Richard Hayes Crowley

07/07/2021

CSC\_242\_Lab\_09

**SOURCE CODE:**

**writersGrid.py**

*import* random

*import* os

*from* simple\_term\_menu *import* TerminalMenu

*from* array\_stack *import* ArrayStack

def readFile(*fileString*) -> list:

*# read data from a file, returns list of lines*

*try*:

dataCollection = []

f = open(*fileString*, "r", *encoding*="utf-8")

content = f.read().splitlines()

*# perform file operations*

*for* line *in* content:

dataCollection.append(line)

*except*:

print(f'Could not read {*fileString*}, are you sure it exists?')

*return* False

*finally*:

f.close()

*return* dataCollection

def writeFile(*fileString*, *lineList*=[]) -> None:

*# write data to a file*

*try*:

f = open(*fileString* + ".csv", "w", *encoding*="utf-8")

*# perform file operations*

*# first line*

*for* line *in* *lineList*:

f.write(f"{line}\n")

*except*:

print(f'Could not write {*fileString*}!')

*return* False

*finally*:

f.close()

print(f"{*fileString*} written with {len(*lineList*)} lines!")

def generateScript(*samples*: ArrayStack) -> list:

scriptList = []

scriptList.append(f"I started the day by looking for the {*samples*.pop()}.")

scriptList.append(f"I planned later to walk to the {*samples*.pop()}.")

scriptList.append(f"Surprisingly, I found the {*samples*.pop()} was empty.")

scriptList.append(f"I wondered if a {*samples*.pop()} would appear.")

scriptList.append(

f"My aunt must have left my cellular telephone with the {*samples*.pop()}.")

scriptList.append(

f"Yesterday, I forgot to take the {*samples*.pop()} to the meeting.")

*return* scriptList

def main():

*# return list of words from each line*

print("\n~\*~\*~\*~ Writer's Grid ~\*~\*~\*~\*~\*~")

wordCollection = readFile("words.csv")

gridStrings = []

wordList = []

*for* idx, line *in* enumerate(wordCollection):

*# split line string into array of words*

splitLine = [word *for* word *in* line.split(',')]

*# create strings to print grid*

wordString = f"Grid line {idx + 1}: {splitLine}"

gridStrings.append(wordString)

*# append split words to wordsList to be used as samples in stack*

wordList.append(splitLine)

*# print each line in grid*

*for* line *in* gridStrings:

print(line)

*# flatten word list*

flatWordList = [item *for* sublist *in* wordList *for* item *in* sublist]

*# instantiate stack*

wordStack = ArrayStack(flatWordList)

*while* True:

choice = TerminalMenu(

["Generate Random Script", "View Saved Scripts", "Exit"], *title*="\nWhat would you like to do?").show()

*if* choice == 0:

*# # randomly select a subset of elements ( the stack )*

samples = random.sample([word *for* word *in* wordStack], 6)

randomScript = generateScript(samples)

print("\nSCRIPT GENERATED\n")

*for* line *in* randomScript:

print(line)

save\_script = TerminalMenu(

["Yes", "No"], *title*="\nWould you like to save this script?").show()

*if* save\_script == 0:

fileName = str(input("Please enter a file name: "))

writeFile(fileName, randomScript)

*elif* choice == 1:

fileList = []

*# look for all files with csv exentsion in this directory*

*for* file *in* os.listdir("./"):

*if* file.endswith(".csv") and file != "words.csv" and file != "entropy.csv":

fileList.append(file)

*if* len(fileList) > 0:

chooseFile = TerminalMenu(

fileList, *title*="\nWhich script would you like to read?").show()

openFile = readFile(fileList[chooseFile])

print(f"\nReading {fileList[chooseFile]}\n")

*for* line *in* openFile:

print(line)

*else*:

print("No scripts yet! Please generate a random script and save it.")

*else*:

print("\nGoodbye!\n")

exit()

*if* \_\_name\_\_ == "\_\_main\_\_":

main()

**ArrayStack.py**

*from* arrays *import* Array

*from* abstract\_stack *import* AbstractStack

class ArrayStack(AbstractStack):

DEFAULT\_CAPACITY = 10

def \_\_init\_\_(*self*, *sourceCollection*=None):

*self*.items = Array(ArrayStack.DEFAULT\_CAPACITY)

AbstractStack.\_\_init\_\_(

*self*, *sourceCollection*).\_\_init\_\_(*sourceCollection*)

*# accessors*

def \_\_iter\_\_(*self*):

cursor = 0

*while* cursor < len(*self*):

*yield* *self*.items[cursor]

cursor += 1

def peek(*self*):

*if* *self*.isEmpty():

*raise* KeyError("The stack is Empty")

*return* *self*.items[len(*self*)-1]

*# mutators*

def clear(*self*):

*self*.size = 0

*self*.items = Array(ArrayStack.DEFAULT\_CAPACITY)

def push(*self*, *item*):

*# check array memory / load factor here, increase size if necessary*

*# increasing size of array*

*if* *self*.size == len(*self*.items):

*# create new array and copy data from old array*

*# double the size of the array instead of adding one new cell each time the array needs to be resized to ensure better performance*

temp = Array(len(*self*.items)\*2)

*for* i *in* range(*self*.size):

temp[i] = *self*.items[i]

*# reset old array variable to new array, old arrays memory is left out for the garbage collector*

*self*.items = temp

*self*.items[len(*self*)] = *item*

*self*.size += 1

def pop(*self*):

*if* *self*.isEmpty():

*raise* KeyError("The stack is Empty")

oldItem = *self*.items[len(*self*)-1]

*self*.size -= 1

*# check array memory / load factor here, decrease size if necessary*

*# performant choice is, if logical size is less than or equal to the 1/4 the length of b and the length of b is greater than 2x the default capacity*

*if* *self*.size <= len(*self*.items) // 4 and len(*self*.items) >= ArrayStack.DEFAULT\_CAPACITY \* 2:

temp = Array(len(*self*.items) // 2)

*for* i *in* range(*self*.size):

temp[i] = *self*.items[i]

*self* = temp

*return* oldItem

**OUTPUT:**

**See attached demo video!**