# lsd.py

#!.venv/bin/python  
import re  
  
import docx  
from docx.text.run import Run  
  
  
def list\_the\_source\_to\_docx(  
 destination: str,  
 source\_files: list[str],  
 source\_style: Run,  
 title\_heading\_level: int  
):  
 destination\_doc = docx.Document()  
  
 for source\_file in source\_files:  
 destination\_doc.add\_heading(source\_file, title\_heading\_level)  
 print("adding", source\_file)  
 with open(source\_file, 'r') as f:  
 paragraph = destination\_doc.add\_paragraph()  
 \_add\_run\_copy(paragraph, source\_style, f.read())  
 destination\_doc.add\_page\_break()  
 print(source\_file, "added")  
  
 destination\_doc.save(destination)  
  
  
def \_add\_run\_copy(paragraph, run, text=None):  
 # source: https://github.com/python-openxml/python-docx/issues/519  
  
 r = paragraph.add\_run(text=run.text if text is None else text, style=run.style)  
 r.bold = run.bold  
 r.italic = run.italic  
 r.underline = run.underline  
 r.font.all\_caps = run.font.all\_caps  
 r.font.bold = run.font.bold  
 r.font.color.rgb = run.font.color.rgb  
 r.font.color.theme\_color = run.font.color.theme\_color  
 #r.font.color.type = run.font.color.type  
 r.font.complex\_script = run.font.complex\_script  
 r.font.cs\_bold = run.font.cs\_bold  
 r.font.cs\_italic = run.font.cs\_italic  
 r.font.double\_strike = run.font.double\_strike  
 r.font.emboss = run.font.emboss  
 r.font.hidden = run.font.hidden  
 r.font.highlight\_color = run.font.highlight\_color  
 r.font.imprint = run.font.imprint  
 r.font.italic = run.font.italic  
 r.font.math = run.font.math  
 r.font.name = run.font.name  
 r.font.no\_proof = run.font.no\_proof  
 r.font.outline = run.font.outline  
 r.font.rtl = run.font.rtl  
 r.font.shadow = run.font.shadow  
 r.font.size = run.font.size  
 r.font.small\_caps = run.font.small\_caps  
 r.font.snap\_to\_grid = run.font.snap\_to\_grid  
 r.font.spec\_vanish = run.font.spec\_vanish  
 r.font.strike = run.font.strike  
 r.font.subscript = run.font.subscript  
 r.font.superscript = run.font.superscript  
 r.font.underline = run.font.underline  
 r.font.web\_hidden = run.font.web\_hidden  
 return r  
  
  
def \_source\_files\_by\_regex(regex: re.Pattern[str]) -> list[str]:  
 current\_working\_directory = os.getcwd()  
  
 return [  
 file[len(current\_working\_directory) + 1:]  
 for root, dirs, files in os.walk(current\_working\_directory)  
 for file in (f'{root}/{file}' for file in files)  
 if regex.match(file) and '/.' not in file  
 ]  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 from argparse import ArgumentParser  
 from sys import argv  
 import os  
 import re  
  
 if argv[0].endswith('.py'):  
 argv = argv[1:]  
  
 dir\_path = os.path.dirname(os.path.realpath(\_\_file\_\_))  
  
 parser = ArgumentParser(  
 prog="List the Source to Docs : lsd",  
 description="This program creates single .docx file from the source code provided"  
 )  
 parser.add\_argument(  
 '-d', '--destination',  
 default='./listing.docx',  
 help='.docx file where result shall be put'  
 )  
 parser.add\_argument(  
 '-f', '--source\_files',  
 default=\_source\_files\_by\_regex(re.compile('(.\*/)\*[a-z\_]+\\.py')),  
 type=lambda arg: \_source\_files\_by\_regex(re.compile(arg)),  
 help='Regexp filter of files that shall be included into listing'  
 )  
 parser.add\_argument(  
 '-t', '--title\_heading\_level',  
 default=1,  
 type=int,  
 help="Heading level used for titling styling"  
 )  
 parser.add\_argument(  
 '-s', '--source\_style',  
 default=docx.Document(dir\_path + "/\_source\_style.docx").paragraphs[0].runs[0],  
 type=lambda path: docx.Document(path).paragraphs[0].runs[0],  
 help=".docx file, which first symbol will be used as source code style reference"  
 )  
  
 args = parser.parse\_args(argv)  
 if argv != ['-h'] and argv != ['--help']:  
 list\_the\_source\_to\_docx(\*\*vars(args))