Coverage report: 29%

Files | Functions | Classes

filter...

hide covered

coverage.py v7.6.1, created at 2024-09-23 23:24 -0300

File ▲	statements	missing	excluded	coverage
lib_ESW.py	87	18	0	79%
test_lib.py	43	0	0	100%
Total	**130**	**18**	**0**	**86%**

coverage.py v7.6.1, created at 2024-09-23 23:24 -0300

```python
from loguru import logger
from openpyxl import Workbook
import os
import sys
import tabula
import PyPDF2
import pandas as pd
import pdfplumber
from dataclasses import dataclass

logger.add('logfile.txt', format="{time} {level} {file} {message}", level="INFO")

@dataclass
class LibEsw:
    def __init__(self):
        self.dir_atual = os.path.dirname(os.path.abspath(sys.argv[0]))

    def pdf_para_tabela(self, arquivo):
        """
        Recebe o nome do arquivo PDF e retorna uma lista de tabelas extraídas.
        """
        logger.info('Iniciando a conversão do PDF para tabelas.')
        caminho_absoluto = os.path.join(self.dir_atual, arquivo)
        try:
            tabelas = tabula.read_pdf(caminho_absoluto, pages='all', stream=True, multiple_tables=True, encoding='ISO-8859-1')

            if not tabelas:
                logger.warning('Nenhuma tabela foi extraída do PDF.')
                return None

            logger.info('Conversão do PDF para tabelas concluída com sucesso.')
            return tabelas

        except Exception as e:
            logger.error(f'Erro ao converter PDF para tabelas: {e}')
            return None

    def tabela_para_excel(self, tabelas, arquivo):
        """
        Recebe uma lista de tabelas, o diretório para salvar o arquivo Excel e o nome do arquivo, e cria um arquivo Excel.
        """
        logger.info('Iniciando a conversão das tabelas para arquivo Excel.')
        caminho_absoluto = os.path.join(self.dir_atual, arquivo)
        try:
            if not tabelas:
                logger.warning('Nenhuma tabela foi fornecida.')
                return None

            workbook = Workbook()
            sheet = workbook.active

            for tabela in tabelas:
                linhas = tabela.values.tolist()

                for linha in linhas:
                    if linha:
                        sheet.append(linha)
                    else:
                        logger.warning('Linha vazia encontrada e ignorada.')

            workbook.save(caminho_absoluto)
            logger.info(f'Arquivo Excel salvo com sucesso em: {caminho_absoluto}')

        except Exception as e:
            logger.error(f'Erro ao salvar a tabela como Excel: {e}')

    def ler_pdf(caminho_pdf):
        try:
            print(f"Tentando abrir o arquivo PDF em: {caminho_pdf}")
            with pdfplumber.open(caminho_pdf) as pdf:
                pagina = pdf.pages[0]
                texto = pagina.extract_text()
                #print(f"Texto extraído: {texto}")
                return texto
        except FileNotFoundError:
            print(f"Arquivo não encontrado: {caminho_pdf}")
            return None
        except Exception as e:
            print(f"Erro ao ler o PDF: {e}")
            return None


    def criar_txt(self, conteudo, arquivo):
        """
        Cria um arquivo de texto a partir do conteúdo fornecido.
        """
        logger.info('Iniciando a criação do arquivo de texto.')
        caminho_absoluto = os.path.join(self.dir_atual, arquivo)
        try:
            with open(caminho_absoluto, 'w', encoding='utf-8') as arquivo_txt:
                arquivo_txt.write(conteudo)
            logger.info(f'Conteúdo escrito com sucesso no arquivo: {arquivo}')

        except Exception as e:
            logger.error(f'Erro ao criar o arquivo de texto: {e}')

    def ajustar_valor(self, valor):
        """
        Ajusta o valor de acordo com o formato esperado.
        """
        if pd.isna(valor):
            logger.info('Valor é NaN, retornando None.')
            return None

        valor_str = str(valor).strip()

        valor_str = valor_str.replace('.', '').replace(' ', '')

        if valor_str.endswith('D') or valor_str.endswith('C'):
            sufixo = valor_str[-1]
            valor_str = valor_str[:-1]
        else:
            sufixo = ''

        try:
            valor_ajustado = float(valor_str.replace(',', '.'))
            if sufixo == 'D':
                valor_ajustado = -valor_ajustado
            logger.info(f'Valor ajustado: {valor_ajustado}')
            return valor_ajustado
        except ValueError:
            logger.error(f'Valor inválido para conversão: {valor_str}')
            return None
```

Coverage for **test_lib.py**: 100%

43 statements | 43 run | 0 missing | 0 excluded


```python
import unittest
from unittest import TestCase
from unittest.mock import patch, mock_open, MagicMock
import os
import sys
import pandas as pd
from lib_ESW import LibEsw

caminho_pdf = os.getcwd()+"\\test\\test.pdf"

class TestLibEsw(unittest.TestCase):

    def setUp(self):
        self.lib_esw = LibEsw()
        self.test_pdf = "test.pdf"
        self.test_excel = "test.xlsx"
        self.test_txt = "test.txt"
        self.test_content = "This is a test content."
        self.test_table = pd.DataFrame({'col1': [1, 2], 'col2': [3, 4]})

    @patch('tabula.read_pdf')
    def test_pdf_para_tabela(self, mock_read_pdf):
        mock_read_pdf.return_value = [self.test_table]
        result = self.lib_esw.pdf_para_tabela(self.test_pdf)
        self.assertIsNotNone(result)
        self.assertEqual(len(result), 1)
        self.assertTrue(mock_read_pdf.called)

    @patch('openpyxl.Workbook.save')
    def test_tabela_para_excel(self, mock_save):
        self.lib_esw.tabela_para_excel([self.test_table], self.test_excel)
        self.assertTrue(mock_save.called)

    def test_ler_pdf(self):
        resultado = LibEsw.ler_pdf(caminho_pdf)
        self.assertIsNone(resultado)

    @patch('builtins.open', new_callable=mock_open)
    def test_criar_txt(self, mock_file):
        self.lib_esw.criar_txt(self.test_content, self.test_txt)
        mock_file.assert_called_with(os.path.join(self.lib_esw.dir_atual, self.test_txt), 'w', encoding='utf-8')
        mock_file().write.assert_called_once_with(self.test_content)

    def test_ajustar_valor(self):
        self.assertEqual(self.lib_esw.ajustar_valor("1.234,56"), 1234.56)
        self.assertEqual(self.lib_esw.ajustar_valor("1.234,56D"), -1234.56)
        self.assertEqual(self.lib_esw.ajustar_valor("1.234,56C"), 1234.56)
        self.assertIsNone(self.lib_esw.ajustar_valor("invalid"))
        self.assertIsNone(self.lib_esw.ajustar_valor(pd.NA))

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