Foundations of Data Science using Python

Session 2: Understanding Python Data Structures and Pandas

Introduction to Python Data Structures

Data Structure is a way to organize and store data such that we can access and modify it efficiently. Let us look into the non-primitive data structures in python, the lists, tuples, sets and dictionaries.

Python List

A list in python is a heterogeneous container for items. Few operations that we can perform with the python lists are:

- Declaration
- Accessing the whole list
- Accessing individual items in the list
    - Positive indexing
    - Negative indexing
- Slicing
- Reassigning
- Deleting

Python Tuple

A tuple in python is also a heterogeneous container for items like python list. The primary difference between list and tuple is that a tuple is immutable whereas a list is mutable. While you can reassign or delete an entire tuple, you cannot do the same to a single item or a slice. Few operations that we can perform with the python tuples are:

- Declaration, Packing and unpacking a tuple
- Accessing
- Reassigning and Deleting
- Deleting

Python Set

A set in python is like a mathematical set. It is mutable. It does not hold duplicate values and is unordered. Hence, indexing is not appropriate in accessing and deleting the elements in the python set.

Python Dictionary

A real-life dictionary holds word-meaning pairs, like wise a python dictionary holds key-value pairs. However, you may not use an unhashable item as a key. To declare a python dictionary, we use curly braces. It has key-value pairs instead of single values that differentiates a dictionary and a set in python.

Pandas Library

Pandas is an open-source, BSD-licensed Python library high performance data manipulation and data analysis using its powerful data structures. Python with pandas is in use in a variety of academic and commercial domains, including Finance, Economics, Statistics, Advertising, Web Analytics etc. Using pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data – load, organize, manipulate.

Key Features of Pandas

Some important features of Pandas, specifically used for Data Processing and Data Analysis:

- Fast and efficient DataFrame object with default and customized indexing
- Tools for loading data into in-memory data objects from different file formats
- Data alignment and integrated handling of missing data
- Reshaping and pivoting of data sets
- Label-based slicing, indexing and subsetting of large data sets
- Columns from a data structure can be deleted or inserted
- Group by data for aggregation and transformations
- High performance merging and joining of data
- Time Series functionality

Pandas Data Structures

Data Structure	Dimensions	Description
Series	One	One dimensional labelled homogeneous array, the size is immutable
DataFrames	Two	Two dimensional labelled, size is mutable, tabular structure, capable of holding heterogeneous typed columns

Mostly, we will use, pandas data frame for data handlingPandas data frame is a two-dimensional data structure i.e., data is aligned in a tabular fashion in rows and columns. The features of the data frame are potentially the columns are of different types, the size is mutable, labelled axes (rows and columns), arithmetic operations can be performed on rows and columns.

Creating a Pandas DataFrame

The parameters of data frame are data, index, columns, data type of each column, location where the data shall be copied. We can create an empty data frame, data frame from Lists, data frame from Dictionary, indexed data frame as shown below.

Creating an empty dataframe

```
#creating an empty dataframe
import pandas
MyDataFrame = pandas.DataFrame()
print(MyDataFrame)
```

Output:

```
F:\DataScienceFoundations>py UnderstandPandas.py
Empty DataFrame
Columns: []
Index: []
```

Creating a dataframe from a List

```
#creating a dataframe from a List
import pandas
MyListData = [1,2,3,4,5]
MyDataFrame = pandas.DataFrame(MyListData)
print(MyDataFrame)
```

Output

```
F:\DataScienceFoundations>py UnderstandPandas.py
   0
0  1
1  2
2  3
3  4
4  5
```

Creating a dataframe from List of Lists

```
#creating a dataframe from List of Lists
import pandas
MyListData = [['Nischal',7],['Nihal',5],['Nihaan',1],['Diyaan',6]]
MyDataFrame = pandas.DataFrame(MyListData,columns=['Name','Age'])
print(MyDataFrame)
```

Output:

```
F:\DataScienceFoundations>py UnderstandPandas.py
      Name  Age
0  Nischal    7
1    Nihal    5
2   Nihaan    1
3   Diyaan    6
```

Creating a dataframe from Dictionary of ndarrays or Lists

```
#creating a dataframe from Dictionary of ndarrays or Lists
'''
All the ndarrays must be of same length.
If index is passed, then the length of the index should equal to the length of the arrays.
If no index is passed, then by default, index will be range(n), where n is the array length.
Note - Observe the values 0,1,2,3. They are the default index assigned to each using the function range(n).
'''
import pandas
MyDictData = {'Name':['Nishcal', 'Nihal', 'Nihaan', 'Shubham'],'Age':[7,5,1,4]}
MyDataFrame = pandas.DataFrame(MyDictData)
print ("\n",MyDataFrame)
```

Output:

```
F:\DataScienceFoundations>py UnderstandPandas.py

       Name  Age
0  Nishcal    7
1    Nihal    5
2   Nihaan    1
3  Shubham    4
```

Creating a dataframe with indexing from a Dictionary

```
#Creating an Indexed dataframe
import pandas
MyDictData = {'Name':['Durgesh','Indrasen', 'Chandan Pandey', 'Premchand'],'CGPA':[9.83,9.79,9.62,7.25]}
#the index parameter assigns an index to each row
MyDataFrame = pandas.DataFrame(MyDictData, index=['Rank1','Rank2','Rank3','Rank4'])
print ("\n",MyDataFrame)
```

Output:

```
F:\DataScienceFoundations>py UnderstandPandas.py

                   Name  CGPA
Rank1          Durgesh  9.83
Rank2         Indrasen  9.79
Rank3  Chandan Pandey  9.62
Rank4        Premchand  7.25
```

Case Study: Creating a sample students data in a CSV file

The primary objective of this case study is to focus on creating a sample data set of students with the attributes viz., Registration No., Name, Gender, Department, Email, Contact number. The python libraries used in creating a student sample dataset are 'csv', 'os', 'random' which are deployed with the basic python installation and the library 'names' shall be explicitly downloaded using pip as shown below:

```
F:\DataScienceFoundations>pip install names
Collecting names
  Using cached names-0.3.0.tar.gz (789 kB)
Using legacy setup.py install for names, since package 'wheel' is not installed.
Installing collected packages: names
    Running setup.py install for names ... done
Successfully installed names-0.3.0
```

Python Code for creating a sample dataset of 5000 students

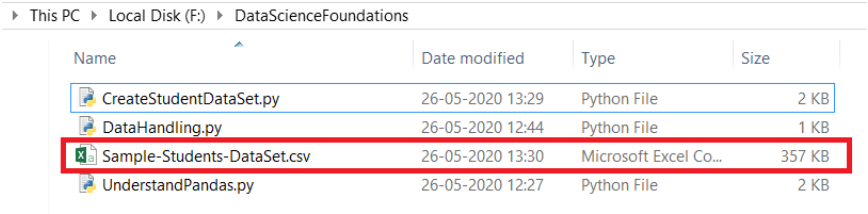
CreateStudentDataSet.py

```python
import csv
import os
import random
import names
delimiter = ','

myCSVFileName = input('Enter a CSV Filename: ')

fileExists = os.path.isfile(myCSVFileName)


#Open the CSV File in append mode for creating Users details
with open(myCSVFileName,'a',newline='') as appendInToCSVFile:
    csvHeader = ['REGISTRATION_NUMBER','STUDENT_NAME','STUDENT_GENDER','STUDENT_DEPARTMENT','STUDENT_EMAILID','STUDENT_CONTACT_NUMBER']
    MyHeader = csv.DictWriter(appendInToCSVFile,fieldnames=csvHeader)
    if not fileExists:
            MyHeader.writeheader()

    ID = 2017000001
    while (ID <= 2017005000):
        studentId = ID
        appendInToCSVFile.write(str(studentId))
        appendInToCSVFile.write(delimiter)
        firstName = names.get_first_name()
        lastName = names.get_last_name()
        studentName = firstName+' '+lastName
        appendInToCSVFile.write(studentName)
        appendInToCSVFile.write(delimiter)
        studentGender = random.choice(['Male','Female'])
        appendInToCSVFile.write(studentGender)
        appendInToCSVFile.write(delimiter)
        studentDepartment = random.choice(['CSE','ECE','EEE','Mechanical','Civil','EIE'])
        appendInToCSVFile.write(studentDepartment)
        appendInToCSVFile.write(delimiter)
        emailTag = random.choice(['@gmail.com','@yahoo.co.in','@rediffmail.com','@hotmail.com'])
        studentEmailId = firstName+lastName+emailTag
        appendInToCSVFile.write(studentEmailId)
        appendInToCSVFile.write(delimiter)
        studentContactNum = random.randint(9986000000,9986999999)
        appendInToCSVFile.write(str(studentContactNum))
        appendInToCSVFile.write('\n')
        ID+=1
```

Output:

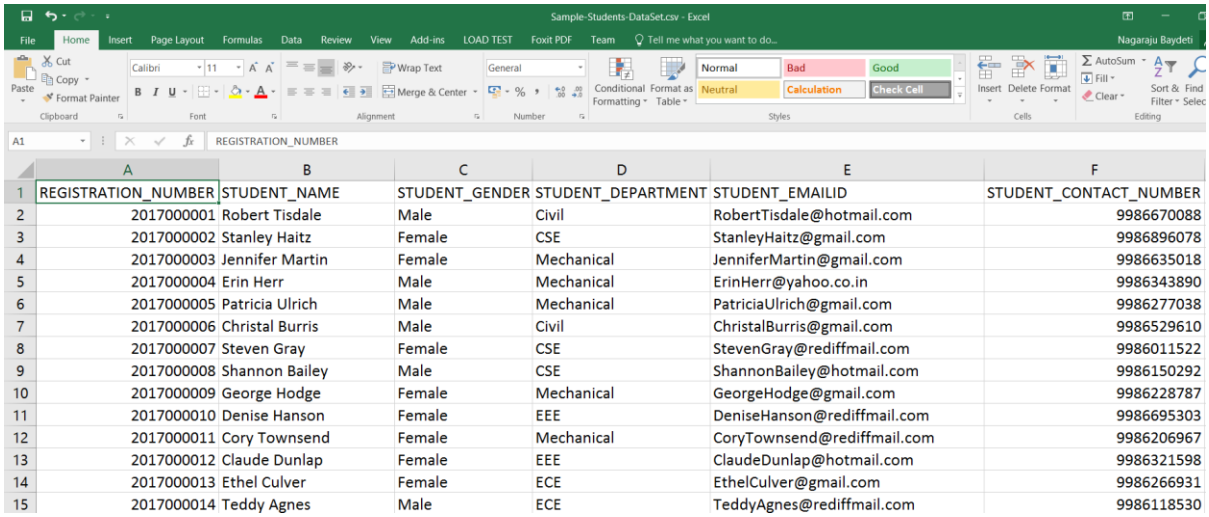
The program upon the successful execution, in the current directory a sample data set in terms of a comma-separated-value file is created.

```
F:\DataScienceFoundations>py CreateStudentDataSet.py
Enter a CSV Filename: Sample-Students-DataSet.csv

F:\DataScienceFoundations>
```



Sample-Students-DataSet.csv

Few records from the sample data set:



REGISTRATION_NUMBER	STUDENT_NAME	STUDENT_GENDER	STUDENT_DEPARTMENT	STUDENT_EMAILID	STUDENT_CONTACT_NUMBER
2017000001	Robert Tisdale	Male	Civil	RobertTisdale@hotmail.com	9986670088
2017000002	Stanley Haitz	Female	CSE	StanleyHaitz@gmail.com	9986896078
2017000003	Jennifer Martin	Female	Mechanical	JenniferMartin@gmail.com	9986635018
2017000004	Erin Herr	Male	Mechanical	ErinHerr@yahoo.co.in	9986343890
2017000005	Patricia Ulrich	Male	Mechanical	PatriciaUlrich@gmail.com	9986277038
2017000006	Christal Burris	Male	Civil	ChristalBurris@gmail.com	9986529610
2017000007	Steven Gray	Female	CSE	StevenGray@rediffmail.com	9986011522
2017000008	Shannon Bailey	Male	CSE	ShannonBailey@hotmail.com	9986150292
2017000009	George Hodge	Female	Mechanical	GeorgeHodge@gmail.com	9986228787
2017000010	Denise Hanson	Female	EEE	DeniseHanson@rediffmail.com	9986695303
2017000011	Cory Townsend	Female	Mechanical	CoryTownsend@rediffmail.com	9986206967
2017000012	Claude Dunlap	Female	EEE	ClaudeDunlap@hotmail.com	9986321598
2017000013	Ethel Culver	Female	ECE	EthelCulver@gmail.com	9986266931
2017000014	Teddy Agnes	Male	ECE	TeddyAgnes@rediffmail.com	9986118530

Data Handling using Pandas

In general, reading data from CSV file is a fundamental necessity in Data Science. Data from various sources can be exported in terms of CSV format so that they can be used for data handling, analysis and visualization. The pandas library provides, features using which we can read the CSV file in full as well as in parts for only selected group of columns and rows.

Creation of a pandas dataframe from reading the data from CSV dataset file:

The **read_csv()** method of the pandas library is used to read the data of a CSV file into python environment as a python pandas dataframe. The file path can be passed as an argument to the **read_csv()** method where the relative path is considered when filename alone is given. In case if you wish to read data from a file outside the current directory, the absolute path shall be provided as an argument. The **shape** tuple returns

the dimensions of the data that is loaded as dataframe in terms of number of rows and columns. The method **head()** is used to extract the data from the dataframe. In case we need to extract the first fifteen rows from the dataset, we can invoke **head()** method on the data frame with 15 as an argument to the method **head()**.

Python code - loading the CSV data, knowing the dimension and extraction of data

DataHandling.py

```python
#Creating a data frame from CSV file
import pandas
#reading the data from a csv file using read_csv() method
MyDataFrame = pandas.read_csv('Sample-Students-DataSet.csv')
Total_Rows_Columns = MyDataFrame.shape
#Displaying the shape tuple
print('\nThe dimensions of the data set are: ',Total_Rows_Columns)
#Displaying
print('\n\nThe Total number of instances are',Total_Rows_Columns[0])
print('The Total number of attributes are',Total_Rows_Columns[1])
#extracting the piece of data using head() method
HeadRows = MyDataFrame.head(15)
print('\n\nThe first 15 rows in the dataset')
print(HeadRows)
```

Output:

```
F:\DataScienceFoundations>py DataHandling.py

The dimensions of the data set are:  (5001, 6)


The Total number of instances are 5001
The Total number of attributes are 6


The first 15 rows in the dataset
    REGISTRATION_NUMBER      STUDENT_NAME STUDENT_GENDER STUDENT_DEPARTMENT              STUDENT_EMAILID  STUDENT_CONTACT_NUMBER
0            2017000001    Robert Tisdale           Male              Civil    RobertTisdale@hotmail.com              9986670088
1            2017000002     Stanley Haitz         Female                CSE       StanleyHaitz@gmail.com              9986896078
2            2017000003   Jennifer Martin         Female         Mechanical     JenniferMartin@gmail.com              9986635018
3            2017000004         Erin Herr           Male         Mechanical         ErinHerr@yahoo.co.in              9986343890
4            2017000005   Patricia Ulrich           Male         Mechanical     PatriciaUlrich@gmail.com              9986277038
5            2017000006   Christal Burris           Male              Civil     ChristalBurris@gmail.com              9986529610
6            2017000007       Steven Gray         Female                CSE    StevenGray@rediffmail.com              9986011522
7            2017000008    Shannon Bailey           Male                CSE    ShannonBailey@hotmail.com              9986150292
8            2017000009      George Hodge         Female         Mechanical        GeorgeHodge@gmail.com              9986228787
9            2017000010     Denise Hanson         Female                EEE  DeniseHanson@rediffmail.com              9986695303
10           2017000011     Cory Townsend         Female         Mechanical  CoryTownsend@rediffmail.com              9986206967
11           2017000012     Claude Dunlap         Female                EEE     ClaudeDunlap@hotmail.com              9986321598
12           2017000013      Ethel Culver         Female                ECE        EthelCulver@gmail.com              9986266931
13           2017000014      Teddy Agnes           Male                ECE    TeddyAgnes@rediffmail.com              9986118530
14           2017000015  Armando Boardman         Female         Mechanical    ArmandoBoardman@gmail.com              9986831491

F:\DataScienceFoundations>
```

Performing operations around a variable

Certain operations can be performed on a variable. For instance, let us understand how to group data on a variable. The **groupby()** method is helpful in grouping the data based on a dataset attribute.

```python
#Performing groupby operations on the dataframe
import pandas
#reading the data from a csv file using read_csv() method
MyDataFrame = pandas.read_csv('Sample-Students-DataSet.csv')
Result = MyDataFrame.groupby('STUDENT_DEPARTMENT').size()
print(Result)
```

Here, the groupby() is performed based on the department of the student and the size() method returns the number of students in each department from the dataset.

Output:

```
F:\DataScienceFoundations>py DataHandling.py
STUDENT_DEPARTMENT
CSE            825
Civil          851
ECE            814
EEE            855
EIE            832
Mechanical     823
dtype: int64

F:\DataScienceFoundations>
```

Total rows: 5000