* Reading & Writing Files with Open
  + Can use python open file function to open .txt files
  + Ex:
    - **File1=open(“/resources/data/Example2.txt”,”w”)**
      * First is the “open” function
      * File1 is the file object.
      * In parenthesis is the file path and file name
      * W is a mode (R for Reading, W for Writing and A for Appending)
    - Can use .name to find the name of the file. This will return a str with the file path and name.
    - Can also use .mode to find what mode is the data attribute in.
    - Table

      Description automatically generatedText

      Description automatically generatedAlways use .close() to close the file.
    - Table

      Description automatically generated with medium confidenceTo Print out each line in the file.
    - A picture containing table

      Description automatically generatedMethods to use
      * **Open(filename, “mode”)**
      * **File1.name**
      * **File1.mode**
      * **File1.read**
      * To read 1 line
        + **With open(example1, “r”) as file1:**

**print(“first line: “ + file1.readline())**

#You can add parameter to readline() but it will only read a single line no matter how large the parameter.

* + - Best way to open and close files
      * **With open(filename, “mode”) as file 1;**
        + **FileContent = file1.read()** #Note you can add parameters into read to only read a certain number of items. Like read(4)
        + **Print(FileContent)**
      * **File1.closed** #use this to verify that the file is closed returns true of false.
    - Read all lines and save as a list
      * **With open(example1, “r”) as file1:**
        + **FileasList = file1.readlines()** #returns all lines as a list.
        + Now you can retrieve information like FileasList[0]
    - Iterating all the line
      * **with open(example1,"r") as file1:**
      * **i = 0;**
      * **for line in file1:**
      * **print("Iteration", str(i), ": ", line)**
      * **i = i + 1**
  + Writing Files with Open
    - Fileobject.write(“This is line1”)
    - To create file
      * **File1 = open (“/resources/data/Example2.txt”, “w”)**
    - To write to a file
      * **File1.write(“This is line A\n”)** # note the \n for a new line.
      * **Writing with loops**
        + **With open (“/resources/data/Example2.txt”, “w”) as file1:**

**For line in lines:**

**File1.write(line)**

* + - * Important that if you write information to an existing file with mode W it overwrites all the existing data.
    - To copy 1 file to another
      * **With open(“example1.txt”,”r”) as readfile:**
        + **With open(“Example3.txt”, “w”) as writefile:**

**For line in readfile:**

**Writefile.write(line)**

* + - **r+** : Reading and writing. Cannot truncate the file.
    - **w+** : Writing and reading. Truncates the file.
    - **a+** : Appending and Reading. Creates a new file, if none exists.

Notes:

To work with a file on existing data, use **r+** and **a+**. While using **r+**, it can be useful to add a .truncate() method at the end of your data. This will reduce the file to your data and delete everything that follows.

**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 **=** [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())

* Pandas
  + How to load data
    - Import pandas as pd #pd is used as a shorthand
    - csv\_path=’file1.csv’
    - df = pd.read\_csv(csv\_path)
  + Dataframes
    - csv\_path=’file1.csv’
    - df = pd.read\_csv(csv\_path)
    - Table

      Description automatically generated with medium confidenceTable

      Description automatically generateddf.head() #head prints out the first 5 rows of the data frame.
    - To find unique elements in a df
      * From df[‘Released’].unique() #released in the release year of the albums.
      * The .unique creates
      * Format
        + Dfname[‘header’].unique() # this generates a list of all unique elements in the list.
    - Logic Operators
      * Df[‘Released’]>=1980 #this creates a column indicating if a song is realised later than or equal to 1980. It returns either true or false.
    - Storing results in a new dataframe
      * Df1=df[df’Realsed]>=1980]
    - Saving it as CSV
      * Df1.to\_csv(‘new\_songs.csv’)
* Numpy in Python
  + First import numpy
    - **Import numpy as np** and import **matplotlib.pyplot as plt**
  + One dimension
    - How to create an array
      * Ndarray – **a=np.array([0,1,2,3,4])**
    - To access use a[0] etc
    - A.size to get size
    - A.ndim to get the number of array dimension
    - .dtype returns the data type of the array.
    - Basic operations
      * A picture containing graphical user interface

        Description automatically generatedVector addition and subtraction
        + Traditional

**U[1,0] and v=[0,1]**

Create a empty array **z=[]**

For n,m in zip(u,v):

**Z.append(n+m)**

* + - * + Using Numpy (This will run way faster since there is no loop)

**U=np.array([1,0])**

**V=np.array([0,1])**

**Z=u+v** #subtraction also works.

**Return z:array([1,1])**

* + - * Array multiplication with a scalar
        + Traditional

**Y=[1,2]**

**Z=[]**

**For n in y:**

**Z.append(2\*n**)

* + - * + Using Numpy

**Y=np.array([1,2])**

**Z=2\*y**

**Returns z:array([2,4])**

* + - * Product of two numpy arrays
        + **u=np.array([1,2])**
        + **v=np.array([3,2])**
        + **z=u\*v**

**z:array([3,4])**

* + - * Dot Product #done by multiplying each top and bottom and adding them together.
        + **u=np.array([1,2])**
        + **v=np.array([3,2])**
        + **result = np.dot(u,v)**

**returns – 5**

* + - * **Text

        Description automatically generated**Adding Constant to an NumPy array (Broadcasting)
      * Universal Functions
        + Finding mean

**A=np.array([1,-1,1,-1])**

**Mean\_a=a.mean()**

* + - * + Finding Max / min just replace max with min

**B=np.array([1,-2,3,4,5])**

**Max\_b=b.max()**

* + - * + Value of pi

**Np.pi**

* + - * + Sin’

**X=np.array([0, np.pi/2, np.pi])**

**Y=np.sin(x)**

* + - * + Standard Deviation

**Standard\_deviation=a.std()**

Returns the standard deviation of a

* + - * + Shape

A.shape

Reutnrs the shape/size of the numpy array

Table

Description automatically generated with medium confidence

* + - * + Linespace

Code:

**Np.linspace(-2,2,num=5)**

-2 = starting point

2 = ending point

5 is how many evenly spaced number in between -2 and 2

* + - * + Chart

          Description automatically generatedPlotting Mathematical Functions
        + Slicing Arrays

D = c[1:4]

This will fetch values form index 1 to 4 and set it to D

* + Two dimensions
    - Assuming this is the list you have
      * a = [ [11,12,13], [21,22,23], [31,32,33] ]
      * A = np.array(a)
      * Text

        Description automatically generated with low confidenceResult
      * A.ndim : 2
      * A.shape: (3,3)
      * A.size : 9
      * Slicing
    - Table

      Description automatically generatedAll operations are similar to it was in single dimension
    - Sample array
      * **Y = np.array ( [ [2,1], [1,2] ] )**
    - Text

      Description automatically generatedDot product of 2 different shape of array
    - Data could be accessed by using a[1,2] or a[1][2]
    - Transposing (flipping)
  + Use .T

0 1 0 1 0

1 0 1 0 1