# Creating a dictionary

my\_dict = {'name': 'Alice', 'age': 25, 'city': 'New York'}

# Accessing a value by key

name = my\_dict['name'] # Output: 'Alice'

# Adding a new key-value pair

my\_dict['email'] = 'alice@example.com'

# Updating an existing key-value pair

my\_dict['age'] = 26

# Using del to remove a key-value pair

del my\_dict['city']

# Using pop() to remove a key and return its value

age = my\_dict.pop('age') # Output: 26

# Iterating over keys

for key in my\_dict:

print(key, my\_dict[key])

# Iterating over values

for value in my\_dict.values():

print(value)

# Iterating over key-value pairs

for key, value in my\_dict.items():

print(key, value)

# Checking if a key exists in the dictionary

if 'name' in my\_dict:

print('Name is present')

# Using update() to merge two dictionaries

another\_dict = {'country': 'USA', 'age': 30}

my\_dict.update(another\_dict)

Task:

Let's consider a real-life scenario where you are analyzing customer feedback for a product. You have a large data set of customer reviews in the form of strings, and you want to extract useful information from them using the three identified tasks:

Task 1. String in lowercase: You want to pre-process the customer feedback by converting all the text to lowercase. This step helps standardize the text. Lower casing the text allows you to focus on the content rather than the specific letter casing.

Task 2. Frequency of all words in a given string: After converting the text to lowercase, you want to determine the frequency of each word in the customer feedback. This information will help you identify which words are used more frequently, indicating the key aspects or topics that customers are mentioning in their reviews. By analyzing the word frequencies, you can gain insights into the most common issues raised by customers.

Task 3. Frequency of a specific word: In addition to analyzing the overall word frequencies, you want to specifically track the frequency of a particular word that is relevant to your analysis. For example, you might be interested in monitoring how often the word "reliable" appears in customer reviews to gauge customer sentiment about the product's reliability. By focusing on the frequency of a specific word, you can gain a deeper understanding of customer opinions or preferences related to that particular aspect.

By performing these tasks on the customer feedback dataset, you can gain valuable insights into customer sentiment

class TextAnalyzer(object):

def \_\_init\_\_ (self, text):

# remove punctuation

formattedText = text.replace('.','').replace('!','').replace('?','').replace(',','')

# make text lowercase

formattedText = formattedText.lower()

self.fmtText = formattedText

def freqAll(self):

# split text into words

wordList = self.fmtText.split(' ')

# Create dictionary

freqMap = {}

for word in set(wordList): # use set to remove duplicates in list

freqMap[word] = wordList.count(word)

return freqMap

def freqOf(self,word):

# get frequency map

freqDict = self.freqAll()

if word in freqDict:

return freqDict[word]

else:

return 0

givenstring="Lorem ipsum dolor! diam amet, consetetur Lorem magna. sed diam nonumy eirmod tempor. diam et labore? et diam magna. et diam amet."

analyzed = TextAnalyzer(givenstring)

# Step 1: Create an instance of TextAnalyzer class

print("Formatted Text:", analyzed.fmtText)

Step 2: Call the function that converts the data into lowercase

print("Formatted Text:", analyzed.fmtText)

Step 3: Call the function that counts the frequency of all unique words from the data

freqMap = analyzed.freqAll()

print(freqMap)

Step 4: Call the function that counts the frequency of a specific word

word = "lorem"

frequency = analyzed.freqOf(word)

print("The word",word,"appears",frequency,"times.")

-----------------------------------------------------------------------------------------------------------

Exercise

Your local university's Raptors fan club maintains a register of its active members on a .txt document. Every month they update the file by removing the members who are not active. You have been tasked with automating this with your Python skills.  
Given the file currentMem, Remove each member with a 'no' in their Active column. Keep track of each of the removed members and append them to the exMem file. Make sure that the format of the original files in preserved. (*Hint: Do this by reading/writing whole lines and ensuring the header remains* )  
Run the code block below prior to starting the exercise. The skeleton code has been provided for you. Edit only the cleanFiles function.

#Run this prior to starting the exercise

from random import randint as rnd

memReg = '/members.txt'

exReg = '/inactive.txt'

fee =('yes','no')

def genFiles(current,old):

with open(current,'w+') as writefile:

writefile.write('Membership No Date Joined Active \n')

data = "{:^13} {:<11} {:<6}\n"

for rowno in range(20):

date = str(rnd(2015,2020))+ '-' + str(rnd(1,12))+'-'+str(rnd(1,25))

writefile.write(data.format(rnd(10000,99999),date,fee[rnd(0,1)]))

with open(old,'w+') as writefile:

writefile.write('Membership No Date Joined Active \n')

data = "{:^13} {:<11} {:<6}\n"

for rowno in range(3):

date = str(rnd(2015,2020))+ '-' + str(rnd(1,12))+'-'+str(rnd(1,25))

writefile.write(data.format(rnd(10000,99999),date,fee[1]))

genFiles(memReg,exReg)

The two arguments for this function are the files:

- currentMem: File containing list of current members

- exMem: File containing list of old members

This function should remove all rows from currentMem containing 'no'

in the 'Active' column and appends them to exMem.

'''

#TODO: Read each member in the currentMem (1 member per row) file into a list.

# Hint: Recall that the first line in the file is the header.

#TODO: iterate through the members and create a new list of the innactive members

# Go to the beginning of the currentMem file

# TODO: Iterate through the members list.

# If a member is inactive, add them to exMem, otherwise write them into currentMem

pass # Remove this line when done implementation

# The code below is to help you view the files.

# Do not modify this code for this exercise.

memReg = '/members.txt'

exReg = '/inactive.txt'

cleanFiles(memReg,exReg)

headers = "Membership No Date Joined Active \n"

with open(memReg,'r') as readFile:

print("Active Members: \n\n")

print(readFile.read())

with open(exReg,'r') as readFile:

print("Inactive Members: \n\n")

print(readFile.read())

def testMsg(passed):

if passed:

return 'Test Passed'

else :

return 'Test Failed'

testWrite = "/testWrite.txt"

testAppend = "/testAppend.txt"

passed = True

genFiles(testWrite,testAppend)

with open(testWrite,'r') as file:

ogWrite = file.readlines()

with open(testAppend,'r') as file:

ogAppend = file.readlines()

try:

cleanFiles(testWrite,testAppend)

except:

print('Error')

with open(testWrite,'r') as file:

clWrite = file.readlines()

with open(testAppend,'r') as file:

clAppend = file.readlines()

# checking if total no of rows is same, including headers

if (len(ogWrite) + len(ogAppend) != len(clWrite) + len(clAppend)):

print("The number of rows do not add up. Make sure your final files have the same header and format.")

passed = False

for line in clWrite:

if 'no' in line:

passed = False

print("Inactive members in file")

break

else:

if line not in ogWrite:

print("Data in file does not match original file")

passed = False

print ("{}".format(testMsg(passed)))

<details><summary>Click here for the solution</summary>

```python

def cleanFiles(currentMem,exMem):

with open(currentMem,'r+') as writeFile:

with open(exMem,'a+') as appendFile:

#get the data

writeFile.seek(0)

members = writeFile.readlines()

#remove header

header = members[0]

members.pop(0)

# inactive - is a new list

inactive = [member for member in members if ('no' in member)]

'''

The above is the same as

for member in members:

if 'no' in member:

inactive.append(member)

'''

#go to the beginning of the write file

writeFile.seek(0)

writeFile.write(header)

for member in members:

if (member in inactive):

appendFile.write(member)

else:

writeFile.write(member)

writeFile.truncate()

memReg = 'members.txt'

exReg = 'inactive.txt'

cleanFiles(memReg,exReg)

# code to help you see the files

headers = "Membership No Date Joined Active \n"

with open(memReg,'r') as readFile:

print("Active Members: \n\n")

print(readFile.read())

with open(exReg,'r') as readFile:

print("Inactive Members: \n\n")

print(readFile.read())

``

</details>