PANDAS & PYTHON PROGRAMMING LANGUAGE FOR DATA ANALYSIS

In [ ]:
```
import os
import zipfile


zip_dataset = "part-of-speech-tagging.zip"
dataset_filename = "words_pos.csv"
dataset_path = "/Users/Chris-Dunamis/Documents/Portfolios/Python/Projects/FreeCodeCamp - Pandas & Python for Data \"\
    "Analysis/Datasets/English Words/"

for directory_path, directory_names, filenames in os.walk(dataset_path):
    # os.walk: This is a "directory tree generator", which traverses through the:
    #       Relative Path   - (The directory_path specified)
    #       Directory Names - (A list of Directory Name(s) from the Relative Path)
    #       File Names      - (A list of File Name(s) from the Relative Path)
    if dataset_filename in filenames:
        print(f"The 'Dataset: {dataset_filename}' exist in:" + "\n\n" +

              f"PATH: {dataset_path}")

        for i, filename in enumerate(filenames):
            print(f"   |File {i + 1}|: {filename}")

        break
else:
    print(f"The 'Dataset: {dataset_filename}' does not exist in:" + "\n" +
          f"PATH: {dataset_path}" + "\n\n" +

          "Commencing Dataset Download..." + "\n" +
          ":^> + 138)")

    # (Stage 1): Since we are located in the "Current Working Directory: dataset_path", the dataset is set to be
    #            downloaded here:
    %cd ./Datasets/English Words/

    # (Stage 2): In order for this line of code to be executed, the Module: Kaggle needs to be imported to your current
    #            Jupyter Notebook Server and needs to be installed onto the system:
    !kaggle datasets download --force -d rtatman/part-of-speech-tagging

    # (Stage 3): When the dataset is successfully downloaded, the code below is executed to extract the files from the
    #            zipped file:
    with zipfile.ZipFile(zip_dataset, 'r') as zip_ref:  # The Current Working Directory is where the un-zipped
        zip_ref.extractall(dataset_path)                # filename(s)/dataset(s) is located...

    # (Stage 4): For sanity check, this brings us back to the working directory of the Notebook:
    print(f":^> + 138) + "\n" +
          "Now working at home directory..." + "\n" +
          "PATH:", end= ' ')
    %cd ~/.
```

```
The 'Dataset: words_pos.csv' does not exist in:
PATH: /Users/Chris-Dunamis/Documents/Portfolios/Python/Projects/FreeCodeCamp - Pandas & Python for Data Analysis/Datasets/English Words/

Commencing Dataset Download...
--------------------------------------------------------------------------------------------------------------------------------------------------------
/Users/Chris-Dunamis/Documents/Portfolios/Python/Projects/FreeCodeCamp - Pandas & Python for Data Analysis/Datasets/English Words
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /Users/Chris-Dunamis/.kaggle/kaggle.jso
n'
Downloading part-of-speech-tagging.zip to /Users/Chris-Dunamis/Documents/Portfolios/Python/Projects/FreeCodeCamp - Pandas & Python for Data Analysis
/Datasets/English Words
100%|████████████████████████████████████████| 1.99M/1.99M [00:00<00:00, 3.53MB/s]
100%|████████████████████████████████████████| 1.99M/1.99M [00:00<00:00, 3.29MB/s]
--------------------------------------------------------------------------------------------------------------------------------------------------------
Now working at home directory...
PATH: /Users/Chris-Dunamis/Documents/Portfolios/Python/Projects/FreeCodeCamp - Pandas & Python for Data Analysis
```

In [ ]:
```
import pandas as pd
```

In [ ]:
```
print("Pandas Version:", pd.__version__)
```

```
Pandas Version: 2.1.4
```

In [ ]:
```
df = pd.read_csv("./Datasets/English Words/words_pos.csv", index_col=0)

df  # After downloading the "Dataset: values_pos.csv", this is what it looks like...
```

Out[ ]:

	word	pos_tag
0	aa	NN
1	aaa	NN
2	aah	NN
3	aahed	VBN
4	aahing	VBG
...	...	...
370097	zwinglianism	NN
370098	zwingianist	NN
370099	zwitter	NN
370100	zwitterion	NN
370101	zwitterionic	NN

370100 rows × 2 columns

In [ ]:
```
# For the purpose of the online tutorial, Pandas & Python for Data Analysis by Example — Full Course for Beginners:
# https://www.youtube.com/watch?v=gtjAHBuaPNA, the code below aims to clean the dataset to resemble the dataset used for
# the online tutorial:

# (Step 1) - This code changes the Dataframe from the one above. In the online tutorial, there are three Dataframes:
#          - Word
#          - Char Count
#          - Value
#
# And since the "Dataset: words_pos.csv" already has two "Dataframes: word and pos_tag", the code below modifies this:
df.columns = ["Word", "Char Count"]
df["Char Count"] = df["Word"].apply(lambda sum_of_characters: len(sum_of_characters))

# (Step 2) - The code below aims to address the third column, as seen on the online tutorial. The "Column: Value", is
#            the sum of each char's value, for instance:
#                                 ┌─────────┬──────┬───────┐
#                                 │ Example │ Char  │ Value │
#                                 ├─────────┼──────┼───────┤
#                                 │         │  a   │   1   │
#                                 │         ├──────┼───────┤
#                                 │    1    │  a   │   1   │
#                                 │         ├──────┼───────┤
#                                 │         │  h   │   8   │
#                                 ├─────────┼──────┼───────┤
#                                 │         │ Sum  │   3   │
#                                 ├─────────┼──────┼───────┤
#                                 │         │  a   │   1   │
#                                 │         ├──────┼───────┤
#                                 │    2    │  h   │   8   │
#                                 │         ├──────┼───────┤
#                                 │         │  r   │   3   │
#                                 ├─────────┼──────┼───────┤
#                                 │         │ Sum  │   6   │
#                                 │         ├──────┼───────┤
#                                 │         │  h   │   8   │
#                                 │         ├──────┼───────┤
#                                 │    3    │  r   │  10   │
#                                 │         ├──────┼───────┤
#                                 │         │  e   │   5   │
#                                 ├─────────┼──────┼───────┤
#                                 │         │ Sum  │  23   │
#                                 └─────────┴──────┴───────┘

# In order to achieve this, an empty list is initialized here, in order to assign the sum for each word from the
char_value_results = []  # "Column: Word" in the dataset; in which the results are appended from the designed code below.

# A dictionary is initialized here, assigning the values for each char:
char_values = {"a": 1, "b": 2, "c": 3, "d": 4, "e": 5, "f": 6, "g": 7, "h": 8, "i": 9, "j": 10, "k": 11, "l": 12,
               "m": 13, "n": 14, "o": 15, "p": 16, "q": 17, "r": 18, "s": 19, "t": 20, "u": 21, "v": 22, "w": 23,
               "x": 24, "y": 25, "z": 26}

for word in df["Word"]:  # Then we traverse each word from the "Column: Word", in the dataset, and get the sum of char
    total_char_values = sum(char_values[char] for char in word)  # values from this line of code.

    # And the sum of word is appended into the defined empty list:
    char_value_results.append(total_char_values)

# Finally, we can display the sum for each word in the given row, on to a "Column: Value":
df["Value"] = char_value_results

# The changes are now saved on to a new csv file, and that is going to be the new dataset file used for this tutorial:
df.to_csv("./Datasets/English Words/words.csv", index=False)  # Specify index=False if you don't want to save row indices
```

In [ ]:
```
df = pd.read_csv("./Datasets/English Words/words.csv")

df  # When calling a dataset, like this, the dataset object returns the concatenation of the head and tail.
```

Out[ ]:

	Word	Char Count	Value
0	aa	2	2
1	aaa	3	3
2	aah	3	10
3	aahed	5	19
4	aahing	6	40
...	...	...	...
370095	zwingianism	12	156
370096	zwingianist	12	163
370097	zwitter	7	121
370098	zwitterion	10	159
370099	zwitterionic	12	171

370100 rows × 3 columns

In [ ]:
```
# The "function: head()" returns the first |n| rows (default=5) for the object based on position. For negative values of
df.head()  # |n|, this function returns all rows except the last |n| rows.
```

Out[ ]:

	Word	Char Count	Value
0	aa	2	2
1	aaa	3	3
2	aah	3	10
3	aahed	5	19
4	aahing	6	40

In [ ]:
```
# This "function: tail()" returns the last |n| rows (default=5) for the object based on position. For negative values of
df.tail()  # |n|, this function returns all rows except the first |n| rows.
```

Out[ ]:

	Word	Char Count	Value
370095	zwingianism	12	156
370096	zwingianist	12	163
370097	zwitter	7	121
370098	zwitterion	10	159
370099	zwitterionic	12	171

In [ ]:
```
df[:10]  # Equivalent to df.head()
```

Out[ ]:

	Word	Char Count	Value
0	aa	2	2
1	aaa	3	3
2	aah	3	10
3	aahed	5	19
4	aahing	6	40
5	aahs	4	29
6	aal	3	14
7	aalii	5	32
8	aaliis	6	51
9	aals	4	33

In [ ]:
```
df[-20:]  # Equivalent to df.tail()
```

Out[ ]:

	Word	Char Count	Value
370080	zuni	4	70
370081	zunian	6	85
370082	zunyite	7	120
370083	zurvis	5	89
370084	zuparate	8	104
370085	zurich	6	85
370086	zurlite	7	111
370087	zutugil	7	116
370088	zuurveldt	9	149
370089	zuza	4	74
370090	zwanziger	9	129
370091	zwieback	8	80
370092	zwieback	9	99
370093	zwieselite	10	133
370094	zwinglian	9	115
370095	zwingianism	12	156
370096	zwingianist	12	163
370097	zwitter	7	121
370098	zwitterion	10	159
370099	zwitterionic	12	171

Activities

(1): How many elements does this dataframe have?

In [ ]:
```
# The "function: info()" prints a concise summary of a DataFrame. This method prints information about a DataFrame
df.info()  # including the index dtype and columns, non-null values and memory usage.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 370100 entries, 0 to 370099
Data columns (total 3 columns):
#   Column      Non-Null Count   Dtype
---  ------      --------------   -----
0   Word        370100 non-null  object
1   Char Count  370100 non-null  int64
2   Value       370100 non-null  int64
dtypes: int64(2), object(1)
memory usage: 8.5+ MB
```

In [ ]:
```
df.shape  # This returns a tuple representing the dimensionality of the DataFrame.
```

Out[ ]:
```
(370100, 3)
```

(2): What is the value of the word `humuhumunukunukuapuaa` ?

In [ ]:
```
df.loc[df["Word"]=="humuhumunukunukuapuaa"]
```

Out[ ]:

	Word	Char Count	Value
145096	humuhumunukunukuapuaa	21	300

(3): What is the highest possible value of a word?

In [ ]:
```
df.max()
```

Out[ ]:
```
Word          zyzzyvas
Char Count          21
Value              300
dtype: object
```

In [ ]:
```
highest_word_string = df["Word"].max()
highest_word_value = df["Value"].max()

print("From the cell above, we can confirm that the highest possible value of a word is:" + "\n" +
      "    Word: {word}".format(word=highest_word_string) + "\n" +
      "    Value: {value}".format(value=highest_word_value))
```

```
From the cell above, we can confirm that the highest possible value of a word is:
    Word: zyzzyvas
    Value: 300
```

In [ ]:
```
df.describe()
```

Out[ ]:

	Char Count	Value
count	370100.000000	370100.000000
mean	9.442553	110.955231
std	2.916930	40.764817
min	1.000000	2.000000
25%	7.000000	82.000000
50%	9.000000	107.000000
75%	11.000000	136.000000
max	21.000000	390.000000

(4): Which of the following words have a Char Count of `7` and a Value of `87` ?

- pinfish
- superhetrodyne
- glowing
- enfold
- microbrew

In [ ]:
```
df.loc[
    df["Word"].isin([
        "pinfish",
        "superhetrodyne",
        "glowing",
        "enfold",
        "microbrew"
    ])
]
```

Out[ ]:

	Word	Char Count	Value
100235	enfold	6	56
126800	glowing	7	87
237010	pinfish	7	81
311975	superhetrodyne	15	196

In [ ]:
```
print("The following word with a Char Count of '7' and a Value of '87' is:")
df.loc[df["Word"] == "glowing"]
```

```
The following word with a Char Count of '7' and a Value of '87' is:
```

Out[ ]:

	Word	Char Count	Value
126800	glowing	7	87

(5): What is the highest possible length of a word?

In [ ]:
```
highest_word_char_count = df["Char Count"].max()

print("We can confirm that the highest possible length of a word is:" + "\n" +
      "    Word:       {word}".format(word=highest_word_string) + "\n" +
      "    Char Count: {char_count}".format(char_count=highest_word_char_count))
```

```
We can confirm that the highest possible length of a word is:
    Word:       zyzzyvas
    Char Count: 21
```

(6): What is the word with the Value of `377` ?

The word `humuhumunukunukuapuaa` is the only word with a Value of `300`.
Find the only word with a Value of `377` :

In [ ]:
```
df.loc[df["Value"] == 377]
```

Out[ ]:

	Word	Char Count	Value
139035	hydroxyethylorthocresotoluene	28	377