**How does Python Memory Management works?**

**What is use of \_\_main\_\_ in python? What happens when \_\_main\_\_ is not present?**

**What is difference between pop and remove methods?**

[**https://stackoverflow.com/questions/11520492/difference-between-del-remove-and-pop-on-lists/11520540**](https://stackoverflow.com/questions/11520492/difference-between-del-remove-and-pop-on-lists/11520540)

**What is difference between range and arange in python?**

range returns a regular list (python 2) or a specialized "range object" (like a generator; python 3)

np.arange gives a numpy array.

If you care about speed enough to use numpy, use numpy arrays.

**Map vs Apply function? Difference between Map, Apply and Applymap in Pandas?**

* **Map:** It iterates over each element of a series.  
  df[‘column1’].map(lambda x: 10+x), this will add 10 to each element of column1.  
  df[‘column2’].map(lambda x: ‘AV’+x), this will concatenate “AV“ at the beginning of each element of column2 (column format is string).
* **Apply:** As the name suggests, applies a function along any axis of the DataFrame.  
  df[[‘column1’,’column2’]].apply(sum), it will returns the sum of all the values of column1 and column2.
* **ApplyMap:** This helps to apply a function to each element of dataframe.  
  func = lambda x: x+2  
  df.applymap(func), it will add 2 to each element of dataframe (all columns of dataframe must be numeric type)

**How do you apply if else condition on data frame?**

Using apply method:

* df['name\_match'] = df['First\_name'].apply(lambda x: 'match' if x == 'Bill' else 'Mis-match')
* df['elderly'] = np.where(df['age']>=50, 'yes', 'no')
* df['color'] = df.Set.map( lambda x: 'red' if x == 'Z' else 'green')

**Write a python code snippet to sample data?**

data = data.sample(frac = 0.1, random\_state = 1)

**What is pd.get\_dummies used for?**Pandasget\_dummies function converts categorical variable into dummy/indicator variables.  
It can be used as an alternative to encoding technique called as one hot encoding.

[**Panda's get\_dummies vs. Sklearn's OneHotEncoder() :: What is more efficient?**](https://stackoverflow.com/questions/36631163/pandas-get-dummies-vs-sklearns-onehotencoder-what-is-more-efficient)OneHotEncoder cannot process string values directly. If your nominal features are strings, then you need to first map them into integers.

pandas.get\_dummies is kind of the opposite. By default, it only converts string columns into one-hot representation, unless columns are specified.

OneHotEncoder() gives us a sparse matrix.

**When do we use Label Encoder and when One Hot Encoder?**

Label encoder is used when having more than two encoded values is not going to affect model. So generally we use Label encoder for target variable. It takes less disk space as encoding is done in single column contrast to One Hot Encoder. Also Random Forrest and Decision tree work fine on Label encoders.

One Hot encoder results are binary rather than ordinal, but takes more space. The disadvantage is that for high cardinality, the feature space can really blow up quickly and you start fighting with the curse of dimensionality. In these cases, I typically employ one-hot-encoding followed by PCA for dimensionality reduction. I find that the judicious combination of one-hot plus PCA can seldom be beat by other encoding schemes. PCA finds the linear overlap, so will naturally tend to group similar features into the same feature.

**How to deal with categorical variables?**

Dummy\_encoding is used to encode categorical variables, so that our algorithms doesn’t evaluate based on encoded categorical variables. That means our model should not treat encoded categorical variable as numeric values and do comparison or other arithmetic operation based on that.

From sklearn.preprocessing import LabelEncoder  
labelencoder\_x = LabelEncoder()  
X[:, 0] = labelencoder\_X.fit\_transform(X[:, 0])

From sklearn.preprocessing import OneHotEncoder

onehotencoder\_X = OneHotEncoder(categorical\_features = [0]) //first column

X = onehotencoder.fit\_transform(X).toarray()

**Difference between for loop and while loop?**

For loop need iterator, that means we need to know how many times loop will run.

While in case of while loop it runs based on a condition, that means no need to know number of times loop will run.

**For loop:**  
for(intialization; condition; Increment or decrement){  
// statements to be excuted.  
}

**While loop:**  
while(condtion) {  
//statements to excute.  
}

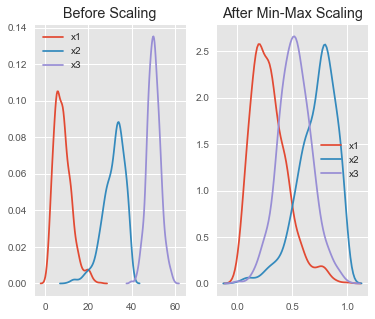
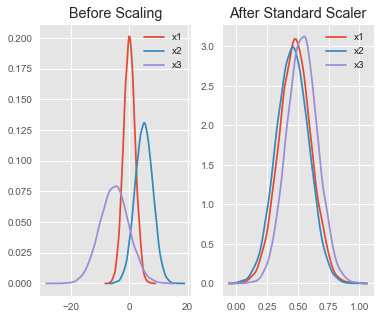
**What scaling model have you used? When to use MinMaxScalar?**

from sklearn.preprocessing import MinMaxScaler (fit, transform and fit\_transform function.)  
from sklearn.preprocessing import StandardScaler

Standard Scaler: The StandardScaler assumes your data is normally distributed within each feature and will scale them such that the distribution is now centred around 0, with a standard deviation of 1. The mean and standard deviation are calculated for the feature and then the feature is scaled based on: Xi-mean(x)/stdev(x).

The MinMaxScaler is the probably the most famous scaling algorithm, and follows the following formula for each feature:

(xi–min(x))/(max(x)–min(x)). It essentially shrinks the range such that the range is now between 0 and 1 (or -1 to 1 if there are negative values). This scaler works better for cases in which the standard scaler might not work so well. If the distribution is not Gaussian or the standard deviation is very small, the min-max scaler works better.



[**How are Pandas iloc, ix and loc different and related?**](https://stackoverflow.com/questions/31593201/how-are-pandas-iloc-ix-and-loc-different-and-related)

* loc gets rows (or columns) with particular labels from the index.
* iloc gets rows (or columns) at particular positions in the index (so it only takes integers).
* ix usually tries to behave like loc but falls back to behaving like iloc if a label is not present in the index.
* .loc is primarily label based, but may also be used with a boolean array. .loc will raise KeyError when the items are not found. Allowed inputs are: A single label, e.g. 5 or 'a', (note that 5 is interpreted as a *label* of the index. This use is **not** an integer position along the index) A list or array of labels ['a', 'b', 'c'] A slice object with labels 'a':'f', (note that contrary to usual python slices, **both** the start and the stop are included!) A boolean array A callable function with one argument (the calling Series, DataFrame or Panel) and that returns valid output for indexing (one of the above) New in version 0.18.1. See more at [Selection by Label](https://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-label)
* .iloc is primarily integer position based (from 0 to length-1 of the axis), but may also be used with a boolean array. .iloc will raise IndexError if a requested indexer is out-of-bounds, except *slice*indexers which allow out-of-bounds indexing. (this conforms with python/numpy *slice* semantics). Allowed inputs are: An integer e.g. 5 A list or array of integers [4, 3, 0] A slice object with ints 1:7 A boolean array A callable function with one argument (the calling Series, DataFrame or Panel) and that returns valid output for indexing (one of the above)

**What is difference between List and Tuple?**

List can be created using list() or using Square Bracket. List is mutable so we can add or remove or modify its values. Functions available on list are len(), It is indexed from 0. We can also use negative indexing. List also supports advance slicing like list[2:], list[:7], list[2:9:2], list[::-1]

colors=['red','blue','green']

mLIst = list (range(100, 111, 2))

A tuple is immutable and can be created using small brackets. Tuples are faster than list. We use it when we want to be sure that data will not be changed by some method in code.

numbers=(1,2,’three’)

**How do you write exponent operator in python?**numpy.exp or num1\*\* num2

**Example:** Calculate exp(x) - 1 for all elements in the array.

**Example:** Calculate 2\*\*x for all elements in the array.

**What is meshgrid function in numpy?** [What is the purpose of meshgrid in Python / NumPy?](https://stackoverflow.com/questions/36013063/what-is-the-purpose-of-meshgrid-in-python-numpy)

The purpose of meshgrid is to create a rectangular grid out of an array of x values and an array of y values.

So, for example, if we want to create a grid where we have a point at each integer value between 0 and 4 in both the x and y directions. To create a rectangular grid, we need every combination of the x and y points.

This is going to be 25 points, right? So if we wanted to create an x and y array for all of these points, we could do the following.

xvalues = np.array([0, 1, 2, 3, 4]);

yvalues = np.array([0, 1, 2, 3, 4]);

Now, when we call meshgrid, we get the previous output automatically.

xx, yy = np.meshgrid(xvalues, yvalues)

plt.plot(xx, yy, marker='.', color='k', linestyle='none')

**Pandas Series vs numpy Array?**

[**https://penandpants.com/2014/09/05/performance-of-pandas-series-vs-numpy-arrays/**](https://penandpants.com/2014/09/05/performance-of-pandas-series-vs-numpy-arrays/)

Pandas series is slower compared to numpy Array. In general pandas is built over numpy so numpy will be faster than pandas.

**Difference between modules and packages in python?**

A module is a single file (or files) that are imported under one import and used. e.g.

import my\_module

A package is a collection of modules in directories that give a package hierarchy.

from my\_package.timing.danger.internets import function\_of\_love

**How do you drop a column or row in pandas?**

Dropping row is simple as df.drop(‘indexname’), While for dropping column we need to specify axis =1 apart from column name. like df.drop(‘year’, axis =1)

**Selecting entries from data frame.**

dframe[dframe[‘c’]>10], dframe.ix[‘LA’], dframe.ix[1], dframe1.add(dframe2, fill\_value=0)

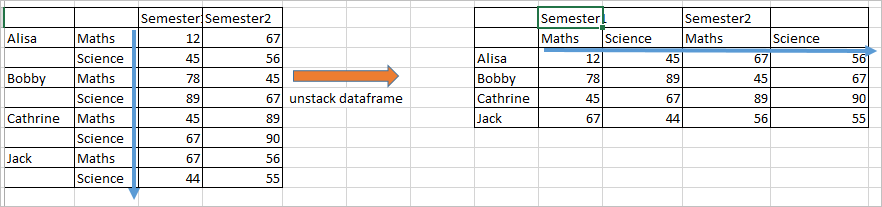
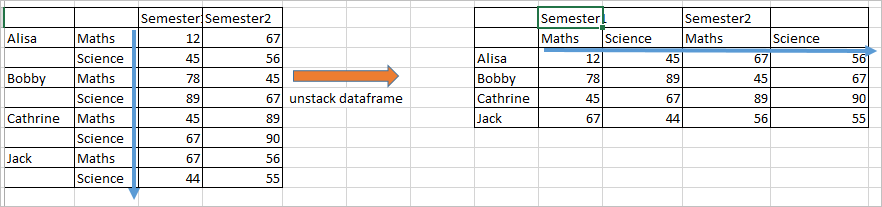
Series1.sort\_index(), series1.order(), ser2.sort(), ser2.rank(),

**How to display YouTube videos using python in Jupyter notebook?**  
From IPyton.display import YouTubeVideo  
YouTubeVideo(‘’)

**When do we use Stack and unstack method?**

Stacking a dataframe at level 1 will stack maths and science columns row wise

Stacking a dataframe at level 0 will stack semester1 and semester2 columns row wise.

   
DataFrame can be created by calling unstack method on series.

**How do you deal with Missing Data?**

There are various ways to deal with missing data. Sometimes we can use machine learning classification or Regression model to predict missing value. Here missing value records are used as test set and remaining dataset is used as training set.

Apart from this we also use imputers to impute data, first of all we will check number of missing values by using describe(), or sum().

missing\_val\_count\_by\_column = (data.isnull().sum())

print(missing\_val\_count\_by\_column[missing\_val\_count\_by\_column > 0]

data\_without\_missing\_values = original\_data.dropna(axis=1)

Checking columns that have missing values.

cols\_with\_missing = [col for col in original\_data.columns

if original\_data[col].isnull().any()]

redued\_original\_data = original\_data.drop(cols\_with\_missing, axis=1)

reduced\_test\_data = test\_data.drop(cols\_with\_missing, axis=1)

from sklearn.impute import SimpleImputer

my\_imputer = SimpleImputer()

data\_with\_imputed\_values = my\_imputer.fit\_transform(original\_data)

# make copy to avoid changing original data (when Imputing)

new\_data = original\_data.copy()

# make new columns indicating what will be imputed

cols\_with\_missing = (col for col in new\_data.columns

if new\_data[col].isnull().any())

for col in cols\_with\_missing:

new\_data[col + '\_was\_missing'] = new\_data[col].isnull()

# Imputation

my\_imputer = SimpleImputer()

new\_data = pd.DataFrame(my\_imputer.fit\_transform(new\_data))

new\_data.columns = original\_data.columns

**Can we change indexes of dataframes?**

Indexes are immutable that means we can’t change Index. We need to use reindex() to change indexes.

series1.reindex(range, method = ‘ffill’) //forwardfill

Apart from reindex function we can also use ix for the purpose of reindexing.

**What is np.linspace function used for?**

The NumPy linspace function creates sequences of evenly spaced values within a defined interval.

np.linspace(start = 0, stop = 100, num = 5)

arange(**start**,**stop**,step)

numpy.linspace(*start*, *stop*, *num=50*, *endpoint=True*, *retstep=False*, *dtype=None*)

arr.argmax()index location for max value similarly there is arr.argmin()  
np.random.randint (0, 1000, (3, 3))  
np.ramdom.seed(100)

**Can you explain combine\_first method?**

used for conditional combine, it can replace complex np.where easily.  
df1.combine\_first(df2)

**How to read JSON in Python?**

Import json

Data = json.load(json\_obj)  
json.dump(data)

Df = DataFrame(data[‘diet’])

**How to use stdout in python to show results on screen rather than saving it?**

Import sys  
dframe.to\_csv(sys.stdout, sep = ‘\_’, columns = [0,1,2])

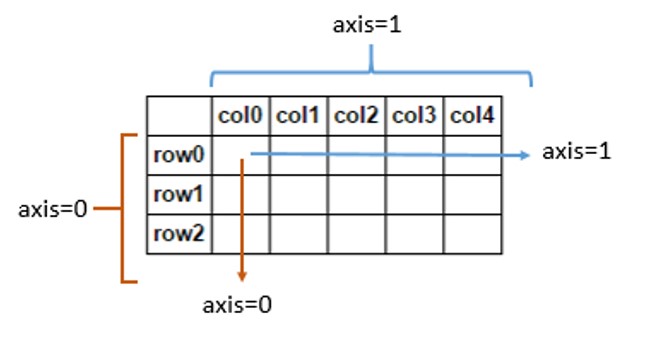
**What is difference between Slicing vs Subsetting?**

**How do you understand Axis in python?**

axis 0 = rows

axis 1 = columns

If you “sum” through axis=0, you are summing all rows, and the output will be a single row with the same number of columns. If you “sum” through axis=1, you are summing all columns, and the output will be a single column with the same number of rows



**What is Argmax() function and when do we use it?**

Returns the indices of the maximum values along an axis

**What is Eigen vector decomposition?**

**What is singular value decomposition?**

**Matrix operations and its physical meanings?**

**What is difference between Inverse vs pinverse?**

**Trigonometric vs hyperbolic trigonometric functions?**

**How do you select all elements of an Array?**

arrayAll = arr[:]

arr2d[:2,1:] row upto 2nd and column from 1st to end

**What are some Jupyter notebook shortcuts?**

* Alt+Enter to run current cell and create new cell below
* Shift+Tab to show suggestion about function parameters.

**Why do we use % matplotlib inline?**

To plot graphs in jupyter notebook

**When we use .plot() and .imshow() functions?**

.Imshow(): Display an image, i.e. data on a 2D regular raster.

.plt(): Plot y versus x as lines and/or markers.