details/50347485) [copy](https://blog.csdn.net/dapeng0802/article/details/50347485)

1. **import** difflib
2. **from** difflib\_data **import** \*
4. diff = difflib.unified\_diff(text1\_lines, text2\_lines, lineterm='',)
5. **print** '\n'.join(list(diff))

        lineterm 参数用来告诉 unified\_diff() 不必为它返回的控制行追加换行符，因为输入行不包括这些换行符。输出时所有行都会增加换行符。对于 subversion 或其他版本控制工具的用户来说，输出看上去应该很熟悉。

**[python]** [view plain](https://blog.csdn.net/dapeng0802/article/details/50347485) [copy](https://blog.csdn.net/dapeng0802/article/details/50347485)

1. ---
2. +++
3. @@ -1,11 +1,10 @@
4. Lorem ipsum dolor sit amet, consectetuer adipiscing
5. elit. Integer eu lacus accumsan arcu fermentum euismod. Donec
6. -pulvinar porttitor tellus. Aliquam venenatis. Donec facilisis
7. -pharetra tortor.  In nec mauris eget megna consequat
8. -convallis. Nam sed sem vitae odio pellentesque interdum. Sed
9. +pulvinar, porttitor tellus. Aliquam venenatis. Donec facilisis
10. +pharetra tortor. In nec mauris eget megna consequat
11. +convallis. Nam cras vitae mi vitae odio pellentesque interdum. Sed
12. consequat viverra nisl. Suspendisse arcu metus, blandit quis,
13. rhoncus ac, pharetra eget, velit. Mauris urna. Morbi nonummy
14. molestie orci. Praesent nisi elit, fringilla ac, suscipit non,
15. tristique vel, mauris. Curabitur vel lorem id nisl porta
16. -adioiscing. Suspendisse eu lectus. In nunc. Duis vulputate
17. -tristique enim. Donec quis lectus a justo imperdiet tempus.
18. +adioiscing. Duis vulputate tristique enim. Donec quis lectus a justo imperdiet tempus.  Suspendisse eu lectus. In nunc.

        使用 context\_diff() 会生成类似可读的输出。

**无用数据**  
        所有生成差异序列的函数都可以接受一些参数来指示应当忽略哪些行，以及应当忽略一行中的哪些字符。例如，可以用这些参数指定跳过一个文件两个版本中的标记或空白字符变更。

**[python]** [view plain](https://blog.csdn.net/dapeng0802/article/details/50347485) [copy](https://blog.csdn.net/dapeng0802/article/details/50347485)

1. # This example is adapted from the source for difflib.py
3. **from** difflib **import** SequenceMatcher
5. **def** show\_results(s):
6. i, j, k = s.find\_longest\_match(0, 5, 0, 9)
7. **print** '  i = %d' % i
8. **print** '  j = %d' % j
9. **print** '  k = %d' % k
10. **print** '  A[i:i+k] = %r' % A[i:i+k]
11. **print** '  B[j:j+k] = %r' % B[j:j+k]
13. A = " abcd"
14. B = "abcd abcd"
16. **print** 'A = %r' % A
17. **print** 'B = %r' % B
19. **print** '\nWithout junk detection:'
20. show\_results(SequenceMatcher(None, A, B))
22. **print** '\nTreat spaces as junk:'
23. show\_results(SequenceMatcher(**lambda** x: x==" ", A, B))

        默认情况下，Differ 不会显式忽略任何行或字符，而会依赖 SequenceMatcher 的能力检测噪声。ndiff() 的默认行为是忽略空格和制表符（tab）。

**[python]** [view plain](https://blog.csdn.net/dapeng0802/article/details/50347485) [copy](https://blog.csdn.net/dapeng0802/article/details/50347485)

1. A = ' abcd'
2. B = 'abcd abcd'
4. Without junk detection:
5. i = 0
6. j = 4
7. k = 5
8. A[i:i+k] = ' abcd'
9. B[j:j+k] = ' abcd'
11. Treat spaces as junk:
12. i = 1
13. j = 0
14. k = 4
15. A[i:i+k] = 'abcd'
16. B[j:j+k] = 'abcd'

**比较任意类型**  
        SequenceMatcher 类用于比较任意类型的两个序列，只要它们的值是可散列的。这个类使用一个算法来标识序列中最长的连续匹配块，并删除对实际数据没有贡献的无用值。

**[python]** [view plain](https://blog.csdn.net/dapeng0802/article/details/50347485) [copy](https://blog.csdn.net/dapeng0802/article/details/50347485)

1. **import** difflib
2. **from** difflib\_data **import** \*
4. s1 = [ 1, 2, 3, 5 ,6, 4 ]
5. s2 = [ 2, 3, 5, 4, 6, 1 ]
7. **print** 'Initial data:'
8. **print** 's1 =', s1
9. **print** 's2 =', s2
10. **print** 's1 == s2:', s1==s2
11. **print**
13. matcher = difflib.SequenceMatcher(None, s1, s2)
14. **for** tag, i1, i2, j1, j2 **in** reversed(matcher.get\_opcodes()):
16. **if** tag == 'delete':
17. **print** 'Remove %s from positions [%d:%d]' % \
18. (s1[i1:i2], i1, i2)
19. **del** s1[i1:i2]
20. **elif** tag == 'equal':
21. **print** 's1[%d:%d] and s2[%d:%d] are the same' % \
22. (i1, i2, j1, j2)
23. **elif** tag == 'insert':
24. **print** 'Insert %s from s2[%d:%d] into s1 at %d' % \
25. (s2[j1:j2], j1, j2, i1)
26. s1[i1:i2] = s2[j1:j2]
27. **elif** tag == 'replace':
28. **print** 'Replace %s from s1[%d:%d] with %s from s2[%d:%d]' % \
29. (s1[i1:i2], i1, i2, s2[j1:j2], j1, j2)
30. s1[i1:i2] = s2[j1:j2]
32. **print** '  s1 =', s1
34. **print** 's1 == s2:', s1==s2

        这个例子比较了两个整数列表，并使用 get\_opcodes() 得出将原列表转换为新列表的指令。这里以逆序应用所做的修改，使得添加和删除元素之后列表索引仍是正确的。

**[python]** [view plain](https://blog.csdn.net/dapeng0802/article/details/50347485) [copy](https://blog.csdn.net/dapeng0802/article/details/50347485)

1. Initial data:
2. s1 = [1, 2, 3, 5, 6, 4]
3. s2 = [2, 3, 5, 4, 6, 1]
4. s1 == s2: False
6. Replace [4] **from** s1[5:6] with [1] **from** s2[5:6]
7. s1 = [1, 2, 3, 5, 6, 1]
8. s1[4:5] **and** s2[4:5] are the same
9. s1 = [1, 2, 3, 5, 6, 1]
10. Insert [4] **from** s2[3:4] into s1 at 4
11. s1 = [1, 2, 3, 5, 4, 6, 1]
12. s1[1:4] **and** s2[0:3] are the same
13. s1 = [1, 2, 3, 5, 4, 6, 1]
14. Remove [1] **from** positions [0:1]
15. s1 = [2, 3, 5, 4, 6, 1]
16. s1 == s2: True

        SequenceMatcher 用于处理定制类以及内置类型，前提是它们必须是可散列的。