Week 4

Working with Data in Python

Reading Files with Open

directory name

```
File1 = open("/resources/data/Example2.txt", "w")
```

Mode

R – reading
W – writing
A – appending

```
File1.name : 'resources/data/Example2.txt'
File1.mode : 'r'
File1.close
```

automatically close the file here

```
with open("Example.txt", "r") as File1:
    file_stuff = File1.read
    print(file_stuff)

print(File1.closed)
print(file_stuff)
```

```
with open("Example1.txt","r") as file1
    file_stuff = file1.readline()
    print(file_stuff)
```

readline() → read just a line

```
file_stuff = file1.readline(4)
```

readline(4) → first 4 characters of the line

from where you stop last

```
with open("Example1.txt","r") as file1
    i=0
    for ln in file1:
        print(str(i), ln)
        i=i+1
```

Writing files with open

File1 = open("/resource/data/Exafile2.txt", "w")

write

with open ("   ", "w") as file1

this creates file Exafile2.txt in the directory

file1.write("This is line A \n")

new line

file1.write("This is line B")

Lines = ["This line A \n", "This line B \n",

"This line C \n", "This line C \n"]

with open("Exafile2.txt", "w") as file1

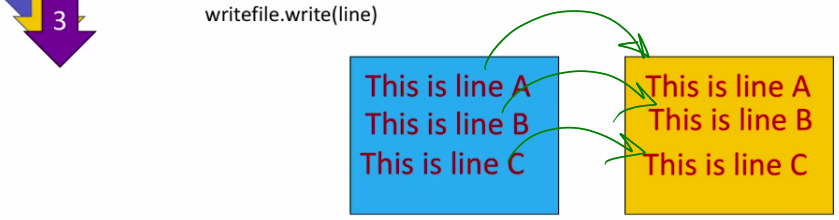
for l in Lines

file1.write(l)

if we use append "a" inted of "w" it will not create text file, but It will override excisting file

Copy file

```
with open("Example1.txt", "r") as readfile:
    with open("Example3.txt", "w") as writefile:
        for line in readfile:
            writefile.write(line)
```



IBM Developer SKILLS NETWORK

Now lets revisit **a+** — Appending and Reading

```
with open('Example2.txt', 'a+') as testwritefile:
    print("Initial Location: {}".format(testwritefile.tell()))

    # my modification
    testwritefile.write("This is by Dulitha\n")

    data = testwritefile.read()
    if (not data):  #empty strings return false in python
            print('Read nothing')
    else:
            print(testwritefile.read())

    testwritefile.seek(0,0) # move 0 bytes from beginning.

    print("\nNew Location : {}".format(testwritefile.tell()))
    data = testwritefile.read()
    if (not data):
            print('Read nothing')
    else:
            print(data)

    print("Location after read: {}".format(testwritefile.tell()))
```

move using seek(0,0)

beginning

```
Initial Location: 153
Read nothing

New Location : 0
Overwrite
This is line C
This is line D
This is line E
This is line E
This is line E
This is line E
This is line E
This is by Dulitha
This is by Dulitha
This is by Dulitha
```

Loading Data with Pandas

```
import pandas as pd

csv_path = 'files1.csv'

df = pd.read_csv(csv_path)
```

DataFrame

```
df = pd.read_csv(csv_path)

df.head()
```

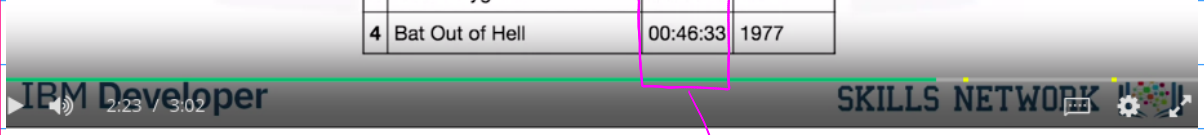
→ shows first five rows of data

```
xlsx_path = 'files1.xlsx'
df = pd.read_excel(xlsx_path)

Songs = {'Album': [ ], 'Released': [ ],
         'Length': [ ]}

Songs_frame = pd.DataFrame(Songs)
```

Dataframes

```
songs = {'Album': ['Thriller','Back in Black', 'The Dark Side of the Moon',\
'The Bodyguard','Bat Out of Hell'],
'Released': [1982,1980,1973,1992,1977],
'Length':['00:42:19','00:42:11','00:42:49','00:57:44','00:46:33']}
```

	Album	Length	Released
0	Thriller	00:42:19	1982
1	Back in Black	00:42:11	1980
2	The Dark Side of the Moon	00:42:49	1973
3	The Bodyguard	00:57:44	1992
4	Bat Out of Hell	00:46:33	1977



new data frame

```
n = df['Length']
```

```
y = df[['Artist', 'Length', 'Genre']]
```

Second video

⊛ List unique values

```
df['Released'].unique():
```

All of the unique elements in 'Released'

```
df['Released'] >= 1980 :
```

Set of boolian satisfy inequality.

```
df1 = df[df['Released'] >= 1980]
```

↑ each album came after 1980

```
df1.to_csv('new_songs.csv')
```

↑ Save as CSV

IBM Cloud Acount

dalikm.mj@gmail.com
Dalikm98

Numpy in Python

One D Numpy

np array Similer to List
each elemt is same type

```
import numpy as np

a = np.array([0,1,2,3])
a: array([0,1,2,3])
type(a): numpy.ndarray
a.dtype : dtype('int64')

a.size :5
a.ndim : 1
a.shape : (5,)
```

Indexing & Slicing

```
c = np.array([20,1,2,3,4])

c[0] = 100
c: array([100,1,2,3,4])

d = c[1:4]
d: array([1,2,3])
```

```
c[3:5] = 300, 400
c: array ([100, 1, 2, 300, 400])
```

Vetors

$$u = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad v = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$z = u + v = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

```
u = np.array([1,0])
v = np.array([0,1])

z = u+v
z: array([1,1])

z = u-v
z: array([1,-1])
```

multiplication

```
y = np.array([1,2])
z = 2*y
z: array([2,4])
```

$$u = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad v = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

Hadamula product

$$Z = u \circ v = \begin{bmatrix} 1 \times 2 \\ 2 \times 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \end{bmatrix}$$

```
u = np.array([1,2])
v = np.array([2,3])
```

```
Z = u*v
Z: array([2,6])
```

Dot product

$$Z = u^T v = \begin{bmatrix} 1, 2 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

$$= 8$$

```
Z = np.dot(u,v)
Z: 8
```

Adding constant. (broadcasting)

```
u = np.array([1,2,3,4])

Z = u+1
Z : array([2,3,4,5])
```

Universal fn

functions operate on ndarray

```
a = np.array([1,-1,1,-1])

mean_a = a.mean()      : 0

max_a = a.max()        : 1
```

```
np.pi   : π

x = np.array([0, np.pi/2, np.pi])   : [0, π/2, π]

y = np.sin(x)   : [sin(0), sin(π/2), sinπ]
```

```
x = np.linspace(-2, 2, num=5)
x:  [-2, -1, 0, 1, 2]
```

Plotting Functions

```
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)

import matplotlib.pyplot as plt
% matplotlib inline   <- Jupiter notebook display the plot
plt.plot(x, y)
```

Two dim Numpy

List → a = [[11, 12, 13], [21, 22, 23], [31, 23, 33]]

A = np.array(a)

A: $\begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{bmatrix}$

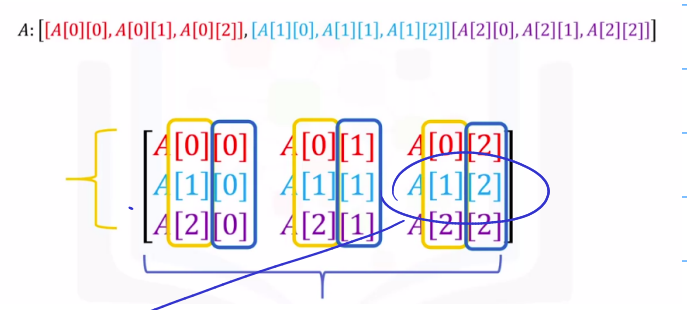
A.ndim : 2 ← Matrix

A.shape : (3,3) ← 3 × 3 matrix

(1 2 3
4 5 6) → (2,3)

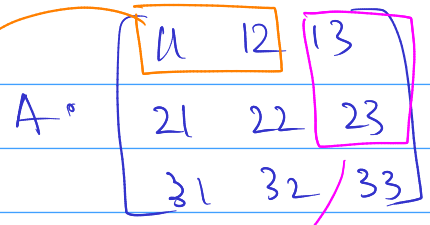
A.size : 9 ← Total number of elements

Index



A[1][2] : 23

Slicing



A[0,0:2]

row

column

A[0:2,2] : [13,23]

Matrix multiplication ← dot product

C = np.dot(A,B)

A and B are same dim

[26]:
```
import numpy as np

a = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
A = np.array(a)
print(A)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

[27]:
```
Z1 = A[0:3,0:2]
print(Z1)
```

```
[[ 1  2]
 [ 5  6]
 [ 9 10]]
```

[28]:
```
Z2 = A[0:3][0:2]
print(Z2)
```

```
[[1 2 3 4]
 [5 6 7 8]]
```

why different?