"""

Compute the ARI for a given body of text loaded in from a file. The automated readability index (ARI) is a formula

for computing the U.S. grade level for a given block of text. In plain English, the score is computed by multiplying the number of characters divided by the number of words by 4.17, adding the number of words divided by the number of sentences multiplied by 0.5, and subtracting 21.43. If the result is a decimal, always round up.

"""

import os

import re

BOOKS = '/Users/michaelevan/temp/pdx\_code/PythonFullStack/1\_Python/3\_Applied\_Python/labs/ari/books/'

BOOKS2 = '/Users/michaelevan/temp/pdx\_code/PythonFullStack/1\_Python/3\_Applied\_Python/labs/ari/books/licenses/'

**def get\_data(path):**

"""

*Retrieves file contents as a string.*

"""

with open(path, 'r') as file: *# Context manager returns a file object*

text = file.read() *# Read the path, passed as an argument in get\_data*

return text *# Return the path object*

**def show\_options():**

"""

*Show booktitles by combining BOOKs file path with enumerated .txt files; and allow user to select .txt file.*

"""

paths = os.listdir(BOOKS) *# os method to show the file path BOOKS, above*

books = {index: book for index, book in enumerate(paths, start=1)} *# book=loop var to create dictionary w/ keys*

update\_data = {len(books) + 1: 'exit'} *# var creates dict at length of len(books) +1*

books.update(update\_data) *# update books using above function*

print(books) *# print books/ or return books*

choice = int(input('To compute its automated readability index, pick from one of the files:>> ')) *# user input*

booktitle = books[choice] *# booktitle returns user's choice of books, generated by enumerated dict*

path\_combo = BOOKS + booktitle *# path\_combo includes BOOKS filepath + booktitle, user's choice above.*

return booktitle, path\_combo *# returns result from path\_combo*

**def display\_output(ari\_score: int, fname: str) -> None:** *# Passes the param of ari\_score as int, and fname as str*

ari\_scale = {

1: {'ages': '5-6', 'grade\_level': 'Kindergarten'},

2: {'ages': '6-7', 'grade\_level': '1st Grade'},

3: {'ages': '7-8', 'grade\_level': '2nd Grade'},

4: {'ages': '8-9', 'grade\_level': '3rd Grade'},

5: {'ages': '9-10', 'grade\_level': '4th Grade'},

6: {'ages': '10-11', 'grade\_level': '5th Grade'},

7: {'ages': '11-12', 'grade\_level': '6th Grade'},

8: {'ages': '12-13', 'grade\_level': '7th Grade'},

9: {'ages': '13-14', 'grade\_level': '8th Grade'},

10: {'ages': '14-15', 'grade\_level': '9th Grade'},

11: {'ages': '15-16', 'grade\_level': '10th Grade'},

12: {'ages': '16-17', 'grade\_level': '11th Grade'},

13: {'ages': '17-18', 'grade\_level': '12th Grade'},

14: {'ages': '18-22', 'grade\_level': 'College'}}

judo = ari\_scale.get(ari\_score, ari\_scale[14]) # from ari\_scale.get(get ari\_score returns the key(int)

ages, level = judo['ages'], judo['grade\_level']

*# triple single quotes allows WYSWIG format for string.*

message = f'''

The ARI for the file '{fname}' is '{ari\_score}'.

This corresponds to a '{level}' level of difficulty

that is suitable for an average person '{ages}' years old.

'''

print(message)

**def processor():**

"""

*Reads and interprets contents of the txt file opened via show\_options.*

*Get qty of sentences, REGEX*

*Get qty of words REGEX*

*Get qty of characters*

*Clean all text; omit periods, commas, and brackets*

"""

bookname, fullpath = show\_options() *# How to call the 'return text' from function above.*

fulltext = get\_data(fullpath)

parsed = re.sub(r'[\:\-\(\)\[\]\–]', '', fulltext) # Regex to parse

pattern = re.compile(r""" *# Multiline uses three double quotations at start and end.*

[A-Z] *# One uppercase letter*

[a-z]+ *# One lowercase letter; meta classes: one class followed by other*

[\s\w\,\:\-\(\)\[\]\–\@]+ *# Followed by a space, a word, and all the other punctuation.*

\. *# Followed by a period.*

""", *# Tells Regex it's a multiline*

re.X) *# Tells Python it's a multiline*

splittah = pattern.findall(fulltext) *# Return regex pattern and finds all instances of it.*

sent\_qty = len(splittah) *# Return the length of items found using regex criteria*

*# print(sent\_qty)*

char\_qty = len(parsed) *# Return length of characters from parsed, met regex criteria*

*# print(char\_qty)*

word\_qty = len(fulltext.split()) *# Return length of words from fulltext after split operation*

*# print(word\_qty)(word\_qty)*

ari\_score = round((4.71 \* (char\_qty / word\_qty)) + (.5 \* (word\_qty / sent\_qty)) - 21.43) *# ARI Formula, w above vars*

display\_output(ari\_score=ari\_score, fname=bookname) *# kwargs explicitly call ari\_score and fname*

*# which are return param values passed from show\_options*

*# to processor(), now called bookname, fullpath*.

­­­

processor()