CSCC11 - Tutorial 2

What is Pandas?

Pandas is Python library which you can use to load, clean, transform and analyze your data. The library makes it easy for you to work with your dataset before you train a machine learning model on it.

Installation:

- *Terminal:* Traverse into your working directory and run `pip install pandas`
- *Jupyter Notebook:* Run `%pip install pandas` or `!pip install pandas` in a cell

Import Convention: `import pandas as pd`

DataFrame: A datatype within Pandas which is used to store two-dimensional, size-mutable, potentially heterogeneous tabular data. For example, in a clinical dataset, you may have string values (e.g., patient name, hospital name), numerical values (e.g., age, blood pressure) as well as Boolean values (e.g., whether the patient is diabetic).

Construct a DataFrame:

- *Construct a DataFrame from a dictionary:* `df = pd.DataFrame(data=d)`
  - `d` is a dictionary (e.g., `d = {'col1': [1, 2], 'col2': [3, 4]}`)
- *Read CSV into DataFrame:* `df = pd.read_csv('data.csv')`
  - If the CSV file is not in your working directory, pass the path to the file into the function.
  - Refer to Pandas documentation for more information regarding additional optional parameters: https://pandas.pydata.org/docs/reference/api/pandas.read_excel.html
- *Read Excel file into DataFrame:* `df = pd.read_excel('data.xlsx')`
  - Refer to Pandas documentation for more information regarding additional optional parameters: https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html

Some useful functions:

- *View the first few rows of a DataFrame:* `df.head(n)` where `n` is the number of rows
  - If no parameter is passed in, the first 5 rows are displayed.
- *View the last few rows of a DataFrame:* `df.tail(n)` where `n` is the number of rows
  - If no parameter is passed in, the first 5 rows are displayed.
- *View a random row from your DataFrame:* `df.sample()`
- *Get a quick summary of the DataFrame:* `df.info()`
- *Get column labels of a DataFrame:* `df.columns`
- *Get all values in a DataFrame as an array:* `df.values`
- *Get a statistical summary (i.e., min, max, mean, std dev, etc.) about the data:* `df.describe`
- *Query the DataFrame:* `df.query(expression)`
  - E.g., `df.query("Quantity < 100")` returns the rows where the value in the `Quantity` column is less than 100.
- *Access specific cell(s) in the DataFrame:* `df.loc[row_labels, column_names]`
  - `row_label` and `column_name` are the row number and column name of the cell you want to access, respectively

    - Refer to the Pandas documentation for more detailed information: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.loc.html
- *Access specific cell(s) using row and column numbers:* `df.iloc[row_nums, column_nums]`
    - Refer to the Pandas documentation for more detailed information: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.iloc.html
- *Get a list of unique values in a column:* `df[column_name].unique()`
- *Get the number of unique values in a column:* `df[column_name].nunique()`
    - `df.nunique()` will return the number of unique values in each column
- *Check which cells have missing values:* `df.isnull()`
    - Returns a DataFrame of the same size as `df` but with `True` in the cells that are missing in `df` and `False` otherwise
- *Fill missing values in DataFrame:* `df = df.fillna(s)` where `s` is the value that the null cells will be filled with
- *Sort rows in DataFrame based on values in a column:* `df.sort_values(col_name)`
- *Count how many times each value appears in a column:* `df.value_counts(col_name)`
- *Check DataFrame for a condition:* `df.where(condition)`
    - Returns a DataFrame that is the same as `df` except all cells that do not satisfy condition are set to `NaN`
- *Remove specific rows or columns from DataFrame:* `df.drop(label, axis=a)`
    - Convention for `axis` is the same as in NumPy
- *Concatenate two DataFrames:* `df = pd.concat([df1, df2])`
    - Refer to the Pandas documentation for more detailed information: https://pandas.pydata.org/docs/reference/api/pandas.concat.html#pandas.concat
- *Save DataFrame as CSV:* `df.to_csv('output.csv')`

In-class Exercise:

For this exercise, we will use `cities.csv` (https://people.math.sc.edu/Burkardt/datasets/csv/csv.html) as our main dataset and write Python code to address the following:

1. Load `cities.csv` as a DataFrame.
2. View the first 12 rows of the dataset.
3. Store all the states found in the dataset in the form of a list (make sure there are no duplicates).
4. Assuming the `cities.csv` dataset contains all North American cities (which it does not), in which states is there a city named Wilmington?

   Challenge: Once you have found the answer, try coming up with a one-line solution.

5. [*Time-permitting*] `cities_missing.csv` is the same as our original dataset except that it has some missing values. Create a new version of `cities_missing.csv` where the missing entries are filled with the term "Missing".

Exercise to be submitted:

Using the same dataset (`cities.csv`), write Python code that prints out a list of all the cities in North Carolina (state abbreviation is NC).