API Documentation

API Documentation

September 13, 2012

Contents

1 Package htmldiff

1.1 Modules

- **font_lookup** *(Section 2, p. 3)*
- **htmldiff** *(Section 3, p. 4)*

1.2 Variables

Name	Description
___package___	**Value:** `None`

2 Module htmldiff.font_lookup

2.1 Functions

get_spacing(*string*, *font_type*)

Given a string & font, return approximate spacing for making more appropriate whitespace than would be normally generated from just counting characters and replacing with spaces. Currently only has a lookup table for Times New Roman. Possible to add support for others at a later point just by adding them to the font dictionary. The font sizes are an arbitrary unit and merely approximates.

This will get things closer, but outside of test rendering each and manually calculating space and doing the conversion with that, this at least works to get you close.

Parameters

`string:` a string to calculate whitespace for
(type=string)

`font_type:` type of font to calculate space for

`font_tye:` *(type=string)*

Return Value

integer of space characters to use

2.2 Variables

Name	Description
times_new_roman	**Value:** `{' ': 20, '0': 41, '1': 41, '2': 41, '3': 41, '4': 41, '5...`
fonts	**Value:** `{'times new roman': {' ': 20, '0': 41, '1': 41, '2': 41, ...`
___package___	**Value:** `None`

3 Module htmldiff.htmldiff

3.1 Functions

strip_tags(*html_string*)

Remove all HTML tags from a given string of html

Parameters

`html_string:` string of html

(type=string)

Return Value

intial string stripped of html tags

htmlDecode(*s*)

Given a string of html, decode entities

Parameters

`s:` string of html to decode

(type=string)

Return Value

string of html with decoded entities

htmlEncode(*s*, *esc*=`<function escape at 0x1667f50>`)

isJunk(*x*)

Used for the faster but less accurate mode. Original comment said:

Note: Just returning false here gives a generally more accurate but much shower and more noisy result.

False is now set with the -a switch so this function always returns the regex matches or lowercase.

Parameters

`x:` string to match against

(type=string)

Return Value

regex matched or lowercased x

htmldiff(*source1*, *source2*, *accurate_mode*, *addStylesheet*=`False`)

Return the difference between two pieces of HTML

```
>>> htmldiff('test1', 'test2')
'<span class="delete">test1 </span> <span class="insert">test2 </span> '
>>> htmldiff('test1', 'test1')
'test1 '
>>> htmldiff('<b>test1</b>', '<i>test1</i>')
'<span class="tagDelete">delete: <tt>&lt;b&gt;</tt></span> <span class="tagInsert">insert: <tt>&lt;i&gt;</tt>
```

diffStrings(*orig*, *new*, *accurate_mode*)

Given two strings of html, return a diffed string.

Parameters

`orig`: original string for comparison
(type=string)

`new`: new string for comparision against original string
(type=string)

`accurate_moode`: use accurate mode or not
(type=boolean)

Return Value

string containing diffed html

diffFiles(*f1*, *f2*, *accurate_mode*)

Given two files, open them to variables and pass them to diffStrings for diffing.

Parameters

`f1`: initial file to diff against
(type=object)

`f2`: new file to compare to f1
(type=object)

`accurate_mode`: use accurate mode or not
(type=boolean)

Return Value

string containing diffed html from f1 and f2

whitespacegen(*spaces*)

From a certain number of spaces, provide an html entity for non breaking spaces in an html document.

Parameters

`spaces:` Number of html space entities to return as string

(type=integer)

Return Value

string containing html space entities ( ) wrapped in a html span that properly wraps the whitespace.

span_to_whitespace(*html_string*, *span*)

Given an html string and a span tag name, parse the html and find the document areas containing those pieces and then replace them with nonbreaking whitespace html entities.

Parameters

`html_string:` string of html to parse

(type=string)

`string:` the span class to parse for

`span:` *(type=string)*

Return Value

html string with specified span replaced with whitespace

gen_side_by_side(*file_string*)

Given an html file as a string, return a new html file with side by side differences displayed in a single html file.

Parameters

`file_string:` string of html to convert

(type=string)

Return Value

string of html with side-by-side diffs

split_html(*html_string*)

Divides an html document into three seperate strings and returns each of these. The first part is everything up to and including the <body> tag. The next is everything inside of the body tags. The third is everything outside of and including the </body> tag.

Parameters

`html_string:` html document in string form @ return: three strings start, body, and ending

(type=string)

diff()

main()

3.2 Variables

Name	Description
commentRE	**Value:** `re.compile(r'(?s)<!-.*?->')`
tagRE	**Value:** `re.compile(r'(?s)<script.*?>.*?</script>|.*?>')`
headRE	**Value:** `re.compile(r'(?is)<\s*head\s*>')`
wsRE	**Value:** `re.compile(r'^([ \n\r\t]| )+$')`
stopwords	**Value:** `['I', 'a', 'about', 'an', 'and', 'are', 'as', 'at', 'be',...`
___package___	**Value:** `'htmldiff'`

3.3 Class TagStrip

markupbase.ParserBase
HTMLParser.HTMLParser
htmldiff.htmldiff.TagStrip

Subclass of HTMLParser used to strip html tags from strings

3.3.1 Methods

___init___(*self*)

Initialize and reset this instance.

Overrides: markupbase.ParserBase.___init___

handle_data(*self*, *s*)

Overrides: HTMLParser.HTMLParser.handle_data

get_stripped_string(*self*)

Inherited from HTMLParser.HTMLParser

check_for_whole_start_tag(), clear_cdata_mode(), close(), error(), feed(), get_starttag_text(), goahead(), handle_charref(), handle_comment(), handle_decl(), handle_endtag(), handle_entityref(), handle_pi(), handle_startendtag(), handle_starttag(), parse_bogus_comment(), parse_endtag(), parse_html_declaration(), parse_pi(), parse_starttag(), reset(), set_cdata_mode(), unescape(), unknown_decl()

Inherited from markupbase.ParserBase

getpos(), parse_comment(), parse_declaration(), parse_marked_section(), updatepos()

3.3.2 Class Variables

Name	Description
Inherited from HTMLParser.HTMLParser	
CDATA_CONTENT_ELEMENTS, entitydefs	

3.4 Class HTMLMatcher

difflib.SequenceMatcher
└
htmldiff.htmldiff.HTMLMatcher

Known Subclasses: htmldiff.htmldiff.NoTagHTMLMatcher, htmldiff.htmldiff.TextMatcher

3.4.1 Methods

___init___(*self*, *source1*, *source2*, *accurate_mode*)

```
Construct a SequenceMatcher.

Optional arg isjunk is None (the default), or a one-argument
function that takes a sequence element and returns true iff the
element is junk.  None is equivalent to passing "lambda x: 0", i.e.
no elements are considered to be junk.  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.

Optional arg a is the first of two sequences to be compared.  By
default, an empty string.  The elements of a must be hashable.  See
also .set_seqs() and .set_seq1().

Optional arg b is the second of two sequences to be compared.  By
default, an empty string.  The elements of b must be hashable. See
also .set_seqs() and .set_seq2().

Optional arg autojunk should be set to False to disable the
"automatic junk heuristic" that treats popular elements as junk
(see module documentation for more information).
```

Overrides: difflib.SequenceMatcher.___init___ extit(inherited documentation)

set_seq1(*self*, *a*)

Set the first sequence to be compared.

The second sequence to be compared is not changed.

```
>>> s = SequenceMatcher(None, "abcd", "bcde")
>>> s.ratio()
0.75
>>> s.set_seq1("bcde")
>>> s.ratio()
1.0
>>>
```

SequenceMatcher computes and caches detailed information about the second sequence, so if you want to compare one sequence S against many sequences, use .set_seq2(S) once and call .set_seq1(x) repeatedly for each of the other sequences.

See also set_seqs() and set_seq2().

Overrides: difflib.SequenceMatcher.set_seq1 extit(inherited documentation)

set_seq2(*self*, *b*)

Set the second sequence to be compared.

The first sequence to be compared is not changed.

```
>>> s = SequenceMatcher(None, "abcd", "bcde")
>>> s.ratio()
0.75
>>> s.set_seq2("abcd")
>>> s.ratio()
1.0
>>>
```

SequenceMatcher computes and caches detailed information about the second sequence, so if you want to compare one sequence S against many sequences, use .set_seq2(S) once and call .set_seq1(x) repeatedly for each of the other sequences.

See also set_seqs() and set_seq1().

Overrides: difflib.SequenceMatcher.set_seq2 extit(inherited documentation)

splitTags(*self*, *t*)

splitWords(*self*, *t*)

splitHTML(*self*, *t*)

htmlDiff(*self*, *addStylesheet*=`True`)

isInvisibleChange(*self*, *seq1*, *seq2*)

textDelete(*self*, *lst*, *out*)

textInsert(*self*, *lst*, *out*)

outDelete(*self*, *s*, *out*)

outInsert(*self*, *s*, *out*)

stylesheet(*self*)

addStylesheet(*self*, *html*, *ss*)

startInsertText(*self*)

endInsertText(*self*)

startDeleteText(*self*)

endDeleteText(*self*)

formatInsertTag(*self*, *tag*)

formatDeleteTag(*self*, *tag*)

Inherited from difflib.SequenceMatcher

find_longest_match(), get_grouped_opcodes(), get_matching_blocks(), get_opcodes(), quick_ratio(), ratio(), real_quick_ratio(), set_seqs()

3.5 Class NoTagHTMLMatcher

difflib.SequenceMatcher

htmldiff.htmldiff.HTMLMatcher

htmldiff.htmldiff.NoTagHTMLMatcher

3.5.1 Methods

formatInsertTag(*self*, *tag*)

Overrides: htmldiff.htmldiff.HTMLMatcher.formatInsertTag

formatDeleteTag(*self*, *tag*)

Overrides: htmldiff.htmldiff.HTMLMatcher.formatDeleteTag

Inherited from htmldiff.htmldiff.HTMLMatcher(Section 3.4)

___init___(), addStylesheet(), endDeleteText(), endInsertText(), htmlDiff(), isInvisibleChange(), outDelete(), outInsert(), set_seq1(), set_seq2(), splitHTML(), splitTags(), splitWords(), startDeleteText(), startInsertText(), stylesheet(), textDelete(), textInsert()

Inherited from difflib.SequenceMatcher

find_longest_match(), get_grouped_opcodes(), get_matching_blocks(), get_opcodes(), quick_ratio(), ratio(), real_quick_ratio(), set_seqs()

3.6 Class TextMatcher

difflib.SequenceMatcher

htmldiff.htmldiff.HTMLMatcher

htmldiff.htmldiff.TextMatcher

3.6.1 Methods

set_seq1(*self*, *a*)

Set the first sequence to be compared.

The second sequence to be compared is not changed.

```
>>> s = SequenceMatcher(None, "abcd", "bcde")
>>> s.ratio()
0.75
>>> s.set_seq1("bcde")
>>> s.ratio()
1.0
>>>
```

SequenceMatcher computes and caches detailed information about the second sequence, so if you want to compare one sequence S against many sequences, use .set_seq2(S) once and call .set_seq1(x) repeatedly for each of the other sequences.

See also set_seqs() and set_seq2().

Overrides: difflib.SequenceMatcher.set_seq1 extit(inherited documentation)

set_seq2(*self*, *b*)

Set the second sequence to be compared.

The first sequence to be compared is not changed.

```
>>> s = SequenceMatcher(None, "abcd", "bcde")
>>> s.ratio()
0.75
>>> s.set_seq2("abcd")
>>> s.ratio()
1.0
>>>
```

SequenceMatcher computes and caches detailed information about the second sequence, so if you want to compare one sequence S against many sequences, use .set_seq2(S) once and call .set_seq1(x) repeatedly for each of the other sequences.

See also set_seqs() and set_seq1().

Overrides: difflib.SequenceMatcher.set_seq2 extit(inherited documentation)

htmlDiff(*self*, *addStylesheet*=`False`)

Overrides: htmldiff.htmldiff.HTMLMatcher.htmlDiff

writeLines(*self*, *lines*, *out*)

Inherited from htmldiff.htmldiff.HTMLMatcher(Section 3.4)

___init___(), addStylesheet(), endDeleteText(), endInsertText(), formatDeleteTag(), formatInsertTag(), isInvisibleChange(), outDelete(), outInsert(), splitHTML(), splitTags(), splitWords(), startDeleteText(), startInsertText(), stylesheet(), textDelete(), textInsert()

Inherited from difflib.SequenceMatcher

find_longest_match(), get_grouped_opcodes(), get_matching_blocks(), get_opcodes(), quick_ratio(), ratio(), real_quick_ratio(), set_seqs()

Index