Chapter1_BasicDataHandling

October 11, 2023

1 Chapter I: Basic Data Handling

```
[ ]: # step 0:
     #    py -m pip install pandas
     #    py -m pip install numpy

     # import pandas
     import pandas as pd # alias with a shorter name for reference
     import numpy as np
```

1.0.1 Importing Data Sets from Files

The first step in any data science project is get data into a programming environment. Functions from Python package *pandas* can be used to import microsoft excel and csv files.

To import a data file into a programming environment, the path to the file needs to be specified. You need to know where files are stored on your computer, and how to navigate to them. Here is an example of how to import a data file into Python.

```
[ ]: path = "/Users/avery/OneDrive/Documents/GitHub/Clinical_TLB_2023-2024/
       ↪Python_for_Data_Science/Iris.csv"

     iris_df = pd.read_csv(path)
```

The *path* variable stores the location to the file being imported. Passing it to the pd.read_csv function allows the file to be located imported into the Python environment. The iris dataset now stored as a pandas dataframe in a variable called *iris_df*.

1.0.2 Investigating Dataframes

You can view the iris dataset by calling the variable it was assigned to.

```
[ ]: iris_df
```

```
[ ]:      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
     0     1            5.1           3.5            1.4           0.2
     1     2            4.9           3.0            1.4           0.2
     2     3            4.7           3.2            1.3           0.2
     3     4            4.6           3.1            1.5           0.2
```

```
4      5             5.0           3.6            1.4           0.2
..   …             …           …            …           …
145  146             6.7           3.0            5.2           2.3
146  147             6.3           2.5            5.0           1.9
147  148             6.5           3.0            5.2           2.0
148  149             6.2           3.4            5.4           2.3
149  150             5.9           3.0            5.1           1.8

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              …
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
```

You can view the top n rows by using the head() method. Or the bottom n rows by using the tail() method.

```
[ ]: iris_df.head(5)
```

```
[ ]:    Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
     0   1            5.1           3.5            1.4           0.2  Iris-setosa
     1   2            4.9           3.0            1.4           0.2  Iris-setosa
     2   3            4.7           3.2            1.3           0.2  Iris-setosa
     3   4            4.6           3.1            1.5           0.2  Iris-setosa
     4   5            5.0           3.6            1.4           0.2  Iris-setosa
```

Or the bottom n rows by using the tail() method.

```
[ ]: iris_df.tail(5)
```

```
[ ]:       Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
     145  146            6.7           3.0            5.2           2.3
     146  147            6.3           2.5            5.0           1.9
     147  148            6.5           3.0            5.2           2.0
     148  149            6.2           3.4            5.4           2.3
     149  150            5.9           3.0            5.1           1.8

                 Species
     145  Iris-virginica
```

```
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica
```

You can view the column in the iris dataset by using the columns attribute.

```
[ ]: iris_df.columns
```

```
[ ]: Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
            'Species'],
           dtype='object')
```

You can confirm that this dataframe is a pandas dataframe by calling the type command.

```
[ ]: type(iris_df)
```

```
[ ]: pandas.core.frame.DataFrame
```

You can check the data type of each column by calling the dtypes attribute.

```
[ ]: iris_df.dtypes
```

```
[ ]: Id                 int64
     SepalLengthCm    float64
     SepalWidthCm     float64
     PetalLengthCm    float64
     PetalWidthCm     float64
     Species           object
     dtype: object
```

1.0.3 Basic Dataframe Actions

Using the *pandas* package, you can merge dataframes together, add rows/columns, perform column-wise calculations, and sort rows.

```
[ ]: # import both halves of the iris dataset
     path = '/Users/avery/OneDrive/Desktop/Fall_2023/STA485/tbl_lc/
      ↪Python_For_DataScience/iris_attributes.xlsx'
     iris_attributes = pd.read_excel(path)

     path = '/Users/avery/OneDrive/Desktop/Fall_2023/STA485/tbl_lc/
      ↪Python_For_DataScience/iris_types.xlsx'
     iris_types = pd.read_excel(path)
```

Two pandas dataframes can be combined into a single dataframe using the *join* method. The *left_on* and *right_on* parameters allows you to specify which column exists in both dataframes to match rows together correctly.

```
[ ]: iris_all = iris_types.merge(iris_attributes, left_on='Id', right_on='Id')

     iris_all.head(5)
```

```
[ ]:    Id      Species  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
     0   1  Iris-setosa            5.1           3.5            1.4           0.2
     1   2  Iris-setosa            4.9           3.0            1.4           0.2
     2   3  Iris-setosa            4.7           3.2            1.3           0.2
     3   4  Iris-setosa            4.6           3.1            1.5           0.2
     4   5  Iris-setosa            5.0           3.6            1.4           0.2
```

Rows can be added to pandas dataframes by using the .loc method. A new row is added to a dataframe by appending it after the last row.

```
[ ]: # create a new row as a list
     new_row = [151, "New Flower", 5.0, 3.15, 1.2, 0.1]

     # add list to the end of the dataframe as a new row
     iris_all.loc[len(iris_all.index)] = new_row

     #
     iris_all.tail(1)
```

```
[ ]:       Id     Species  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
     150  151  New Flower            5.0          3.15            1.2           0.1
```

We can add a column to a pandas dataframe declaring a new column and assigning values to it.

```
[ ]: iris_all['new_column'] = iris_all['SepalLengthCm'] - iris_all['SepalWidthCm']

     iris_all.head(2)
```

```
[ ]:    Id      Species  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
     0   1  Iris-setosa            5.1           3.5            1.4           0.2
     1   2  Iris-setosa            4.9           3.0            1.4           0.2

        new_column
     0         1.6
     1         1.9
```

Column-wise calculations can be performed on pandas dataframes. The describe method can be used to get basic statistics about each numeric column in the dataframe.

```
[ ]: iris_all.describe()
```

```
[ ]:                Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
     count  151.000000     151.000000    151.000000     151.000000    151.000000
     mean    76.000000       5.837748      3.054636       3.741722      1.191391
     std     43.734045       0.828150      0.432217       1.770814      0.765849
```

```
min      1.000000        4.300000       2.000000        1.000000       0.100000
25%     38.500000        5.100000       2.800000        1.550000       0.300000
50%     76.000000        5.800000       3.000000        4.300000       1.300000
75%    113.500000        6.400000       3.300000        5.100000       1.800000
max    151.000000        7.900000       4.400000        6.900000       2.500000

       new_column
count  151.000000
mean     2.783113
std      0.975566
min      1.000000
25%      1.800000
50%      3.000000
75%      3.600000
max      5.100000
```

You can calculate the median of a numeric column by using the median method.

```
[ ]: iris_all['SepalLengthCm'].median()
```

```
[ ]: 5.8
```

You can calculate the sum of a numeric column by using the sum method.

```
[ ]: iris_all['SepalWidthCm'].sum()
```

```
[ ]: 461.25
```

You can sort a row based on the contents of a specific column using using the sort_values method. The ascending parameter controls the direction of sorting.

```
[ ]: iris_sorted = iris_all.sort_values(by='SepalLengthCm', ascending=False)
     iris_sorted.head(5)
```

```
[ ]:       Id         Species  SepalLengthCm  SepalWidthCm  PetalLengthCm  \
     131  132  Iris-virginica            7.9           3.8            6.4
     122  123  Iris-virginica            7.7           2.8            6.7
     118  119  Iris-virginica            7.7           2.6            6.9
     135  136  Iris-virginica            7.7           3.0            6.1
     117  118  Iris-virginica            7.7           3.8            6.7

          PetalWidthCm  new_column
     131           2.0         4.1
     122           2.0         4.9
     118           2.3         5.1
     135           2.3         4.7
     117           2.2         3.9
```