Basic Python Data Type: demo03

- Numpy arrays: must have same data type
  - np.array([1, 2, 3, 4]): a vector of length 4
  - np.array([[1, 2, 3, 4], [5, 6, 7, 8]]): a 2x4 matrix
  - np.arange(start=0, end=n+1, by=1): an integer sequence 0, 1, … n

- List: can have different data types
  - list = [element1, element2, …]

- Dictionary: can have key and value pairs
  - dict = {key1: value1, key2: value2, …}

- User defined functions
  - def function_name(arg1, arg2, …):

```
        function body
        return (value)
```

Useful String Methods: one reference

- Str.lower() or str.upper(): convert to all uppercase letters in string to lowercase or vice versa
- Str.capitalize(): capitalizes first letter of string
- Str.strip(): remove all leading and trailing white space in a string
- Str.replace(‘o,’ ‘a’): replace all “o” with “a” in a string
- Str.count(“ao”): count how many times the “ao” occurs in
- Str.split(sep=“, ”): splits the string at the specified separator, and returns a list

Create a Dataframe: demo04

- df = **pd.read_csv**(“file_name.csv”)
  - csv file needs to be in the same folder as jupyter notebook (ipynb file)
  - or include path “file_path/file_name.csv”
  - **pd.read_excel**(“file_name.xlsx”, sheet_name=“Sheet1”)

- df = **pd.DataFrame**({“col1”: [1, 2, 3], “col2”:[4,5,6], “col3”:[7,8,9]}, index=[0,1,2])
  - df = pd.DataFrame([[1,4,7],[2,5,8],[3,6,9]], columns=[“col_1”, “col2”, “col3”], index=[0,1,2])
- **df.to_csv**(“file_name.csv”): save df as a csv file

Take a look at your data: demo04

- **df.columns**: return all col names
- **df.dtypes**: return col names and data types
- **df.shape**: return (# of rows, # of cols)
    - df.shape[0] only return # of rows
- **df.index**: return the index as a series
- **df.head**(#): return first # of rows
    - df.tail(#)

Rename, Drop and Add columns: demo04

- **df.rename**(columns = {“old_col_name”:“new_col_name”, …})

- **df.drop**(columns = [“col_1”, “col_2”,…])

- **df[“new_col_name”] = new_col_content**
  - new_col_content has to be a series or a list
  - common use is to create new column based on existing ones:
  - df[“new_col_name”] = (df[“col1”] + df[“col2”]) / df[“col3”]

Sort a column and Set Index: demo04

- **df.sort_values**(“col1”): sort “col1” in ascending order
  - df.sort_values(“col1”, ascending=False)

- **df.sort_index**(): sort the index
- **df.reset_index**(): reset index to row numbers
  - df.reset_index(drop=True): drop original index
- **df.set_index**(“col1”): set “col1” to be the index

Subset a DataFrame: demo04

- Select only rows or only cols: **df[]**
  - df[“col1”]: return col1 as a series
  - df[[“col1”]]: return col1 as a df
  - df[[“col1”, “col2”, …]]: return multiple cols as a df
  - df[df[“col1”]>10]: filtering on rows where col1>10

- Select rows and cols together: **df.loc**
  - df.loc[df[“col1”]>10, [“col1”, “col2”, …]]
- Select rows and cols by position: **df.iloc**
  - df.iloc[10:20, [1,2,5]]

Data Summary: demo05

- df[“cat_col”].**value_counts**()
  - count number rows in each category for categorical varible “cat_col”
- **df.describe()**
  - basic summary statistics for all columns of numerical variables

- **df.groupby**(by=“cat_col”).[[“num_cols”]].aggfunc()
- **df.pivot_table**(index=“cat_col1”, columns=“cat_col2”, values=“num_col”, aggfunc=“mean”)
  - common aggfunc: count, sum, mean, median, min, max, var, std…

Join DataFrames: demo05

- **df1.merge**(df2, how=“inner”, left_on=“df1_col”, right_on=“df2_col”)
  - inner: only keep rows that df1 matched df2
  - outer: keep all rows in df1 and df2
  - left: keep all rows in df1, but only matching rows in df2
  - right: keep all rows in df2, but only matching rows in df1

- **df1.append**(df2)
  - append rows of df1 and df2 together
  - only if df1 and df2 have all identical columns (same variables)

Missing Values: demo05

- **df.isna().any()**: check missing value for each column
  - df.isna().sum(): count missing values for each column
  - df[df[“col1”].isna()]: return rows of df where “col1” is missing
- **df.dropna()**: remove rows with any column having NaN
  - df.drop_duplicates(): remove duplicate rows
- **df.fillna**(value): replace all NaN with value
  - df[“col1”].fillna(col1_mean)
- **df.replace**({“?”:np.nan, “*”:np.nan, …}): replace certain values with NaN

Data Visualization: demo06

- **Histogram**:
  - df[“col”].plot(kind=“hist”, bins=…, density=True)
  - plots.hist(list_or_series, bins=…)
- **Line** plot: df.plot(x=“col1”, y=“col2”)
- **Scatter** plot: df.plot(kind=“scatter”, x=“col1”, y=“col2”)
- **Bar** plot: df[“col”].value_counts().plot(kind="bar")
  - df.pivot_table(…).plot(kind="bar")
- **Box** plot: df.plot(kind=“box”, y= “col”)
  - sns.boxplot(data=df, x=“cat_col”, y=“num_col”): side-by-side box plot