This tutorial is on Pandas a data analysis library in Python

We will use pandas to read and clean datasets, undertake some data mining and transformation, plotting some graphs, and write our results.

#open jupyter notebook from windows search window

Series and List

Series can be understood as a 1 dimensional labelled / indexed array/list. You can access individual elements of this series through these labels.

To start with first import the pandas library

import pandas as pd

Create a series

s1 = pd.Series(['CIS6005: Computational Intelligence', 125, 36.256, 125, 256, 'Imtiaz Khan'])

You can also create a list (without using pandas)

s2 = ['CIS6005: Computational Intelligence', 125, 36.256, 'Imtiaz Khan']

Access different elements of the series

s1[1] # will print the number 125

# accessing range of elements

#s[from : upto] in no value given by default from first [0] to last element

s1[:2] # first two elements

s1[-2:] # last two elements

s1[:-3] # from the first element to element before the last three elements

s1[2:4] # from second to fourth element \* here it is index based not as count before

**TASK:** perform the same operations on s2

**TASK:** print the first string of the last element, print *Imtiaz*. HINT: s1[element location].split(' ') will split the string as per white space. Also print the module code i.e. CIS6005

The benefit of pandas series is that you can start different data mining activities with simple commands, which you cant do with list although both are 1D data structure.

s1.value\_counts() # returns how many each unique elements are

s1.value\_counts().plot(kind='bar')

You can convert a list into a series and change the index as you like

s3 = [5, 35, 58, 11]

print s3

# labelling index

s4 = pd.Series(s3, index=[2222,'x','Y', 10])

print (s4)

# access by index label

s4['x']

s4.loc['x']

# access by index number

s4.iloc[1]

# applying arithmetic function

s4\*3

Dataframes

A dataframe is similar to Excel workbook – you have column names referring to columns and you have rows, which can be accessed with use of row numbers. The essential difference being that column names and row numbers are known as column and row index, in case of dataframes.

Download (from Moodle) and save the *phone\_data.csv* file in your local folder.

Open the file and observe different attributes (column headers)

**TASK:** identify null values (empty cells).

Readingfile:

Following command will enable you to read the csv file into a dataframe (here the variable is named as *df*, which you can change).

The dot ‘.’ Symbol apply different functions like describe to the dataframe or part of it and for each function within the parenthesis () you can pass values. You can join multiple functions with dot symbols and in that case the later function will work on the returned value of the previous function.

df = pd.read\_csv('path\to\your\folder\phone\_data.csv')

# for Excel file

df = pd.read\_excel('path\to\file.xlsx')

df.head(10) # Show the top 10 rows

df.tail(5) # Show bottom 5 rows

df.shape # Shape (row, col) of the dataset/dataframe

df.describe() # Basic stat for column(s) with numerical values

# if you want to get stat about particular attribute, here we are choosing one column/attribute named 'network'

df['network'].describe()

Selection**:**

Get the column names

df.columns # col headers name

Get a particular column. # column name is case sensitive e.g. network

df['column name']

Get the first 5 rows of column name network

df[:2] # first two rows of the dataframe

# you can also access by location by *loc* or *iloc* function

df.loc[:2]

df.iloc[:2]

df.loc[0:5,'network']

If you don’t know the column name but only the position you can use iloc instead of loc

df.iloc works like iloc[row , col]

within row & col “:” sign means from : upto

If we leave off the first position value, like :5, it's assumed we mean 0. If we leave off the last position value, like 0:, it's assumed we mean the last row or column in the DataFrame.

df.iloc[:5,:] # the first 5 rows, and all of the columns for those rows.

df.iloc[:,:] # the entire DataFrame.

df.iloc[5:,5:] # rows from position 5 onwards, and columns from position 5 onwards

df.iloc[:,0] # the first column, and all of the rows for the column.

df.iloc[9,:] # the 10th row, and all of the columns for that row.

**TASK:** Show rows 35 to 40 for the last 2 columns

Cleaning:

Find out where are the null values. Following code will show you total number of null cells for each attribute. Here two functions isnull() and sum() worked sequentially joined by dot symbol. Does it match with your observation of the csv file.

df.isnull().sum()

You can delete the rows with null values and assign the cleaned dataframe to a new dataframe called *clean\_df*

clean\_df = df.dropna( how='any')

See the documentation bellow for the parameters for deleting for example how='any' will delete rows where any of the cell has null value whereas how='all' will delete rows where all cells are null.

**TASK:** check the null values in df & clean\_df as well as their shapes

Also if you don’t want to assign it to a new data frame i.e. clean\_df = then use

df.dropna(how='any', inplace=True)

Also if you know which column to has these null values you can do something like this

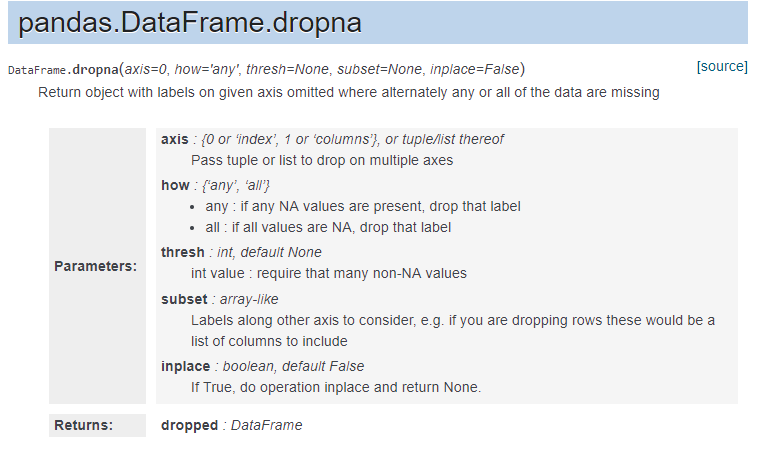
df.dropna(subset=['duration', 'network\_type'], how='any', inplace=True)

#df.dropna(how='any') #to drop if any value in the row has a nan

#df.dropna(how='all') #to drop if all values in the row are nan

For more info on dropna documentation visit

<https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.dropna.html>



Replace parts of string

df['network'] = df['network'].str.replace('Meter', 'Meteor')

TASK: replace other misspelled network names

Basic analysis

value\_counts() provides number of occurrence of items for an attribute/column

df['item'].value\_counts()

# Get the different items in a list

df['item'].value\_counts().index.tolist()

# get highest/lowest value for an attribute/column

df.loc[df['duration'].idxmax()] # idxmin() for lowest

# sort values

sorted\_df = df.sort\_values('duration')

# for high to low df.sort\_values('duration', ascending=False)

# How many seconds of phone calls are recorded in total?

df['duration'][df['item'] == 'call'].sum()

# items (call, sms, data) with duration 5 to 7 min

df[(df['duration'] >= 5) & (df['duration'] <= 7)]

**TASK**

1. Which network has the highest and lowest items (all types)
2. Which network has the highest and lowest sms
3. How many entries for each month
4. Show call (only) with duration between 34 to 35 min

Plotting Data

df['network'].value\_counts().plot(kind='bar')

# bar plot

df['item'].value\_counts().plot(kind='pie')

**TASK**

1. Set labels to your plot

for improved plotting read the documentation

<https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.html>

Writing file

df.to\_csv('path\to\file.csv')

Saving an analysis to csv file with your defined column header name

df['network'].value\_counts().to\_csv('path\to\file.csv', header=['Network ranking'])