**Name :Abdul Basit Reg#210201023**

**Name:Abdullah Ashraf Reg#210201061**

**Name:Ahmed Zafar Reg#210201092**

**Name:Abdul Rehman Reg#210201088**

**Name:Muhammad Zaryab Reg#210201007**

**Pandas**

This chapter focus on the python library pandas

**Introduction to Pandas**

Pandas is a python library which is used for working with data sets, it has functions for analyzing, cleaning, exploring and manipulating data. The name **Pandas** has a reference to both panel data and Python data analysis and was created by **WES McKinney** in 2008.

**Pandas** provides high-level data structures and functions designed to make working with structured or tabular data fast, easy and expressive. Since its emergence in 2008, it helped enable python to be a powerful and productive data analysis environment. The primary objects in pandas that will be used in this book are the Data Frame, a tabular, column-oriented data structure with both row and column labels, and the Series, a one-dimensional labeled array object

**Uses of Pandas**

**Pandas** allows us to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets and make them readable and relevant. Relevant data is very important in data science.

**What Pandas can do?**

**Pandas** gives you answers about the data. Like is there is any correlation between two or more columns average, max, min value. Pandas can also be able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is also called **cleaning data.**

**Installation of Pandas**

If Python and pip is already installed on a system, then installation of Pandas is very easy just open you command line or terminal and run a simple command **pip install pandas.**

**Pip** is a package manager for python. That means it’s a tool that allows you to install additional libraries and dependencies

Throughout the rest of the book, I use the following import convention for pandas:

import pandas as pd

Thus, whenever you see pd. in code, it’s referring to pandas. You may also find it easier to import Series and Data Frame into the local namespace since they are so frequently used:

from pandas import series, DataFrame

**Pandas Data Structures**

There are two types of data structures in Pandas:

1. **Series**
2. **DataFrame**

**Series**

Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called index. You can create a series by calling **pandas.Series().** For example, the following series is a collection of integers 10, 23, 56, …

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | 23 | 56 | 17 | 52 | 61 | 73 | 90 | 26 | 72 |

**Key Points**

* Homogeneous data
* Size Immutable
* Values of Data Mutable

**pandas.Series**

A pandas Series can be created using the following constructor-

|  |
| --- |
| pandas.Series( data, index, dtype, copy) |

The parameters of the constructors are as follows

|  |  |
| --- | --- |
| **Sr.No** | **Parameters & Description** |
| **1** | **data:**  data takes various forms like ndarray, list, constants |
| **2** | **index:**  Index values must be unique and hashable, same length as data. Default  **np.arrange(n)**  if noindex is passed. |
| **3** | **dtype:**  dtype is for data type. If None, data type will be inferred |
| **4** | **copy:**  Copy data. Default False |

The syntax for a series is:

|  |
| --- |
| import pandas as pd  s = pd.Series()  print(s) |

This creates an empty series.

**Create series from list**

To turn a list into series, all you have to do is:

|  |
| --- |
| import pandas as pd  items = [1,2,3,4]  s = pd.Series(items)  print(s) |

It outputs the contents of **‘s’** like this:

|  |
| --- |
| 0 1  1 2  2 3  3 4  dtype: int64 |

By default is assigns an index. First it shows the index, then the element value.

**DataFrame**

Pandas DataFrame is a 2-dimensional data structure, like a 2-dimensional array, or a table with rows and columns. DataFrame have rows and columns, each columns have a name, which is string. Each row has an index, which is an integer. DataFrame can contain many different data types such as strings, int. floats.

|  |
| --- |
| Import pandas as pd  data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada', 'Nevada'],  'year': [2000, 2001, 2002, 2001, 2002, 2003],  'pop': [1.5, 1.7, 3.6, 2.4, 2.9, 3.2]}  frame = pd.DataFrame(data)  print(frame) |

The resulting DataFrame will have its index assigned automatically as with Series, and the columns are placed in sorted order:

|  |
| --- |
| pop state year  0 1.5 Ohio 2000  1 1.7 Ohio 2001  2 3.6 Ohio 2002  3 2.4 Nevada 2001  4 2.9 Nevada 2002  5 3.2 Nevada 2003 |

For large Data Frames, the head method selects only the first five rows:

**head.frame()**

|  |
| --- |
| pop state year  0 1.5 Ohio 2000  1 1.7 Ohio 2001  2 3.6 Ohio 2002  3 2.4 Nevada 2001  4 2.9 Nevada 2002 |

**Locate Row**

As you can see from the result above, the Data frame is like a table with rows and columns.

Pandas use the **loc** attribute to return one or more specified rows

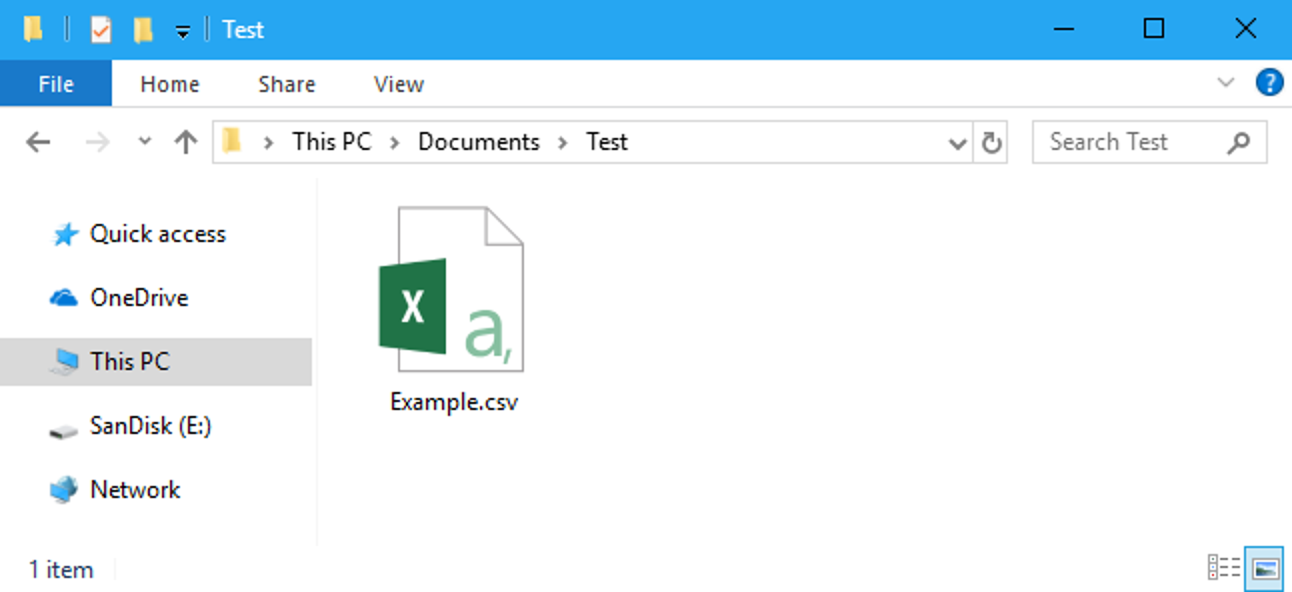
|  |
| --- |
| import pandas as pd  data = {  "calories": [420, 380, 390],  "duration": [50, 40, 45]  }  df = pd.DataFrame(data)  print(df.loc[0]) |

**Pandas Read CSV Files:**

A simple way to store big data sets in tabular format is to use CSV files (comma separated files).

CSV files contains plain text and is a well know format that can be read by everyone including Pandas.

It has (.csv) extension.



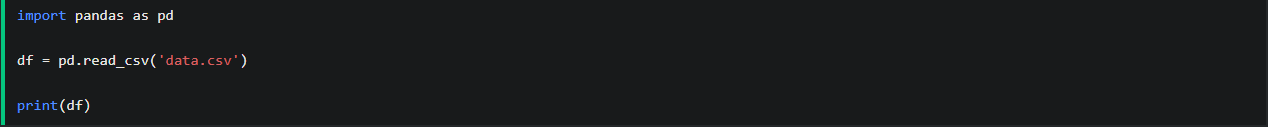
To store data in a .csv (comma separated files) we use

**df.to\_csv(‘myfile.csv)**

**Load Files into a Data Frame:**

If your data sets are stored in a file, Pandas can load them into a Data Frame.

**varaiable\_name=pd.read(‘myfile’)**



**Pandas Read JSON Files:**

JSON is plain text, but has the format of an object, and is well known in the world of programming,

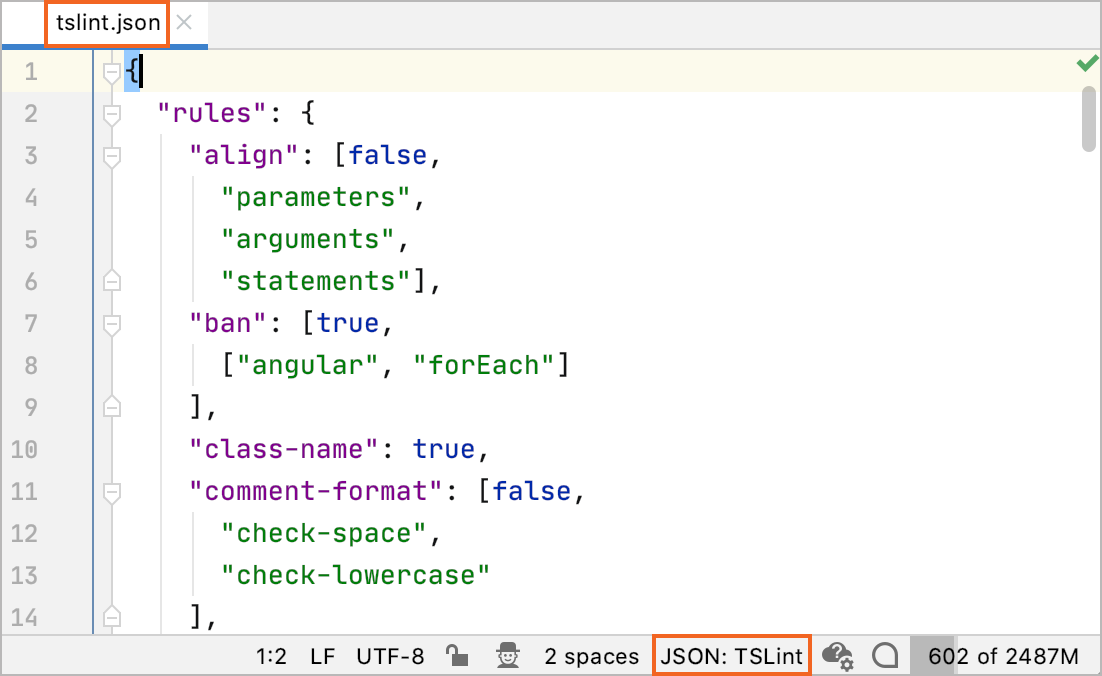
including Pandas. A JSON file is **a file that stores simple data structures and objects in JavaScript**

**Object Notation (JSON) format**, which is a standard data interchange format. It is primarily used for

transmitting data between a web application and a server.

It has (.json) extension.

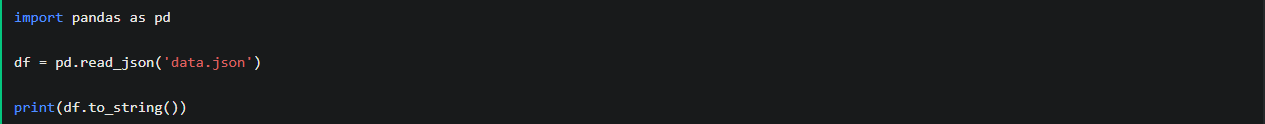
**FOR EXAMPLE**:



**Load Files into a Data Frames:**

If your data sets are stored in a file, Pandas can load them into a Data Frame.

**Variable\_name=pd.read\_json(‘myfile’)**



**Pandas Commands**

Panda has various commands for viewing, analyzing and cleaning data. Here we will try to cover some of these commands. For viewing data we have the following commands:

The **“head( )”** command is used to view top rows in data frame. This command is used because sometimes we have a lot of rows in a data frame and we want to view or show only a particular number of rows from the top for that purpose we use this command. Similarly for viewing rows from bottom in data frame we use **“tail( )”** command.

The syntax of how to write this command is **“** **xyz**.**head(n)”.** The xyz is your data frame name. **“n”** is the number of rows you want to view. If we leave the parenthesis empty by default 5 rows are printed.

print(xyz)

name train age weight

0 zaryab 909 19 70

1 basit 666 20 55

2 ahmad 333 19 72

3 abdullah 619 18 50

print(xyz.head(2))

name train age weight

0 zaryab 909 19 70

1 basit 666 20 55

print(xyz.tail(2))

name train age weight

2 ahmad 333 19 72

3 abdullah 619 18 50

**Describe command**

The describe command is used for calculations. You have a data frame of 1000 rows now calculating.

mean, average, mode, max and min values can be difficult for you in excel sheet and time consuming.

In pandas we can easily calculate all these using describe command in just blink of an eye.

The syntax for the command is **“xyz.describe( )”.**

print(xyz)

name train age weight

0 zaryab 909 19 70

1 basit 666 20 55

2 ahmad 333 19 72

3 abdullah 619 18 50

xyz.describe( )

age weight

count 4.00 4.0

mean 18.75 72.500

std 0.50 9.5

min 18 60.0

25% 18.75 67.5

50% 19.00 75.00

75% 19.00 80.00

max 19.00 80.00

**Locate command**

Locate command can be used for multiple purposes. It can be used to display, change and show particular values under some conditions. Using loc command always remember that write strings in inverted commas in square brackets.

The syntax for locate command is **“xyz.loc[row][colomn]”.**

For finding a particular value in a particular row and column we can use loc command.

print(xyz)

name train age weight

0 zaryab 909 19 70

1 basit 666 20 55

2 ahmad 333 19 72

3 abdullah 619 18 50

xyz.loc[o][‘age’]

19

\*As we can column age had value “19” at row 2. That is how locate command can display a single value.\*

\*For multiple rows and multiple values we need to change the syntax a little bit.\*

xyz.loc[:,[‘train’,’age’]

train age

0 909 19

1 666 20

2 333 19

3 619 18

\*Now in this case all the rows are printed but some columns that were defined in the command. It’s done by adding “:”. “:” indicates all rows and columns. If we want to print particular columns and all rows we add “:” on the right side after specifying the rows\*

xyz.loc[[0,1],:]

name train age weight

0 zaryab 909 19 70

1 basit 666 20 55

We can use the loc command to change the values too.

print(xyz)

name train age weight

0 zaryab 909 19 70

1 basit 666 20 55

2 ahmad 333 19 72

3 abdullah 619 18 50

xyz.loc[1,”age”] = 21

print(xyz)

name train age weight

0 zaryab 909 19 70

1 basit 666 21 55

2 ahmad 333 19 72

3 abdullah 619 18 50

\*In upper command we changed age of row “1” “basit” from 20 to 21.\*

\*Similarly if you want to change everyone’s age to 21 we can simply use “:” in place of “1”.\*

xyz.loc[:,”age”] = 21

print(xyz)

name train age weight

0 zaryab 909 21 70

1 basit 666 21 55

2 ahmad 333 21 72

3 abdullah 619 21 50

\*Similarly we have iloc command. The function of iloc is same as loc. In loc command you have to write the names of rows and columns if you have provided them particular names. Like in all above examples we had to write “age”, ”train”, ”name” but in case od iloc we can simply give index and not write the whole string.\*

**Drop command**

Drop command can be used to remove rows and columns. To indicate row we use **“axis = 0”** and to indicate column we use **“axis = 1”**. Using drop command we see that after running command the row or column mentioned is removed but if print the data frame we see that the row or column mentioned in the command is still present this phenomena is called view and copy.

(all the time we were making the changes in xyz were not implemented permanently. Pandas is always showing us a copy of xyz and we are making changes in that copy and at the same time the real file xyz is still the same and never changed. To implement a permanent change while using drop command we use attribute **“inplace = True”** inside the parenthesis of drop command)

print(xyz)

name train age weight

0 zaryab 909 19 70

1 basit 666 20 55

2 ahmad 333 19 72

3 abdullah 619 18 50

xyz.drop([0], axis=0)

name train age weight

1 basit 666 20 55

2 ahmad 333 19 72

3 abdullah 619 18 50

xyz.drop([“age”], axis=1)

name train weight

0 zaryab 909 70

1 basit 666 55

2 ahmad 333 72

3 abdullah 619 50

\*We can drop multiple rows and columns by mentioning them inside the square brackets.\*

\*Now how to implement the permanent change. We will use inplace attribute.\*

Li Like there are tons of pandas commands that can be used for data frames.

\*\*This is explanation for inplace attribute\*\*

print(xyz)

name train age weight

0 zaryab 909 19 70

1 basit 666 20 55

2 ahmad 333 19 72

3 abdullah 619 18 50

xyz.drop([0], axis=0)

name train age weight

1 basit 666 20 55

2 ahmad 333 19 72

3 abdullah 619 18 50

\*\*Now we think that the row “0” is dropped permanently but if we write\*\*

\*\* print(xyz) we can see that the row “0” is still present in xyz file.\*\*

name train age weight

0 zaryab 909 19 70

1 basit 666 20 55

2 ahmad 333 19 72

3 abdullah 619 18 50

\*\*If we use xyz.drop([0], axis=0 , inplace = True) and then write print(xyz).\*\*

\*\*We get\*\*

name train age weight

1 basit 666 20 55

2 ahmad 333 19 72

3 abdullah 619 18 50

\*\*Now the change is permanently made in the xyz file by using in place attribute.\*\*

\*\*Like there are tons of panda’s commands that can be used for data frames.

**\*\*Dtype** tells us about the type whether it’s series or data frame.\*\*

**\*\*Info** command tells us about the type of data stored in the data frame.\*\*

**\*\*Duplicate** commands shows if there are duplicate values present and can also be used to remove the rows and columns having those values. To use more commands like these you can visit w3school or use google.\*\*

ke there are tons of pandas commands that can be used for data frames.