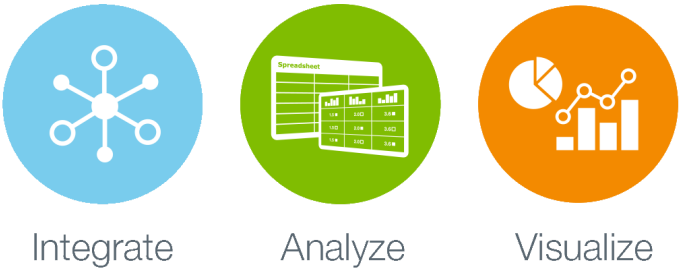
Unit 1 - Data Handling

Introduction



Data Analysis

The process of inspecting, cleaning transforming and modelling the data, intending to extract useful information for decision-making. For example, how temperature is changing in different regions or how petrol prices are varying and on what factors their variation depends.

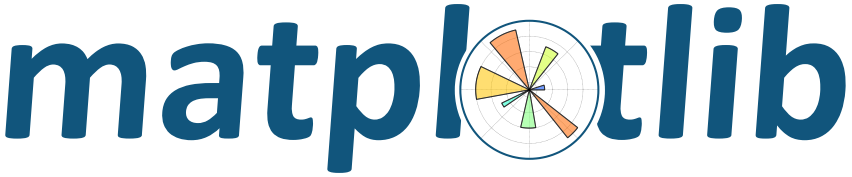
Data Visualization

The process of showing data as graphical elements, like charts, graphs and maps. Our eyes can quickly identify pattern if shown as visual. For example if we take petrol prices in tabular form and in visual form. Visual form is much more easier for identify all low and high points than tabular.

Pandas

Pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, **real world** data analysis in Python.

Matplotlib



Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Matplotlib produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, web application servers, and various graphical user interface toolkits.

Jupyter Notebook

The *Jupyter Notebook* is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Installation of Library

```
pip install pandas matplotlib
```

OR

```
python -m pip install pandas matplotlib
```

OR

If installing from **Jupyter Notebook**

```
!pip install pandas matplotlib
```

Pandas

Datastructure in Pandas

1. **Series**: one-dimensional labeled array capable of holding data of any type. The axis labels are collectively called index.
2. **DataFrame:** two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns).

Pandas Series

Creation of Pandas Series

Pandas Series can be created with scalar value, list, dictionary or 1D Numpy Array

pandas.Series(data, index)

```
# importing pandas library with alias pd
import pandas as pd

# Creating Pandas Series with Scalar Value
scalar = 3
s1 = pd.Series(scalar)


# Creating Pandas Series with List
list1 = [5,6,8,2,1]
s2 = pd.Series(list1)


# Creating Pandas Series with Dictionary
dict1 = {'Alex': 57, 'Anna':55, 'Sophia':56}
s3 = pd.Series(dict1)


# Creating Pandas Series with Numpy 1D Array
arr1 = np.array([50,35,15,22,51])
s4 = pd.Series(arr1)
```

Inspecting Pandas Series

head() - use for inspecting first 5 (if no number specified) rows of series

tail() - use for inspecting last 5 (if no number specified) rows of series

```
# importing Pandas with alias pd
import pandas as pd

# Creating List of 00 elements with List Comprehension
list1 = [x for x in range(100)]

# Creaing Pandas Series
s = pd.Series(list1)

# HEAD FUNCTION
# Using head() without any Parameter or Argument
s.head() #It will show first 5 rows

# Using head() with Parameter or Argument
s.head(2) # It will show first 2 rows
s.head(10) # It will show first 10 rows


# TAIL FUNCTION
# Using tail() without any Parameter or Argument
s.tail() # It will show last 5 rows

# Using tail() with Parameter or Argument
s.tail(3) # It will show last 3 rows
s.tail(15) # It will show last 15 rows
```

Selection (Indexing & Slicing) of Pandas Series

loc: use for selection of Pandas Series with index label.

iloc: use for selection of Pandas Series with regular index

```
# importing Pandas with alias pd
import pandas as pd

# Creating Dictionary
dict1 = {'Alex': 50, 'Smith': 51, 'Steve': 49, 'Julia': 55, 'Nina': 60}

# Creaing Pandas Series with Dictionary
# Index Label will be the Dictionary Key
# Value will be Dictionary Value
s = pd.Series(dict1)

# Indexing
s.iloc[0] # return Value at Index 0
s.loc['Nina'] # return value at Index Label Nina

# Slicing
s.iloc[1:4] # return a slice of series from index 1 to 3
# excluding index 4
s.loc['Smith':'Julia'] # return a slice of series from index label
# "Smith" to "Julia", without excluding Julia

# Slicing with Step/Interval
s.iloc[1:4:2] # slice by skipping every 2 values in-between
s.iloc[::2] # slice from starting to end of series by skipping
# every 2 values in-between.
```

Mathematical Operations on Series

```
# importing Pandas with alias pd
import pandas as pd

# Creating Pandas Series
s1 = pd.Series([1,2,3,4,5])
s2 = pd.Series([4,5,6,7,8])

# Addition
s1.add(s2)

# Subtraction
s1.add(s2)

# Multiplication
s1.mul(s2)

# Division
s1.div(s2)

# Applying mathematical operation without swapping variable
# Instead of applying operation on s1 with s2,
# it will operate on s2 with s1

# Addition -
s1.radd(s2)

# Subtraction
s1.rsub(s2)

# Multiplication
s1.rmul(s2)

# Division
s1.rdiv(s2)
```

Statistical and Aggregate Functions

```
# importing Pandas with alias pd
import pandas as pd

# Creaing Pandas Series
s = pd.Series([1,2,3,4,5])

# Sum - return sum of all values in series
s.sum()

# Count - return number of values in series
s.count()

# Min - return minimum value from series
s.min()

# Max - return maximum value from series
s.max()

# Mean - return average value from series
s.mean()

# Median - return mid value from series
s.median()

# Mode - return value with highest frequency from series
s.mode()

# std - return standar deviation from series
s.std()

# Quantile - return quartile(25%, 50%, 75%) from series
s.quantile([.25, .5, .75])
```

Pandas DataFrame

Creation of Pandas DataFrame

pandas.DataFrame(data, index, columns)

```
# importing Pandas with alias pd
import pandas as pd

# importing Numpy with alias np for creating 2D Array
import numpy as np

# Creating Pandas DataFrame with List
list1 = [1,2,3,4,5,6]
df1 = pd.DataFrame(list1) # only givingData parameter

# Creating Pandas DataFrame with Dictionary
dict1 = {'Name':'Alex', 'City':'New York','Country':'USA'}
# when dictionary is provided as data. Keys are used as Column Name
# and Values as row value, also index must be provided.
df2 = pd.DataFrame(dict1, index=[0])

# Creating Pandas DataFrame with 2D Numpy Array
arr1 = np.array([[1,2,3],[4,5,6]]) # Number of nested list inside
# represent total number of rows
# Pandas DataFrame with all three parameters
df3 = pd.DataFrame(arr1, index = ['A','B'], columns = ['Col1', 'Col2'])

# Creating Pandas DataFrame from list of dictionaries
record1 = {'Name':['Alex','Bob','Steve'],
          'City':['New York','Washington','Boston'],
          'Department': ['IT','R&D','Accounts']}
record_df1 = pd.DataFrame(record1)

record2 = [{'Name':'Alex', 'City':'New York'},
           {'Name':'Bob', 'City':'Washington'},
           {'Name':'Steve', 'City':'Boston'}]
record_df2 = pd.DataFrame(record2)

# Creating Pandas DataFrame from Pandas Series
s1  = pd.Series([1,2,3,4,5])
s2  = pd.Series([1,4,9,16,25])
df4 = pd.DataFrame([s1, s2])
```

Inspecting Pandas DataFrame

head() - used for inspecting first 5 (if no number specified) rows of DataFrame

tail() - used for inspecting last 5 (if no number specified) rows of DataFrame

info() - used to print a concise summary of a DataFrame

```
# importing Pandas with alias pd
import pandas as pd


# Creating List with List Comprehension
list1 = [x for x in range(20)]
list2 = [x for x in range(31,51)]
list3 = [x for x in range(61,81)]


# Creating Pandas DataFrame from list
df = pd.DataFrame([list1, list2, list3])

# Using info()
df.info() # Give concise information about DataFrame

# HEAD FUNCTION
# Using head() without any Parameter or Argument
df.head() # It will show first 5 rows

# Using head() with Parameter or Argument
df.head(2) # It will show first 2 rows
df.head(10) # It will show first 10 rows

# TAIL FUNCTION
# Using tail() without any Parameter or Argument
df.tail() # It will show last 5 rows

# Using tail() with Parameter or Argument
df.tail(3) # It will show last 3 rows
df.tail(15) # It will show last 15 rows
```

Selection (Indexing & Slicing) of Pandas DataFrame

loc: use for selection of Pandas DataFrame with index label.

iloc: use for selection of Pandas DataFrame with regular index

```
# importing Pandas with alias pd
import pandas as pd

# Creating Dictionary
record = {'Name':['Alex','Bob','Steve','John'],
          'City':['New York','Washington','Boston','Florida'],
          'Department': ['IT','R&D','Accounts','Engineering']}

# Creaing Pandas Series with Dictionary
# Index Label will be the Dictionary Key
# Value will be Dictionary Value
df = pd.DataFrame(record, index=['A','B','C'])

# Indexing
df.iloc[0] # return Values at Index 0
df.loc['A'] # return Values at Index Label A

# Slicing
df.iloc[0:3] # return a slice of series from index 0 to 2
# excluding index 3
df.loc['A':'B'] # return a slice of series from index label
# "A" to "B", without excluding B

# Slicing with Step/Interval
df.iloc[1:4:2] # slice by skipping every 2 values in-between
df.iloc[::2] # slice from starting to end of series by skipping
# every 2 values in-between.

# Extracting Columns
df['Name'] # return back Series with Name Column
df[['Name']] #return back DataFrame with Name Column
df[['Name','City']] # return back DataFrame with Name & City Columns
```

Iteration

The repetition of a process in order to generate a sequence of outcomes.

- **iteritems()** − to iterate over the (key,value) pairs
- **iterrows()** − iterate over the rows as (index,series) pairs
- **itertuples()** − iterate over the rows as namedtuples

```
# importing Pandas with alias pd
import pandas as pd

# importing Numpy with alias np
import numpy as np

# Using Numpy randn function from random class for creating
# two arrays with 50 random numbers
arr1 = np.random.randn(50)
arr2 = np.random.randn(50)

# Creating Pandas DataFrame with the two arrays above
df = pd.DataFrame([arr1,arr2])

# Using iteritems(), printing out all keys and values
for key, value in df.iteritems():
    print(key)
    print(value)

# Using iterrows(), printing out all index and rows
for index, row in df.iterrows():
    print(index)
    print(row)

# Using itertuples(), printing out all tuples
for t in df.itertuples():
    print(t)
```

Operations on Rows & Columns

Adding Rows

DataFrame.append(row)

```
# importing Pandas with alias pd
import pandas as pd

# Creating Dictionary
record = {'Name':['Alex','Bob','Steve','John'],
          'City':['New York','Washington','Boston','Florida']}
# Creating Pandas DataFrame
record_df = pd.DataFrame(record)

# Adding new row
row = ['Smith', 'Chicago']
record_df.append(row)
```

Adding Columns

- Adding new column by proving list
- Adding new column by providing dictionary
- Adding new column with insert() function
- Adding new column with assign() function

```
# importing Pandas with alias pd
import pandas as pd

# Creating Dictionary
record = {'Name':['Alex','Bob','Steve','John'],
          'City':['New York','Washington','Boston','Florida']}

# Creating Pandas DataFrame
record_df = pd.DataFrame(record)

# Creating new Column - Department
department = ['IT','R&D','Accounts','Engineering']

# Adding new column by proving list
record_df['Department'] = department

# Creating new Column - Salary
salary = {'John': 2000, 'Steve': 1550, 'Bob': 1560, 'Alex': 1800}

# Adding new column by providing dictionary & map function
record_df['Salary'] = record_df['Name'].map(salary)

# Creating new Column - Age
age = [25,30,24,26]

# Adding new column with insert() function
# insert(index, Column Name, data)
record_df.insert(1, 'Age', age)

# Creating new Column - Zip Code
zipCode = ['10001', '98092', '02101', '32003']
# Adding new column with assign() function
# assign(ColumnName = data)
record_df.assign(ZipCode = zipCode)
```

Deleting Row and Column

DataFrame.drop()

```
# importing Pandas with alias pd
import pandas as pd

# Creating Dictionary
employee = {'Name':[Alex','Bob','Steve','John'],
          'Age': [30,26,31,25]
          'City':['Delhi','Mumbai','Boston','Chicago'],
          'Country': ['India', 'India', 'USA', 'USA']}

# Creating Pandas DataFrame
employee_df = pd.DataFrame(employee)

# Deleting Row 1
employee_df.drop([0])

# Deleting Rows 1 & 3
employee_df.drop([0,2])


# Deleting Column - City
employee_df.drop(['City'], axis=1)
# OR
employee_df.drop(columns=['City'])

# Deleting Columns - City & Country
employee_df.drop(['City', 'Country'], axis=1)
# OR
employee_df.drop(columns=['City', 'Country'])
```

Manipulating Index and Column Labels

```
# importing Pandas with alias pd
import pandas as pd

# Creating Dictionary
record = {'Name':['Alex','Bob','Steve','John'],
          'City':['New York','Washington','Boston','Florida']}

# Creating Pandas DataFrame
record_df = pd.DataFrame(record)

# Getting Current Index
record_df.index

# Setting New Index
record_df.index = ['a','b']

# Resetting Index to default 0 based
record_df.reset_index()


# Getting Current Column Names
record_df.columns

# Setting New Column Names
record_df.columns = ['Employee Name', 'Employee City']
```

Boolean Operations

```
# importing Pandas with alias pd
import pandas as pd

# Creating Dictionary
employee = {'Name':['Alex','Bob','Steve','John'],
          'Age': [30,26,31,25]
          'City':['Delhi','Mumbai','Boston','Chicago'],
          'Country': ['India', 'India', 'USA', 'USA']}

# Creating Pandas DataFrame
employee_df = pd.DataFrame(employee)

# Applying Boolean Filtering
employee_df[employee_df['Age'] < 30] # return DataFrame with only
# rows having age less than 30

# Boolean Reduction

# empty() function - return True if DataFrame is empty else False
df = pd.DataFrame()
df.empty() # True
employee_df.empty() # False


# any() - return True if any values in a row are True
df1 = pd.DataFrame([[1,0],[0,0]])
df1.any()

# Output
# 0 True
# 1 False

# all() - return True if all values in a row are True
df2 = pd.DataFrame([[1,0],[1,1]])
df2.all()

# Output
# 0 False
# 1 True
```

Working with CSV File

A **comma-separated values (CSV)** file is a delimited text file that uses a comma (as a delimeter) to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas.

Importing CSV File as Pandas DataFrame

pandas.read_csv('file_name.csv', delimiter)

```
# importing Pandas with alias pd
import pandas as pd

# Reading Employee.csv as DataFrame
employee_df = pd.read_csv("Employee.csv")

# Reading Employee.csv delimited with | as DataFrame
employee_df = pd.read_csv("Employee.csv", delimeter="|")
```

Exporting CSV file From DataFrame

DataFrame.to_csv("File_name.csv", index=bool)

```
# importing Pandas with alias pd
import pandas as pd

employee = {'Name':[Alex','Bob','Steve','John'],
          'Age': [30,26,31,25]
          'City':['Delhi','Mumbai','Boston','Chicago'],
          'Country': ['India', 'India', 'USA', 'USA']}

# Creating DataFrame
employee_df = pd.DataFrame(employee)

# Exporting employee_df to CSV File inclusing Index
employee_df.to_csv('employee.csv', index=True)

# Exporting employee_df to CSV File exclusing Index
employee_df.to_csv('employee.csv', index=False)
```