Lesson 8: Python File Handling and Modules

8.1. Python File Handling

File handling is an important part of any web application.

Python has several functions for creating, reading, updating, and deleting files.

The key function for working with files in Python is the open() function. The open() function takes two parameters; *filename*, and *mode*.

There are four different methods (modes) for opening a file:

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists

In addition you can specify if the file should be handled as binary or text mode

"t" - Text - Default value. Text mode

"b" - Binary - Binary mode (e.g. images)

Syntax

To open a file for reading it is enough to specify the name of the file:

```
f = open("demofile.txt")
```

The code above is the same as:

```
f = open("demofile.txt", "rt")
```

Because "r" for read, and "t" for text are the default values, you do not need to specify them.

8.1.1. Read Files

Assume we have the following file, located in the same folder as Python:

```
demofile.txt
    Hello! Welcome to demofile.txt
    This file is for testing purposes.
    Good Luck!
```

To open the file, use the built-in open() function.

The open() function returns a file object, which has a read() method for reading the content of the file:

Example

```
f = open("demofile.txt", "r")
print(f.read())
```

If the file is located in a different location, you will have to specify the file path, like this:

Example

Open a file on a different location:

```
f = open("D:\\myfiles\welcome.txt", "r")
print(f.read())
```

Read Only Parts of the File

By default the read() method returns the whole text, but you can also specify how many characters you want to return:

Example

Return the 5 first characters of the file:

```
f = open("demofile.txt", "r")
print(f.read(5))
```

Read Lines

You can return one line by using the readline() method:

Example

Read one line of the file:

```
f = open("demofile.txt", "r")
print(f.readline())
```

By calling readline() two times, you can read the two first lines:

Example

Read two lines of the file:

```
f = open("demofile.txt", "r")
print(f.readline())
print(f.readline())
```

By looping through the lines of the file, you can read the whole file, line by line:

Example

Loop through the file line by line:

```
f = open("demofile.txt", "r")
for x in f:
  print(x)
```

Close Files

It is a good practice to always close the file when you are done with it.

Example

Close the file when you are finish with it:

```
f = open("demofile.txt", "r")
print(f.readline())
f.close()
```

8.1.2. Write/Create Files

To write to an existing file, you must add a parameter to the open() function:

"a" - Append - will append to the end of the file

"w" - Write - will overwrite any existing content

Example

Open the file "demofile2.txt" and append content to the file:

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()
#open and read the file after the appending:
f = open("demofile2.txt", "r")
print(f.read())
```

Example

Open the file "demofile3.txt" and overwrite the content:

```
f = open("demofile3.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()
```

```
#open and read the file after the appending:
f = open("demofile3.txt", "r")
print(f.read())
```

Note: the "w" method will overwrite the entire file.

Create a New File

To create a new file in Python, use the open() method, with one of the following parameters:

"x" - Create - will create a file, returns an error if the file exist

"a" - Append - will create a file if the specified file does not exist

"w" - Write - will create a file if the specified file does not exist

Example

Create a file called "myfile.txt":

```
f = open("myfile.txt", "x")
```

Result: a new empty file is created!

Example

Create a new file if it does not exist:

```
f = open("myfile.txt", "w")
```

8.1.3. Delete Files

To delete a file, you must import the OS module, and run its os.remove() function:

Example

Remove the file "demofile.txt":

```
import os
os.remove("demofile.txt")
```

Check if File exist:

To avoid getting an error, you might want to check if the file exists before you try to delete it:

Example

Check if file exists, *then* delete it:

```
import os
if os.path.exists("demofile.txt"):
  os.remove("demofile.txt")
else:
  print("The file does not exist")
```

Delete Folder

To delete an entire folder, use the os.rmdir() method:

Example

Remove the folder "myfolder":

```
import os
os.rmdir("myfolder")
```

8.2. Python Modules

8.2.1. Numpy

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python.
In Python we have lists that serve the purpose of arrays, but they are slow to process.
NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.
Arrays are very frequently used in data science, where speed and resources are very important.
Install it using this command:

```
C:\Users\Your Name>pip install numpy
```

If this command fails, then use a python distribution that already has NumPy installed like, Anaconda, Spyder etc.
Once NumPy is installed, import it in your applications by adding the import keyword:

```
import numpy
```

Now NumPy is imported and ready to use.

```
import numpy
arr = numpy.array([1, 2, 3, 4, 5])
print(arr)
```

NumPy is usually imported under the np alias.
Create an alias with the as keyword while importing:

```
import numpy as np
```

Now the NumPy package can be referred to as np instead of numpy.

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr)
```

8.2.2. Pandas

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.
Pandas allows us to analyze big data and make conclusions based on statistical theories.
Pandas can clean messy data sets, and make them readable and relevant.
Relevant data is very important in data science.
Install and import it using these commands:

```
C:\Users\Your Name>pip install pandas
```

```
import pandas as pd
```

Example

```
import pandas as pd
mydataset = {
  'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}
myvar = pd.DataFrame(mydataset)
print(myvar)
```

8.2.2.1. Pandas Series

A Pandas Series is like a column in a table. It is a one-dimensional array holding data of any type.

If nothing else is specified, the values are labeled with their index number. First value has index 0, second value has index 1 etc. This label can be used to access a specified value. With the index argument, you can name your own labels.

Example: Create a series with your own labels:

```
import pandas as pd
a = [1, 7, 2]
myvar = pd.Series(a, index = ["x", "y", "z"])
print(myvar)
```

When you have created labels, you can access an item by referring to the label.

```
print(myvar["y"])
```

8.2.2.2. Pandas DataFrame

A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.

With the index argument, you can name your own indexes.

Pandas use the loc attribute to return one or more specified row(s).

Example: Create a simple Pandas DataFrame:

```
import pandas as pd
data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}
#load data into a DataFrame object:
df = pd.DataFrame(data)
print(df)
#refer to the row index:
print(df.loc[0])
#use a list of indexes:
print(df.loc[[0, 1]])
```

If your data sets are stored in a file, Pandas can load them into a DataFrame.
Example: Load a comma separated file (CSV file) into a DataFrame:

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df)
```

8.2.2.3. Pandas analyzing data

The head() method returns the headers and a specified number of rows, starting from the top to get a quick overview of the dataframe.
The tail() method returns the headers and a specified number of rows, starting from the bottom.
The DataFrames object has a method called info(), that gives you more information about the data set.
The info() method also tells us how many Non-Null values there are present in each column.
Empty values, or Null values, can be bad when analyzing data, and you should consider removing rows with empty values. This is a step towards what is called *cleaning data.*

```
print(df.head(10))
print(df.tail())
print(df.info())
```

8.2.3. SciPy

SciPy is a scientific computation library that uses NumPy underneath. SciPy stands for Scientific Python. It provides more utility functions for optimization, stats and signal processing. Like NumPy, SciPy is open source so we can use it freely. SciPy was created by NumPy's creator Travis Olliphant. SciPy has optimized and added functions that are frequently used in NumPy and Data Science.

8.2.4. Scikit-learn

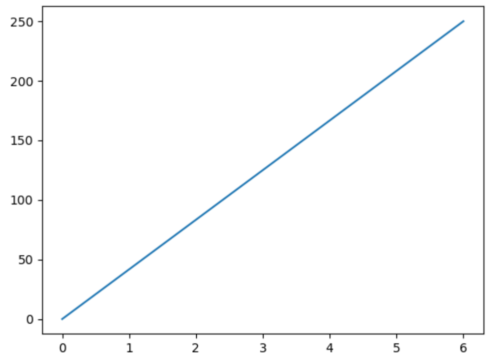
Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, Regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

8.3. Python Matplotlib

Matplotlib is a low level graph plotting library in python that serves as a visualization utility. Matplotlib was created by John D. Hunter. Matplotlib is open source and we can use it freely. Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.
Most of the Matplotlib utilities lies under the `pyplot` submodule, and are usually imported under the `plt` alias.

Example: Draw a line in a diagram from position (0,0) to position (6,250):

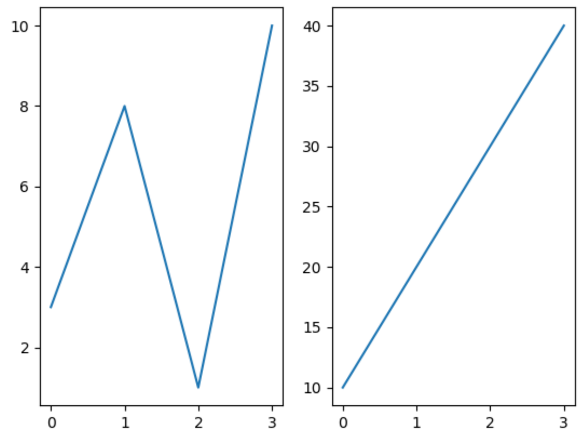
```
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([0, 6])
ypoints = np.array([0, 250])
 plt.plot(xpoints, ypoints)
plt.show()
```



The plot() function is used to draw points (markers) in a diagram. By default, the plot() function draws a line from point to point. The function takes parameters for specifying points in the diagram. Parameter 1 is an array containing the points on the x-axis while Parameter 2 is an array containing the points on the y-axis. If we need to plot a line from (1, 3) to (8, 10), we have to pass two arrays [1, 8] and [3, 10] to the plot function.

With the subplot() function you can draw multiple plots in one figure:
Example: Draw 2 plots:

```
import matplotlib.pyplot as plt
import numpy as np
#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(1, 2, 1)
plt.plot(x,y)
#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(1, 2, 2)
plt.plot(x,y)
plt.show()
```



With Pyplot, you can use the scatter() function to draw a scatter plot.
The scatter() function plots one dot for each observation. It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis.

With Pyplot, you can use the bar() function to draw bar graphs.

In Matplotlib, we use the hist() function to create histograms.
The hist() function will use an array of numbers to create a histogram, the array is sent into the function as an argument.
A histogram is a graph showing *frequency* distributions.
It is a graph showing the number of observations within each given interval.

Lesson 8: Review Question

1. Implement the above methods and modules using an arbitrary dataset.