exercise.M2.exercise.1 => Python Library

Problem summary

This Python exercise uses the MIT Shakespearean corpus to perform essential Python manipulations with built-in data objects such as list and dictionary. Activities build on each other and establish experience in pandas, numpy, and matplotlib library methods while experiencing outcomes from iterators, such as for and while loops, and conditionals, such as the len() function.

Don't worry if this exercise is disorienting! It's a normal reaction and will dissipate as you adapt to the coding environment and essential programming constructs necessary for course success.

Objectives:

1. Import and manipulate a .xlsx file.
2. Assess your Python programming skills.
3. Prepare questions for class discussion to help source additional tools.

Codebook and data files

file naming conventions:

a. = article (news, journal)	c. = cheatsheet code. = .py or .ipynb	g = graphic
howTo. = explanandum	py.M. exercise or assignment python file	r = reading

File Name	Purpose\Description
https://github.com/cosc-526/cosc.526.home.page/blob/main/code_notebook_cosc_526.ipynb **save your own copy!**	> Course Codebook in Jupyter Notebook > name = code.notebook.cosc.526.ipynb
data.exercise.M.2.exercise.1.shakespeare.xlsx	Data: shakespeare text files

Note.1: The codebook is formatted differently, and below highlights expected outcomes.

Note.2: The instructions below are an overview with additional details in the Notebook.

Note.3: Perform your work in your Notebook and generate outcomes for each code block. Export the Notebook as a .pdf for submission. If you have issues generating a .pdf, ensure to submit a .ipynb file at the very minimum.

Problem 0 - Import, inspect, and view descriptive statistics

Import data

Problem.1 - Description =>

The United S

Task.0 - Expected outcome:

```
 """# -*- coding: utf-8 -*- Created Feb 15 07:58:23 2023
@author:17574 b.hogan@snhu.edu
Objective: import data and apply zipper to transform, iterate,
use conditionals, apply functions, leading to python classes work
Library homebase = Python package index:  https://pypi.org
"""
'''====================
#=========================
#============================================================
#=>STEP 1 get pip library install path from
#============================================================
#=========================
#===================='''


     obj_Name    | charcter code      |   explicit code
      --------- | --------------     | ------------------------------
  i) mytuple =  | ( , )              |=> mytuple = tuple(myobject)
 ii) mylist =   | [  ]               |=> mylist  = list(myobject)
iii) mydict =   | { key: value }     |=> mydict  = dict(myobject)
 iv) myset  =   | set(myobject)      |=> myset   = set((myobject)
  v) dataframe =| pd.DataFrame()     |=> df      = pd.DataFrame(myobject)
 vi) mystring = | " "                |=> mystring= str(myobject)
     integer  = |                    |=> <casting> int(<value>)
"""
#-------------------------------------------------------------------------------
#=> Part 0.1 - Download files
#--------------------------

# a) CREATE a folder on desktop C:\\Users\\17574\\Desktop\\data
# b) create a folder in data called “Shakespear_txt_name”
# b) unpack the zipped github shakespeare corpus text files into this folder.
#    https://github.com/bbe2/noodle3blob/Shakespeare-Corpus/shakespeare_txt_fullname.zip


import pandas as pd               #dataframe library
import numpy as np                #numeric library
import matplotlib.pyplot as plt  #visualization library
import os
os.getcwd()          #where am i? <get working directory>
#os.chdir('c:\\Users\BBE\DATA\') #some op.sys use one slash
os.chdir('c:\\Users\\17574\\Desktop\\data') #microsoft uses 2 \\
```

```
os.getcwd()

df0 = pd.DataFrame() #explicitly set the data object
#df0 = pd.read_csv("shakes_corpus_v1.csv")   #ETL method 1
df0 = pd.read_excel("shakes_corpus_v1.xlsx") #ETL method 2
df0.info()
   #             RangeIndex: 37 entries, 0 to 36
   #          Data columns (total 3 columns):
   #              #   Column  Non-Null Count  Dtype
   #             ---  ------  --------------  -----
   #              0   title   37 non-null       object
   #              1   script  37 non-null       object
   #              2   type  37 non-null   object
   #              3   ID      37 non-null       int64
   #             dtypes: int64(1), object(3) memory usage: 1.3+ KB

print(type(df0))  #use type() to always see what an object is
df0.head()
       #                                 title  ...    type
       #             0  Alls Well That Ends Well  ...  Comedy
       #             1             As You Like It  ...  Comedy

#2.1 use pandas df.to_dict() to move data into dictionary object
mydict = df0.to_dict()
print(mydict.keys())    #['title', 'script', 'type', 'ID'])
type(mydict.keys())  # object itself is keys

#2.2 understand what a dictionary and zip is doing
mylist_keys = list(zip(mydict.keys()))
mylist_keys   # [('title',), ('script',), ('type',), ('ID',)]

#Inspect huge data and then break into smaller chunks
mylist_values = list(zip(mydict.values())) #WOW huge !
#point - zip helpful but continue to learn more functions

mylist_values   #======================> #MEGASAURUS
                    #  35: 'Tragedy',
                    # 36: 'Tragedy'},),
                    #{0: 1,
                    # 1: 2,
                    # 2: 3,
'''====================
#=========================
#============================================================
#=>STEP 2 - seperate Megasaurus into usuable object chunks
#============================================================
#=========================
#====================='''

'''2.1'''
type(mylist_values) #=> [({...})],

'''======> packed as [({...})], =>list, tuple, dictionary'''

type(type(mylist_values[1]) )#hmm doesn't unpack
len(mylist_values)  #=> 4 columns in spreadsheet, ie data objects

'''megasaurus - all plays and words'''
mylist_values
                      # => format is list[(tuple(dict))]
                      # [  ({id:title}),({id:script}),
                      #    ({id:type}), ({id:id}) ]
```

```
                        # zip added an key sequential value
'''==>2.2'''
'''use slicing [0:1], [2] to view next level down'''
type(mylist_values[0])  # tuple
mylist_values[0]        #=> [x] is called slicing

                  Out[23]:
                  ({ 0: 'Alls Well That Ends Well',
                     1: 'As You Like It',

'''now think data like in spreadsheet'''
#  columns
#      0      1     2       3
# |title |script| type |  id  |
#  hamlet,oh joy,tragedy, 29

mylist_values[1] #displays all the script text!

'''==>2.3'''
len(mylist_values[1])  # waits its '1' so need to unpack my data

mylist = []
for i in mydict['title'].values():
    mylist.append(i)
mylist
len(mylist)  #37 - does htat match spreadsheet? always know your bounds

title_total_characters = 0  #how many characters?
for i in mylist:
    title_total_characters = title_total_characters + len(i)
title_total_characters  #do you get 560 ?

'''============================================================
==>2.4 autoBOTs304 - repeat this for total script words
#============================================================
===> moved  this into the graded_assign_wk7'''

#====================
#==========================
#============================================================
#=>STOP! : view 'Variable Explorer' window
#  use this feature to propel data transformation learning
#============================================================
#==========================
#====================

'''#====================
#=> WRAP - UP  Housekeeping
#  delete variables not using; help avoid unnecssary mistakes
#==========
# be mindful how you stage both variable and data names
#   df0 = baseline import
#      df1 = analysis 1
#             df2 = analysis 2
#==================='''

'''==>2.5'''
del mylist_keys      # del removes a variable

'''
mylist2 = []
for i in [mydict.get('title')]:
```

```
    mylist.append(i)   #so what happended here a. wrote name list wrong
print(len(mylist2), len(mylist))
#make a note here on what happended.............
mylist    #stacked a list on a dictionary bc meant to use list2
'''
#go back and rest data for part 2
mylist = []
for i in mydict['title'].values():
    mylist.append(i)

'''====================
#==========================
#============================================================
#=>STEP 3:  Use dir(object) to learn its methods to get work done
#============================================================
#==========================
#======================'''
'''==>3.1'''
#===============> use dir() to get functions available for an object
myset = set()
print(type(myset))
dir(myset)
 # '__xor__',  ==> these are constructors, more later
 #     'add', 'clear',      ==> these are methods
 #   'copy','difference', 'difference_update', 'discard',
 #  'intersection','intersection_update', 'isdisjoint', 'issubset',
 #   'issuperset','pop', 'remove','symmetric_difference',
 #   'symmetric_difference_update','union',  'update']'''

'''==>3.1'''# ====> SETS
mylist2 = mylist
mylist2.append("Winters Tale")  #add one duplicate title
myset = set(mylist2)
print(len(mylist),len(myset))  #so got rid of duplicate
del mylist2

#============> ACTION learn what you need and go find it
mystring = ""
print(type(mystring))
dir(mystring)
#'''__subclasshook__',  'capitalize', 'casefold',, 'center',
#'count', 'encode', 'endswith', 'expandtabs', 'find', 'format',
#'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal',
#'isdigit', isidentifier', 'islower', 'isnumeric', 'isprintable',
#'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip',
#'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust',
#'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines','startswith',
#'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']'''
'''====================
#==========================
#============================================================
#=>STEP 4: More dictionary: .keys(), .values(), .get(<key>)
#============================================================
#==========================
#======================'''

'''==>4.1'''
mydict.get('title')  #.get() views one series
play_names = [mydict.get('title')]
play_names
       [{0: 'Alls Well That Ends Well',
       1: 'As You Like It',
```

```
-----------------------------------------
       2: 'The Comedy of Errors',
mylist
# Now add titles to a different object with an iterator
mylist2 = []
for i in [mydict.get('title')]: #method returns a dict obj
    mylist2.append(i)
mylist2
       [{0: 'Alls Well That Ends Well',
       1: 'As You Like It',
       2: 'The Comedy of Errors',

#3.2 =>  Learn dictionary key, value, items parameters
mylist_key =  []
mylist_values = []
for k,v in mydict.items():
    mylist_key.append(k)
    mylist_values.append(v)
mylist_key              #['title', 'script', 'type', 'ID']
mylist_values   #'''again megasaurus'''

'''==>4.2''' #=>  Understand and count items in a list
len(mylist_values) #hmm  why is this only four ?
mylist_values[0]
mylist_values[1]
mylist_values[2]
mylist_values[3]

#=====================
#==========================
#============================================================
#=>STEP 3:  Use Functions and get Meta Data
#============================================================
#==========================
#=====================
https://docs.python.org/3/library/functions.html#built-in-functions

sum(mylist_values[3])-1
sum(df0['ID'])-1
len(set(df0['ID']))
```

==========

Problem.2 - Description =>

Which v

Task.0 - Expected outcome:

Problem.3 - Description =>

Which cou

Task.0 - Expected outcome:

Problem.4 - Description =>

Task.0 - Expected outcome:

Problem.5 - Description =>

Determine e

Task.0 - Expected outcome:

Problem.6 - Description =>

Determine

Task.0 - Expected outcome:

Additional resources

- https://github.com/cosc-526/cosc.526.home.page
- Jupyter Community Forum

Additional resources

- need help? Jupyter Community Forum

10. Additional resources

- Anaconda for windows
- Install scientific packages.
- Anaconda installation documentation.
- Jupyter Notebook documentation (including get started guides).
- Jupyter Discourse Forum.
    - Search here for tips, tricks, and solutions.
- Python Package Index (pypi)