

# test\_extract\_sizes.py

MIT License

Copyright 2023 auto\_anki

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Unit tests for extract sizes

Test cases for the extract sizes  
Includes testing None cases and valid PDFs

Asserts that when None is passed into get\_sizes,  
no error happens and None is returned

assert when no document is passed, it returns an empty list

Asserts that when None is passed into tag\_text,  
no error happens and None is returned

assert when no document is passed, it returns an empty dict

Tests the unique font size is as expected for Test1

Tests the groupings given a file

check the text

```
import json
import unittest
from extract_sizes import (get_sizes, tag_text, text_to_groupings)

class TestExtractSizes(unittest.TestCase):

    def test_font_doc_none(self):

        unique_fonts = get_sizes([])
        self.assertEqual(unique_fonts, [])

    def test_dict_fonts_none(self):

        dict_none = tag_text([], [])
        self.assertEqual(dict_none, [])

    def test_font_size_1(self):

        filename = "code/data/Test_1.json"
        with open(filename, encoding='utf-8') as file_pointer:
            doc = json.load(file_pointer)
            actual_fonts = get_sizes(doc)

            expected_fonts = [10, 11, 12, 14]
            self.assertEqual(expected_fonts, actual_fonts)

    def test_text_to_groupings_1(self):

        filename = "code/data/Test_1.json"

        with open(filename, encoding='utf-8') as file_pointer:
            doc = json.load(file_pointer)
            actual_dict = text_to_groupings(doc)

            page0 = {'Header': 'Possible title',
                    'Paragraph': '',
                    'slide': 0}
            page1 = {'Header': 'Possible heading',
                    'Paragraph': 'Possible subheading Possible paragraph',
                    'slide': 1}
            page2 = {'Header': 'Heading Another Heading',
                    'Paragraph': 'Paragraph',
                    'slide': 2}

            self.assertEqual(page0, actual_dict[0], f'Expected Page 0 to be {page0}')
            self.assertEqual(page1, actual_dict[1], f'Expected Page 1 to be {page1}')
            self.assertEqual(page2, actual_dict[2], f'Expected Page 2 to be {page2}')
            self.assertEqual([page0, page1, page2], actual_dict)
```

Tests the unique font size is as expected for Test2

Tests the groupings given a file

check the text

```
def test_font_size_2(self):

    filename = "code/data/Test_2.json"
    with open(filename, encoding='utf-8') as file_pointer:
        doc = json.load(file_pointer)
        actual_fonts = get_sizes(doc)

        expected_fonts = [12, 28, 44, 60]
        self.assertEqual(expected_fonts, actual_fonts)

def test_text_to_groupings_2(self):

    filename = "code/data/Test_2.json"
    with open(filename, encoding='utf-8') as file_pointer:
        doc = json.load(file_pointer)

        actual_dict = text_to_groupings(doc)

        page0 = {'Header': 'Test 1 : Computer Science',
                  'Paragraph': '',
                  'slide': 0}
        page1 = {'Header': 'Large Heading',
                  'Paragraph': 'Medium text Small text',
                  'slide': 1}
        page2 = {'Header': 'Large Heading 2',
                  'Paragraph': 'Medium text 2 Small text 2',
                  'slide': 2}
        self.assertEqual(page0, actual_dict[0], f'Expected Page 0 to be {page0}')
        self.assertEqual(page1, actual_dict[1], f'Expected Page 1 to be {page1}')
        self.assertEqual(page2, actual_dict[2], f'Expected Page 2 to be {page2}')
        self.assertEqual([page0, page1, page2], actual_dict)

if __name__ == '__main__':
    unittest.main()
```