= Python Mod 4 =

READING FILES WITH OPEN

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

open function — File directory & file name — Mode: w = writing, r = reading, a = appending

Data attributes of a file

File1.name
'resources/data/Example2.txt'

File1.mode
'r'

File1.close() - to close the file using the method close

With statement to open and close a file

```
with open("Example1.txt","r") as File1:
    file_stuff = File1.read()
    print(File_stuff)
print(File.closed)
print(File_stuff)
```

read() → all

We can use the method .readline to read just the 1st line of the file BUT if we add the comand more than time we get line by line.

We can also out put every line as an element in a list using the method .readlines()

We can use a loop to print out each line individually as follows

```
with open("Example1.txt","r") as File1:
    for line in File1:
        print(line)
```

Use to print just the 1st 4 characters of the 1st line:

```
with open("Example1.txt","r") as File1:
    file_stuff = File1.readlines(4)
    print(file_stuff)
    file_stuff = File1.readlines(5)
```

Hello world line1
Hell
Hi there line2
Hi th

WRITING FILES WITH OPEN

File object.write("This is line 1")
→ text will be written to the file

File 1 = open("/resources/data/Example2.txt", "w")
File 1: file object; open: directory; Example2.txt: file name; "w": mode parameter = writing; /resources/data/Example2.txt: file path

→ If I have the file in my directory it will be overwriten

```
with open("/resources/data/Example2.txt","w") as File1:
    File1.write("This is line A")
```
(File 1: file object)

→ Creates a new file example2.txt in my directory, then it writes "This is a new line A" and close the file → add \n to breakdown lines

→ Write each element of a list to a file (creates a new file)

```
Lines = ["This is line A\n", "This is line B\n", "This is line C\n"]
with open("/resources/data/Example2.txt","w") as File1:
    for line in Lines:
        file.write(line)
```

iteration 1 - This is line A
iteration 2 - This is line B
iteration 3 - This is line C
Example2.txt

at the end of the loop the file will be closed

→ Append new lines to an existing file:

```
with open("/resources/data/Example2.txt","a") as File1:
    File1.write("This is line D")
```

... This is line D

→ Copy one file to a new file

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

Example1.txt: This is line A / This is line B / This is line C
Example3.txt: This is line A / This is line B / This is line C

LOADING DATA : PANDAS

Dependancies or libraries are pre-written code to help solve problems.
Pandas popular library for data analysis
pre-built classes and functions

IMPORTING PANDAS

```
import pandas as pd        # this is the abbreviation, it can be anything. Pd.es pandas
csv_path = 'file1.csv'
df = pd.read_csv(csv_path)
```
df: dataframe

...

Dataframe - df

csv:
```
csv_path = 'file1.csv'
df = pd.read_csv(csv_path)
df.head()
```

Exel:
```
xlsx_path = 'file1.xlsx'
df = pd.read_excel(xlsx_path)
df.head()
```

One way pandas allows to work with data is with a data frame.
We can use the method .head() to examine the first 5 rows of a dataframe

- We can create a dataframe out of a dictionary

```
Songs = {'Album': ['Thriller', 'Back in Black', 'Dark Moon', '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']}
```

Keys = Colums titles
Values = Rows

	0	1	2
	Album	Released	Length
0	Thriller	1982	00:42:19
1	Back in Black	1980	00:42:11

- We can create a new dataframe consisting of one column

```
X = df [['Length']]
```

X

	Length
0	00:42:19
1	00:42:11

- Do the same for multiple colums

```
y = df [['Album', 'Length']]
```

- One way to access unique elements is the ix method

Opt 1 (.iloc) — Row, Column

```
df.ix [0,0] : 'Thriller'        df.ix [0,2] : 'Teeny' 00:42:19'
df.ix [1,0] : 'Back in Black'   df.ix [1,2] : '00:42:11'
```

Opt 2 (.loc)

```
df.ix [0, 'Album'] : 'Thriller'        df.ix [0, 'Lenght'] : '00:42:19'
```

- Slice dataframes

Opt 1 (.iloc) — range

```
z = df.ix [0:2, 0:3]
```

Opt 2 (.loc)

```
z = df.ix [0:0, 'Album':'Lenght']
```

z

	Album	Released	Lenght
0	Thriller	1982	00:42:19

WORKING WITH AND SAVING DATA WITH PANDAS

List unique values

- Use method unique

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

	Released
0	1982
1	1900
2	1482

⇨ 1982, 1900

- Create a new dataframe with a specific criteria i.e songs from 80's and after

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

	Released	Song	Lenght
0	1970	A1	2min
1	1983	B2	4min
2	1960	C3	2min
3	1985	D4	1min

⇨

Released	Song	Lenght
1983	B2	4min
1985	D4	1min

df1

- Save as CSV

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