1. **How to create objet in Python?**
2. **How to reverse an Array?**

My\_Array[::-1]

1. **How can you randomize the items of a list in place in Python?**

from random import shuffle

x = ['Keep', 'The', 'Blue', 'Flag', 'Flying', 'High']

shuffle(x)

print(x)

1. **What is a Python Generator?**
2. **How to implement main function in Python?**
3. **How to implement inheritance in Python?**
4. **What is namespace in python?**
5. **How Python is interpreted?**
6. **How memory is managed in Python?**
7. **How Zip function works, what is its output format?**
8. **What is counter in Python?**
9. **What is list comprehension in Python?**
10. **Uses of Map, Filter and Reduce function in python?**
11. **What is \_\_init\_\_?**

Ans: \_\_init\_\_ is a method or constructor in Python. This method is automatically called to allocate memory when a new object/ instance of a class is created. All classes have the \_\_init\_\_ method.

1. **Here is an example of how to use it.**

class Employee:

def \_\_init\_\_(self, name, age,salary):

self.name = name

self.age = age

self.salary = 20000

E1 = Employee("XYZ", 23, 20000)

# E1 is the instance of class Employee.

#\_\_init\_\_ allocates memory for E1.

print(E1.name)

print(E1.age)

print(E1.salary)

1. **What is self in Python?**

Self is an instance or an object of a class. In Python, this is explicitly included as the first parameter. However, this is not the case in Java where it’s optional. It helps to differentiate between the methods and attributes of a class with local variables.

The self-variable in the init method refers to the newly created object while in other methods, it refers to the object whose method was called.

1. **What is a lambda function?**

An anonymous function is known as a lambda function. This function can have any number of parameters but, can have just one statement.

a = lambda x,y : x+y

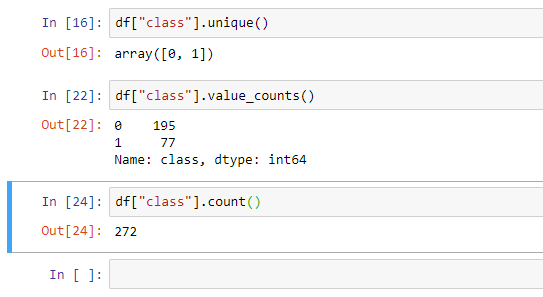
print(a(5, 6))

1. **Map vs Apply function? Difference between Map, Apply and Applymap in Pandas.**
   1. **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).
   2. **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.
   3. **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)
2. **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')

1. **How to pot bar plot of target variable?**



1. **How to perform Sampling of data?**

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

1. **What is pd.get\_dummies used for?**

Converts categorical variable into dummy/indicator variables.

1. **How to deal with categorical variables?**

Dummy encoding, 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()

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

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

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

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

To plot graphs in jupyter notebook

1. **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.

1. **What scaling model have you used?**

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

1. [**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.

1. **List vs Tuple?**

|  |  |
| --- | --- |
| List | Tuple |
| colors=['red','blue','green'] | numbers=(1,2,’three’) |
| A List is Mutable | A Tuple is Immutable   1. You can't add elements to a tuple. Tuples have no append or extend method. 2. You can't remove elements from a tuple. Tuples have no remove or pop method. 3. **Tuples are faster than lists.** If you're defining a constant set of values and all you're ever going to do with it is iterate through it, use a tuple instead of a list. 4. It makes your code safer if you “write-protect” data that does not need to be changed. Using a tuple instead of a list is like having an implied assert statement that this data is constant, and that special thought (and a specific function) is required to override that. 5. Some tuples can be used as dictionary keys (specifically, tuples that contain immutable values like strings, numbers, and other tuples). Lists can never be used as dictionary keys, because lists are not immutable. |

1. **How to give exponent to numpy array in python?**

**numpy.exp**

**expm1:** Calculate exp(x) - 1 for all elements in the array.

**Exp2:** Calculate 2\*\*x for all elements in the array.

1. **\*\* vs ^ operator in python?**
2. **arr2d[:2,1:] row upto 2nd and column from 1st to end**
3. [**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')

1. **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.

1. **How to convert series to dictionary?**

Using to\_dict() over series. Similarly, dictionary can be passed to Series function to make a series.

1. **How to read clipboard using python pandas?**

Pd.Read\_clipboard()

Indexes are immutable, that means we can’t change indexes, we need to assign new index in that case.

To reindex we use series1.reindex(range, method = ‘ffill’) function. Forward fill reindexing.

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

1. **Dropping Row vs Dropping clolumn in pandas dframe?**

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)

1. **Selecting entries from data frame.**

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

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

1. **How to display YouTube videos using python in Jupyter notebook?**

From IPyton.display import YouTubeVideo  
YouTubeVideo(‘’)

1. **How we deal with missing Data?**

[**https://www.kaggle.com/dansbecker/handling-missing-values**](https://www.kaggle.com/dansbecker/handling-missing-values)

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)

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

1. **Index Hierarchy in pandas?**

Series can have multiple index level

Similarly, data frame can have multiple level of index.

1. **DataFrame can be created by calling unstack method on series.**
2. **Correlation plot?**sns.corrplot(df, annotation = False, diag\_name = False)  
   prices.plot()
3. **Selecting all elements in a numpy array** = arr[:]
4. **Default data type is np.float64**  
    Empty () can return garbage than zero.
5. **Fancy indexing creates copy of data**
6. Argmax () what is?

Returns the indices of the maximum values along an axis

1. **Axis 0 means column, 1=row**
2. What is Eigen vector decomposition?
3. What is singular value decomposition?
4. Matrix operations and its physical meanings?
5. Inverse vs pinverse.
6. Trigonometric vs hyperbolic trigonometric functions?
7. Slicing vs Subsetting?
8. **pandas iloc vs ix vs loc explanation, how are they different?**  
   <https://stackoverflow.com/questions/31593201/pandas-iloc-vs-ix-vs-loc-explanation-how-are-they-different>
9. **How to show data on screen than saving it?**

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

1. **JSON with Python?**

Import json

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

Df = DataFrame(data[‘diet’])

1. **Data Frame Merge? What happens after merging if both df has a column with same name?**

Both columns get suffixed with x and y to distinguish source.

We can specify that suffix by passing suffixes=(‘\_lefty’, ‘\_righty’) into pd.merge

1. [**What is difference between join and merge?**](https://stackoverflow.com/questions/22676081/what-is-the-difference-between-join-and-merge-in-pandas)

DataFrames provide the pandas.DataFrame.merge() and pandas.DataFrame.join() methods as a convenient way to access the capabilities of pandas.merge(). For example, df1.merge(right=df2, ...) is equivalent to pandas.merge(left=df1, right=df2, ...).

These are the main differences between df.join() and df.merge():

1. lookup on right table: df1.join(df2) always joins via the index of df2, but df1.merge(df2)can join to one or more columns of df2 (default) or to the index of df2 (with right\_index=True).
2. lookup on left table: by default, df1.join(df2) uses the index of df1 and df1.merge(df2)uses column(s) of df1. That can be overridden by specifying df1.join(df2, on=key\_or\_keys)or df1.merge(df2, left\_index=True).
3. left vs inner join: df1.join(df2) does a left join by default (keeps all rows of df1), but df.merge does an inner join by default (returns only matching rows of df1 and df2).

So, the generic approach is to use pandas.merge(df1, df2) or df1.merge(df2). But for a number of common situations (keeping all rows of df1 and joining to an index in df2), you can save some typing by using df1.join(df2) instead.

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

Can be used for conditional combine, it can replace complex np.where easily.

1. **Df.stack(), df.unstack(‘col1’), df.unstack().stack(), df.stack(dropna=False)**
2. **Data Frame Pivoting**
3. **What is np.linspace function used for?**

Returns evenly spaced numbers over a specified interval. Example below:

**>>>** np.linspace(2.0, 3.0, num=5)

array([ 2. , 2.25, 2.5 , 2.75, 3. ])

**>>>**np.random.randint (0, 1000, (3, 3))

**>>>**np.ramdom.seed(100)

**>>>**arr.argmax()index location for max value similarly there is arr.argmin()

1. 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.