Pandas Cheat sheet for Shay (clean examples)

- Pandas
  - Pandas data structures
    - Series
    - Dataframe
      - Creating a dataframe
      - Get dimensions of a dataframe
      - Get dataframe except specific rows
      - Get column names
      - Get column index for a given specific name
      - Dropping columns in pandas
      - Concatentaing columns and rows
      - Converting from numpy to panda
      - Converting categorical columns to numbers (4 Methods)
      - Accessing an element in pandas:
      - Methods of slicing in pandas
      - Filtering data within a dataframe
      - Method #1 (Similar to R language)
      - Method #2 (Similar to Filter function in R language)
      - Method #3 (less clean way)
  - CSV
    - Importing data from CSV
    - Exporting data into CSV
  - Displaying data cleaner
  - Get information of the data types for a dataframe
  - Get statistics (count, mean, std, min, max))
  - Get counts for spcific column
  - Datatypes conversions
  - Dealing with NA's
    - Retrieve NaN values
    - Remove rows with NA's
    - Replace NA's with the median
    - Retrieve NaN values
  - fill
  - Get the index of the min or the max element
  - Get the nsmallest or nlargest element
  - Group by:
    - Group by time slot
  - Concat Dataframes
    - Join two dataframes one below the other.
    - Join two dataframes one besides the other.
    - Printing data so all columns will be presented
  - Reference

Pandas

pandas adopts significant parts of NumPy's idiomatic style of array-based computing, especially array-based functions and a preference for data processing without for loops.

While pandas adopts many coding idioms from NumPy, the biggest difference is that pandas is designed for working with tabular or heterogeneous data. NumPy, by contrast, is best suited for working with homogeneous numerical array data.

```
import pandas as pd
```

Pandas data structures

Series

A series is one-dimensional array like object conject containing a sequense of values.

```
obj = pd.Series([4, 7, -5, 3])
```

you can use labels in the index when selecting single values or a set of values:

```
obj2 = pd.Series([4, 7, -5, 3], index=['d', 'b', 'a', 'c'])
my_matrix = pd.Series()
```

Dataframe

Creating a dataframe

```
# Converting a a simple dictionary to a dataframe

my_data_states = {'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]}

my_df1 = pd.DataFrame(my_data_states)
```

Another example:

```
my_df2 = pd.DataFrame(data = [[0,0,0],[1,2,3],[4,5,6],[7,8,9]],
                      index = range(0,4),
                      columns=['col1','col2','col3'])

print("\n",my_df2)
```

Another example:

Creating a dataframe of specific size and initializing it with -1's:

```
my_df3 = pd.DataFrame(index=range(5), columns=range(3))
my_df3 = my_df3.fillna(-1)
```

Get dimensions of a dataframe

```
df = pd.Dataframe(np.array([1,2,3],[4,5,6]))
print("\nThe shape of our dataframe is:",df.shape) # The shape of the dataframe is (2,3)
```

Get dataframe except specific rows

I'd like to get all rows **excepts rows 3 and 5**:

```
not_relevant_rows = my_df1.index.isin([3,5])
df_relevant = my_df1[~not_relevant_rows]
```

Get column names

```
print(df.columns.values)
```

Get column index for a given specific name

```
   df.columns.get_loc('my_column')
```

Dropping columns in pandas

```
df.drop('column_name',1,inplace=True)
```

Concatentaing columns and rows

concatentaing columns:

```
# Axis 1 means columns
result = pd.concat([df['person_name'], df['person_weight']], axis = 1)
```

a different approach for adding a column will be:

```
df['my_new_column'] = pd.Series(list_of_values)
```

Retriving specific columns:

```
df1 = df[['column1','column2']]
```

concatentaing rows:

```
# Here i'm concatentaing two first rows with two last rows.
result = pd.concat([df[0:2], df[-2:]], axis = 0)
```

```
# Adding a row to my_df:
my_df.loc["two"] = [4,5,6]
```

Converting from numpy to panda

```
my_2darray = np.array([[1, 2, 3], [4, 5, 6]])
print(pd.dataframe(my_2darray,columns=['a','b','c']))
```

Converting categorical columns to numbers (4 Methods)

Worked well for me!

Great reference:

Link

Accessing an element in pandas:

```
print(my_df.iloc[row_num, col_num] )
```

Methods of slicing in pandas

- `loc` get rows/columns with praticular **labels** (label-based indexing).
- `iloc` get rows/columns at praticular **index** (it only takes integers).
- `get_loc()` is and index method meaning "get the position of the label in this index"

```
df.iloc[:df.index.get_loc('row_bla') + 1, :4]
```

Filtering data within a dataframe

Method #1 (Similar to R language)

```
newdf = df[(df.column_name_1 == "JFK") & (df.colunm_name_2 == "B6")]
```

Method #2 (Similar to Filter function in R language)

```
newdf = df.query('column_name_1 == "JFK" & colunm_name_2 == "B6"')
```

Method #3 (less clean way)

```
newdf = df.loc[(df.column_name_1 == "JFK") & (df.colunm_name_2 == "B6")]
```

CSV

Importing data from CSV

```
movies_df = pd.read_csv('data/movies.csv')
movies_df.head()
```

Exporting data into CSV

```
movies_df.to_csv('./my_folder/movies.csv', index = False)
```

Displaying data cleaner

```
display(df[0:5])
```

Get information of the data types for a dataframe

```
movies_df.info()
```

Get statistics (count, mean, std, min, max))

```
df[''].describe()
```

Get counts for spcific column

```
data_df['my_column'].value_counts()
```

Datatypes conversions

```
movies_df['average rating'] = movies_df['average rating'].astype('float')
movies_df['Date'] = pd.to_datetime(movies_df['Date'])
movies_df['Star Ratings'] = movies_df['Star Ratings'].astype('int')
```

Dealing with NA's

Retrieve NaN values

```
<columnname>.notnull()
```

Remove rows with NA's

```
my_df = my_df.dropna()
```

Replace NA's with the median

```
the_median = df['horse_power'].median()
my_df['horse_power'] = my_df['horse_power'].fillna(med)
```

Retrieve NaN values

```
<columnname>.notnull()
```

fill

Get the index of the min or the max element

```
data_example = pd.Series([
        1,3,2,8,124,4,2,1
])

print('The index of the minimum value is: ', data_example.idxmin())
print('The index of the maximum value is: ', data_example.idxmax())
```

Get the nsmallest or nlargest element

```
df = pd.Dataframe({
    'Name': ['Bob', 'Mark', 'Steph', 'Jess', 'Becky'],
    'Points': [55, 98, 46, 77, 81]
})

print('The fourth element in size is:', str(df.mslargest(4,'Points')))
print('The 2nd smallest element  is:', str(df.msmallest(2,'Points')))
```

Group by:

```
# This will create a data frame object consists of
# few tables each table is seperated for each city (we have splitted the data into smaller g
my_groups = df.groupby('city')

# Running iteratively and retrieving the table for the corresponding group.
```

```
for city,city_df in my_groups:
    print(city)
    print(city_df)

# Get the dataframe of group city 'new york'
my_groups.get_group('new york')


# Apply the function max on each group:
my_groups.max()


# Get all the analytics in one shot (count, mean, std, min, max):
my_groups.describe()
```

Reference

Group by time slot

Link

Concat Dataframes

Join two dataframes one below the other.

```
import pandas as pd

israel_weather = pd.Dataframe({
   'city':['Ramat-Gan', 'Tel-Aviv', 'Haifa'],
   'tempature':['35','33','40'],
   'humidity':[60,65,75]
   })

us_weather = pd.Dataframe({
   'city':['New york', 'Boston', 'Los Angeles'],
   'tempature':['25','29','30'],
   'humidity':[40,25,55]
   })

df1 = pd.concat([israel_weather, us_weather], ignore_index = True)

# create a sub-table
df2 = pd.concat([israel_weather, us_weather], keys = ['Israel','US'])
```

```
# Retrieve the Israel dataframe:
df2.loc['Israel']
```

Join two dataframes one besides the other.

```
import pandas as pd

tempature_df = pd.Dataframe({
   'city':['New york', 'Boston', 'Los Angeles'],
   'humidity':[60,65,75]
   })

windspeed_df = pd.Dataframe({
   'city':['New york', 'Boston', 'Los Angeles'],
   'widspeed':[7,12,9]
   })

# Axis =1 means concating dataframe beside one the other dataframe
df1 = pd.concat([israel_weather, us_weather], axis = 1)

# Retrieve the Israel dataframe:
df2.loc['Israel']
```

Reference

Printing data so all columns will be presented

```
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
```

Reference

Pandas Cheat Sheet #1

Pandas Cheat Sheet #2