Handout 3

Reading Files

Reading data from CSV(comma separated values) is a fundamental necessity in Data Science. Often, we get data from various sources which can get exported to CSV format so that they can be used by other systems. The Panadas library provides features using which we can read the CSV file in full as well as in parts for only a selected group of columns and rows.

Input as CSV File

The csv file is a text file in which the values in the columns are separated by a comma. Let's consider the following data present in the file named **input.csv**.

You can create this file using windows notepad by copying and pasting this data. Save the file as **input.csv** using the save As All files(*.*) option in notepad.

```
id,name,salary,start_date,dept
1,Rick,623.3,2012-01-01,IT
2,Dan,515.2,2013-09-23,Operations
3,Tusar,611,2014-11-15,IT
4,Ryan,729,2014-05-11,HR
5,Gary,843.25,2015-03-27,Finance
6,Rasmi,578,2013-05-21,IT
7,Pranab,632.8,2013-07-30,Operations
8,Guru,722.5,2014-06-17,Finance
```

Reading a CSV File

The **read_csv** function of the pandas library is used read the content of a CSV file into the python environment as a pandas DataFrame. The function can read the files from the OS by using proper path to the file.

```
import pandas as pd
data = pd.read_csv('path/input.csv')
```

```
print (data)
```

When we execute the above code, it produces the following result. Please note how an additional column starting with zero as a index has been created by the function.

```
   id    name  salary  start_date       dept
0   1    Rick  623.30  2012-01-01         IT
1   2     Dan  515.20  2013-09-23  Operations
2   3   Tusar  611.00  2014-11-15         IT
3   4    Ryan  729.00  2014-05-11         HR
4   5    Gary  843.25  2015-03-27    Finance
5   6   Rasmi  578.00  2013-05-21         IT
6   7  Pranab  632.80  2013-07-30  Operations
7   8    Guru  722.50  2014-06-17    Finance
```

Reading Specific Rows

The **read_csv** function of the pandas library can also be used to read some specific rows for a given column. We slice the result from the read_csv function using the code shown below for first 5 rows for the column named salary.

```
import pandas as pd

data = pd.read_csv('path/input.csv')


# Slice the result for first 5 rows

print (data[0:5]['salary'])
```

When we execute the above code, it produces the following result.

```
0   623.30
1   515.20
```

```
2    611.00
3    729.00
4    843.25
Name: salary, dtype: float64
```

Reading Specific Columns

The **read_csv** function of the pandas library can also be used to read some specific columns. We use the multi-axes indexing method called **.loc()** for this purpose. We choose to display the salary and name column for all the rows.

```
import pandas as pd
data = pd.read_csv('path/input.csv')

# Use the multi-axes indexing funtion
print (data.loc[:,['salary','name']])
```

When we execute the above code, it produces the following result.

```
   salary    name
0  623.30    Rick
1  515.20     Dan
2  611.00   Tusar
3  729.00    Ryan
4  843.25    Gary
5  578.00   Rasmi
6  632.80  Pranab
7  722.50    Guru
```

Reading Specific Columns and Rows

The **read_csv** function of the pandas library can also be used to read some specific columns and specific rows. We use the multi-axes indexing method

called **.loc()** for this purpose. We choose to display the salary and name column for some of the rows.

```
import pandas as pd

data = pd.read_csv('path/input.csv')


# Use the multi-axes indexing funtion

print (data.loc[[1,3,5],['salary','name']])
```

When we execute the above code, it produces the following result.

```
  salary   name
1  515.2    Dan
3  729.0   Ryan
5  578.0  Rasmi
```

Reading Specific Columns for a Range of Rows

The **read_csv** function of the pandas library can also be used to read some specific columns and a range of rows. We use the multi-axes indexing method called **.loc()** for this purpose. We choose to display the salary and name column for some of the rows.

```
import pandas as pd

data = pd.read_csv('path/input.csv')


# Use the multi-axes indexing funtion

print (data.loc[2:6,['salary','name']])
```

When we execute the above code, it produces the following result.

```
  salary    name
2 611.00   Tusar
3 729.00    Ryan
```

```
4  843.25    Gary
5  578.00   Rasmi
6  632.80  Pranab
```