We at The Data Monk hold the vision to make sure everyone in the IT industry has an equal stand to work in an open domain such as analytics. Analytics is one domain where there is no formal under-graduation degree and which is achievable to anyone and everyone in the World.

We are a team of 30+ mentors who have worked in various product-based companies in India and abroad, and we have come up with this idea to provide study materials directed to help you crack any analytics interview.

Every one of us has been interviewing for at least the last 6 to 8 years for different positions like Data Scientist, Data Analysts, Business Analysts, Product Analysts, Data Engineers, and other senior roles. We understand the gap between having good knowledge and converting an interview to a top product-based company.

Rest assured that if you follow our different mediums like our blog cum questions-answer portal www.TheDataMonk.com , our youtube channel - The Data Monk, and our e-books, then you will have a very strong candidature in whichever interview you participate in.

There are many blogs that provide free study materials or questions on different analytical tools and technologies, but we concentrate mostly on the questions which are asked in an interview. We have a set of 100+ books which are available both on Amazon and on The Data Monk e-shop page

We would recommend you to explore our website, youtube channel, and e-books to understand the type of questions covered in our articles. We went for the question-answer approach both on our website as well as our e-books just because we feel that the best way to go from beginner to advance level is by practicing a lot of questions on the topic.

We have launched a series of 50 e-books on our website on all the popular as well as niche topics. Our range of material ranges from SQL, Python, and Machine Learning algorithms to ANN, CNN, PCA, etc.

We are constantly working on our product and will keep on updating it. It is very necessary to go through all the questions present in this book.

Give a rating to the book on Amazon, do provide your feedback and if you want to help us grow then please subscribe to our Youtube channel.

Python Pandas Interview Questions

Q1. What is Pandas ?

A1. Pandas is a powerful, flexible, open source and easy to use data analysis and manipulation tool. It aims to be the fundamental building block for data analysis , data manipulation tasks.

Q2. What is python pandas used for ?

A2. Pandas is a open source library of python programming language. Which is mostly use for Data manipulation and analysis.

Q3. Write Steps to install Pandas on Windows.

A2. These are the following steps :

1. The initial step would be to download Python on windows
2. Run the Python executable installer
3. Install pip on Windows
4. Install Pandas in Python using pip

“pip install pandas”

Q4. What are the key features of pandas library ?

A3. These are various features in pandas library :

- Memory Efficient
- Reshaping
- Merge and join
- Time Series
- Data Alignment

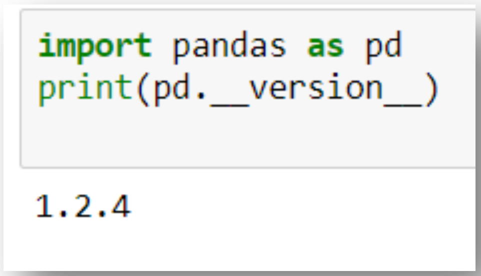
Q5. What is pandas dataframe ?

A5. Pandas dataframe is a 2- dimensional heterogeneous data structure with labeled axes (rows and columns). Pandas dataframe consists of three principle components Data, rows and columns.

Q6. How to Import Pandas Library and also check the version of Library.

A6.

```
# Load the Pandas library with alias pd

import pandas as pd

print(pd.__version__)
```



```
import pandas as pd
print(pd.__version__)
```

```
1.2.4
```

Output : 1.2.4

Q7. How to read the different – different format files using pandas??

A7.

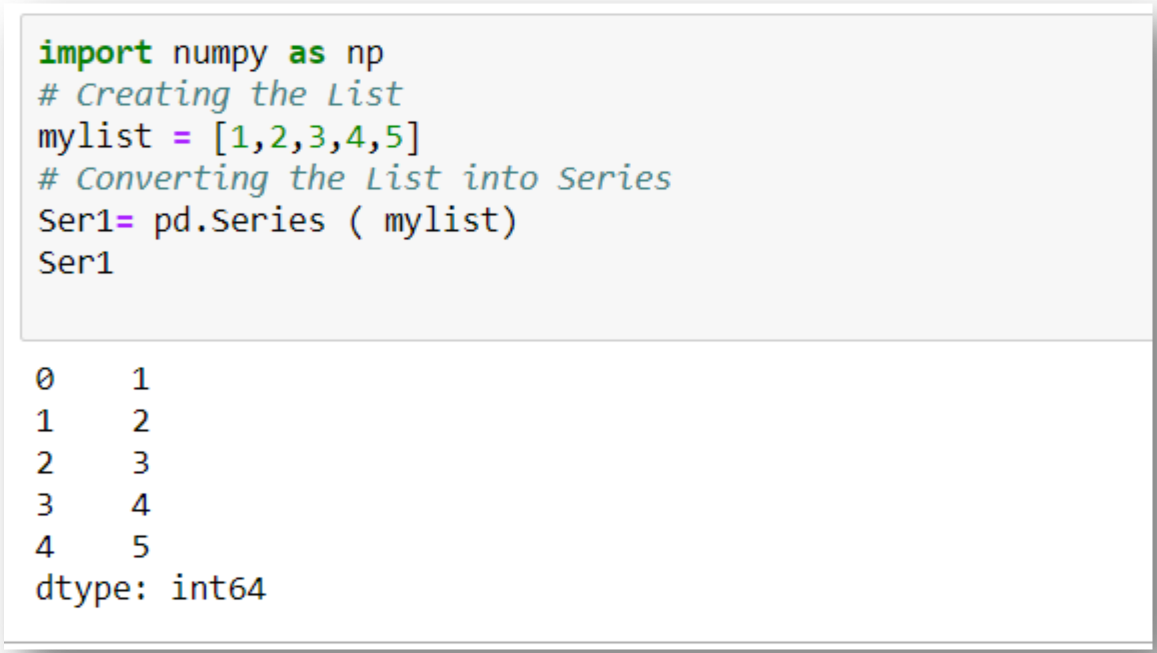
```
# Reading the Comma Separated file
 pd.read_csv( 'filename.csv')

# Reading the tab separated file
pd.read_table ('filename.tsv')

# Reading the Excel File using Pandas
```

```
Pd.read_excel( ‘filename.xlsx’)

# Reading the Html file using pandas
Pd.read_html(‘filename.html’)
```

Q8. How to create a Series from a numpy array, list and dictionary?

A8. Series : Pandas Series is nothing but a Single Column of the Excel Sheet or we can say that Series is a 1D array capable of holding the data of any type(str, int, float etc)

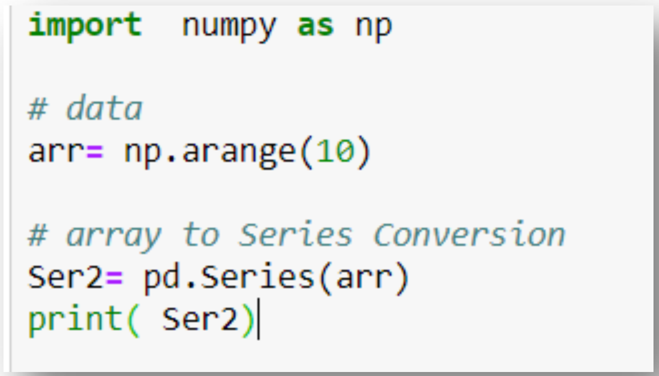
```
#Load the numpy library with alias np
import numpy as np

# Creating the List
mylist = [1,2,3,4,5]

# Converting the List into Series
Ser1= pd.Series ( Mylist)
print ( Ser1 )
```



```
import numpy as np
# Creating the List
mylist = [1,2,3,4,5]
# Converting the List into Series
Ser1= pd.Series ( mylist)
Ser1
```

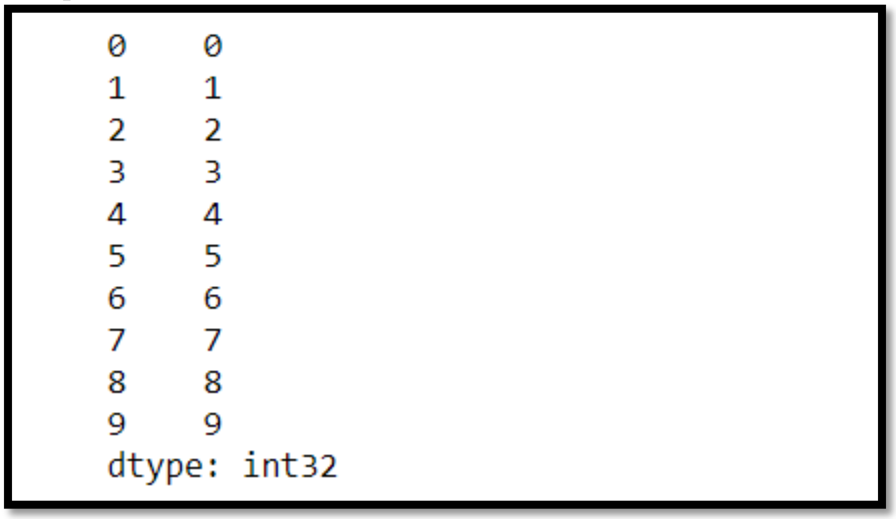
```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

```
# Series Creation using Numpy array

# importing numpy library
import  numpy as np

# data
arr= np.arange(10)

# array to Series Conversion
Ser2= pd.Series(arr)
print( Ser2)
```



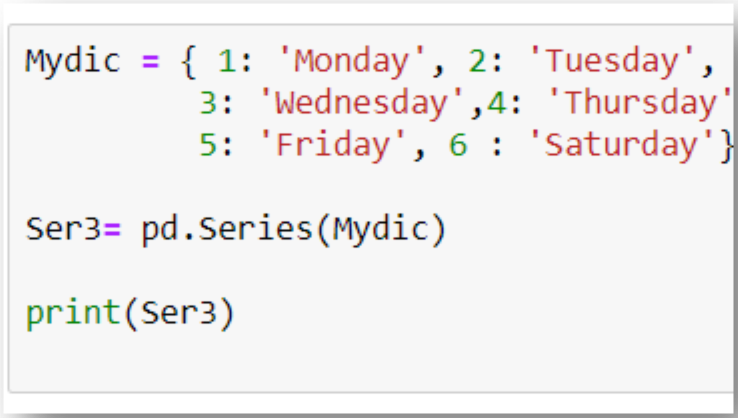
```
import  numpy as np

# data
arr= np.arange(10)

# array to Series Conversion
Ser2= pd.Series(arr)
print( Ser2)
```

Output :



```
0    0
1    1
2    2
3    3
4    4
5    5
6    6
7    7
8    8
9    9
dtype: int32
```

```
#  Series using Dictionary

Mydic = { 1: 'Monday', 2: 'Tuesday',3: 'Wednesday', 4: 'Thursday', 5: 'Friday', 6
: 'Saturday'}

Ser3= pd.Series(Mydic)

print(Ser3)
```



```
Mydic = { 1: 'Monday', 2: 'Tuesday',
         3: 'Wednesday',4: 'Thursday'
         5: 'Friday', 6 : 'Saturday'}

Ser3= pd.Series(Mydic)

print(Ser3)
```

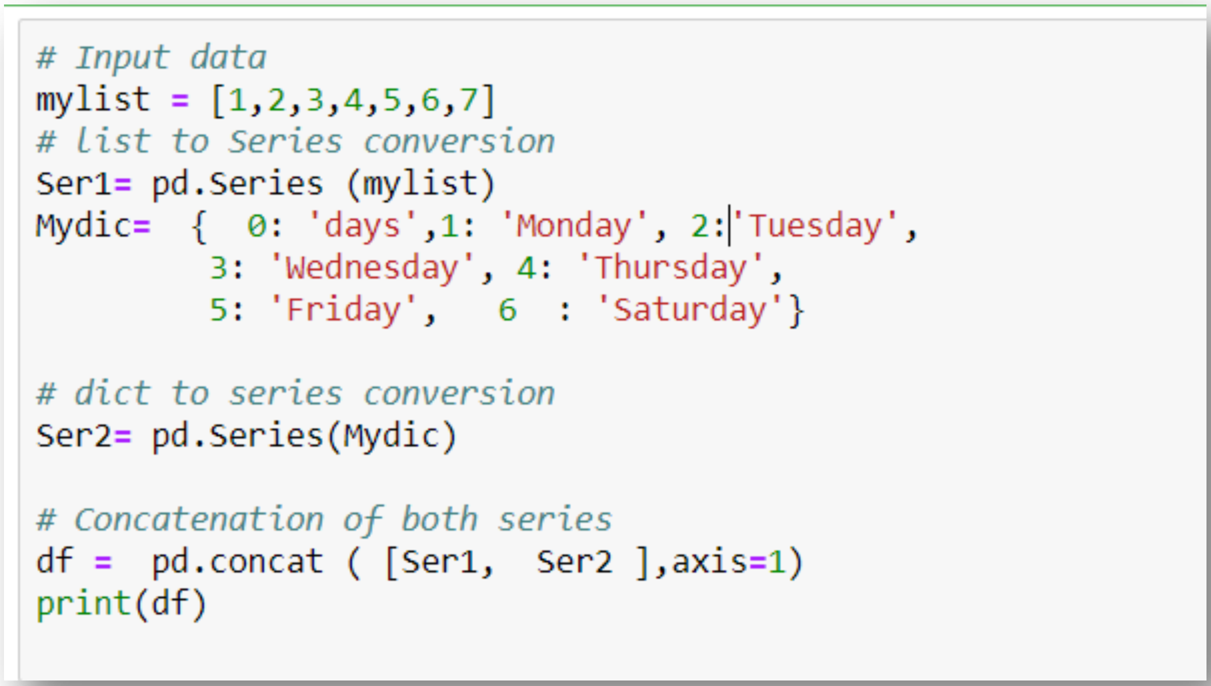
Output :

```
1       monday
2      Tuesday
3    Wednesday
4     Thursday
5       Friday
6     Saturday
dtype: object
```

Q 9. How to make DataFrame using 2 Series?

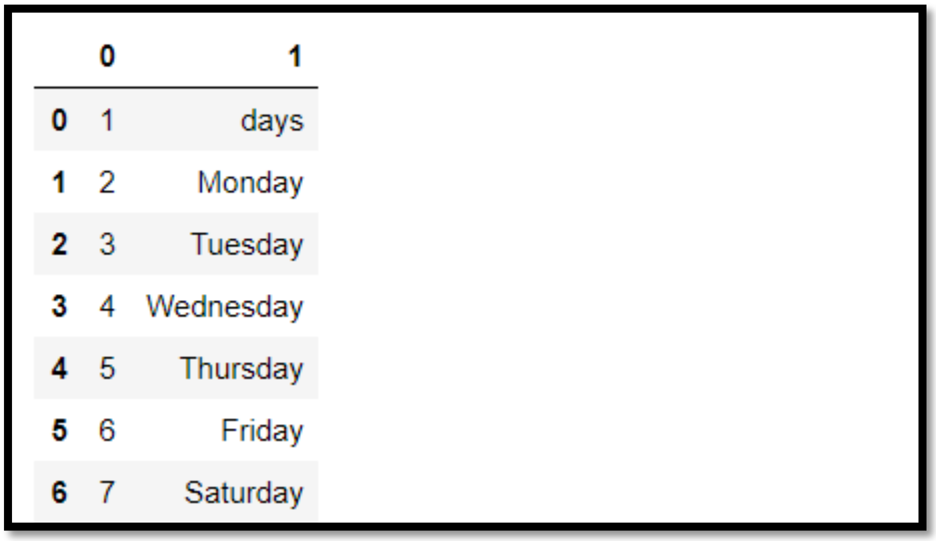
A9.

```
# Input data
mylist = [1,2,3,4,5,6,7]

# list to Series conversion
Ser1= pd.Series (mylist)

Mydic= {  0: ‘days’,1: 'Monday', 2: 'Tuesday',3: 'Wednesday', 4: 'Thursday', 5:
'Friday',   6  : 'Saturday'}

# dict to series conversion
Ser2= pd.Series(Mydic)

# Concatenation of both series
df =  pd.concat ( [Ser1,  Ser2 ],axis=1)
print(df)
```



```
# Input data
mylist = [1,2,3,4,5,6,7]
# list to Series conversion
Ser1= pd.Series (mylist)
Mydic=  {  0: 'days',1: 'Monday', 2:'Tuesday',
         3: 'Wednesday', 4: 'Thursday',
         5: 'Friday',   6  : 'Saturday'}

# dict to series conversion
Ser2= pd.Series(Mydic)

# Concatenation of both series
df =  pd.concat ( [Ser1,  Ser2 ],axis=1)
print(df)
```

Output :



	0	1
0	1	days
1	2	Monday
2	3	Tuesday
3	4	Wednesday
4	5	Thursday
5	6	Friday
6	7	Saturday

Q 10. What is the name of pandas library tools used to create a scatter plot matrix?

A10 . Scatter_matrix is used to create a scatter_plot matrix.

Q11 . What is the describe() method in pandas ?

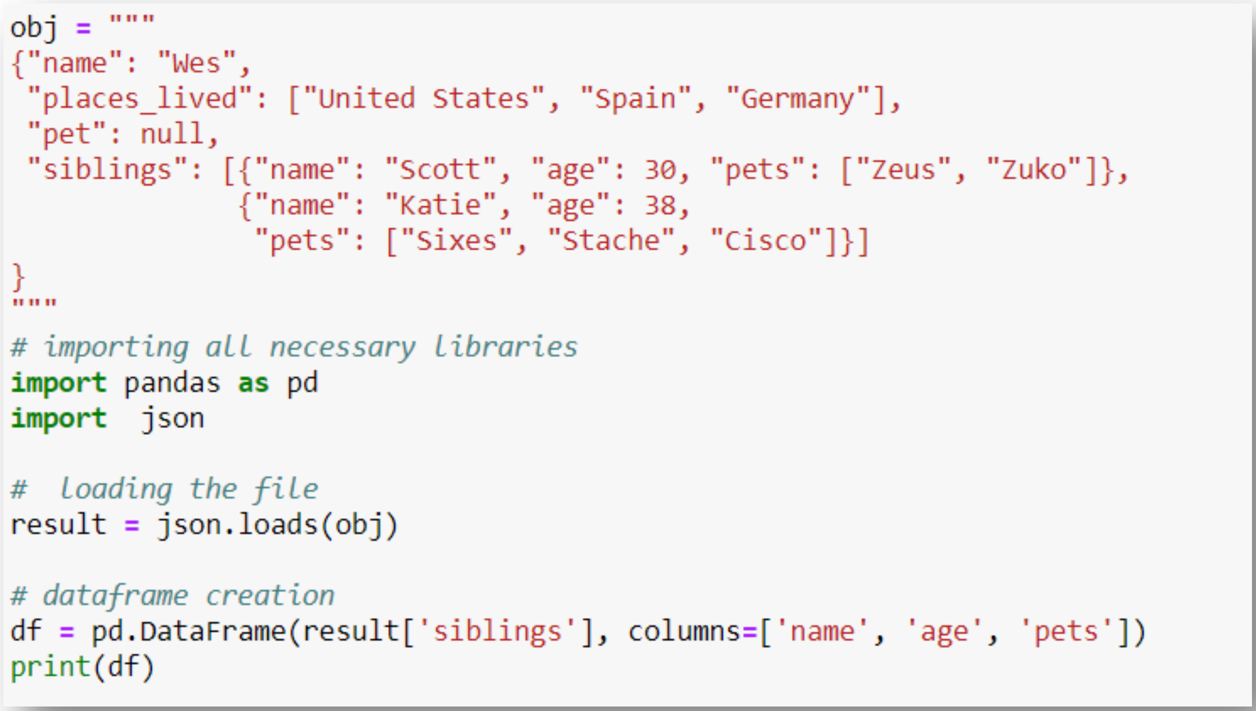
A11. The describe method is used for calculating mean, min, max and standard deviation of each column of the dataset. It analyzes both numeric and object series.

```
Syntax :  dataframe.describe()
```

Q 12. How to Covert Json(Javascript object notation) data into Dataframe ?

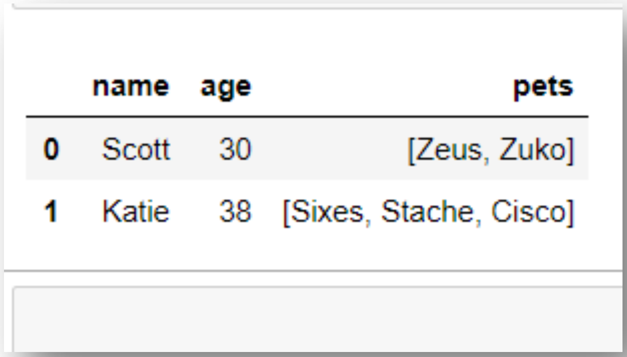
A12. Json is widely used data format for data interchange on the web. We can convert json files into dataframe by using pandas library.

```
#Let Suppose this is our Json data
obj = """
{"name": "Wes",
 "places_lived": ["United States", "Spain", "Germany"],
 "pet": null,
 "siblings": [{"name": "Scott", "age": 30, "pets": ["Zeus", "Zuko"]},
             {"name": "Katie", "age": 38,
              "pets": ["Sixes", "Stache", "Cisco"]}]
}
"""
# importing all necessary libraries
import pandas as pd
import  json

#  loading the file
result = json.loads(obj)

# dataframe creation
df = pd.DataFrame(result['siblings'], columns=['name', 'age', 'pets'])
print(df)
```



```
obj = """
{"name": "Wes",
 "places_lived": ["United States", "Spain", "Germany"],
 "pet": null,
 "siblings": [{"name": "Scott", "age": 30, "pets": ["Zeus", "Zuko"]},
              {"name": "Katie", "age": 38,
               "pets": ["Sixes", "Stache", "Cisco"]}]
}
"""
# importing all necessary libraries
import pandas as pd
import  json

#  loading the file
result = json.loads(obj)

# dataframe creation
df = pd.DataFrame(result['siblings'], columns=['name', 'age', 'pets'])
print(df)
```

Output :



```
    name  age                   pets
0  Scott   30           [Zeus, Zuko]
1  Katie   38  [Sixes, Stache, Cisco]
```

Q13. What is pylab?

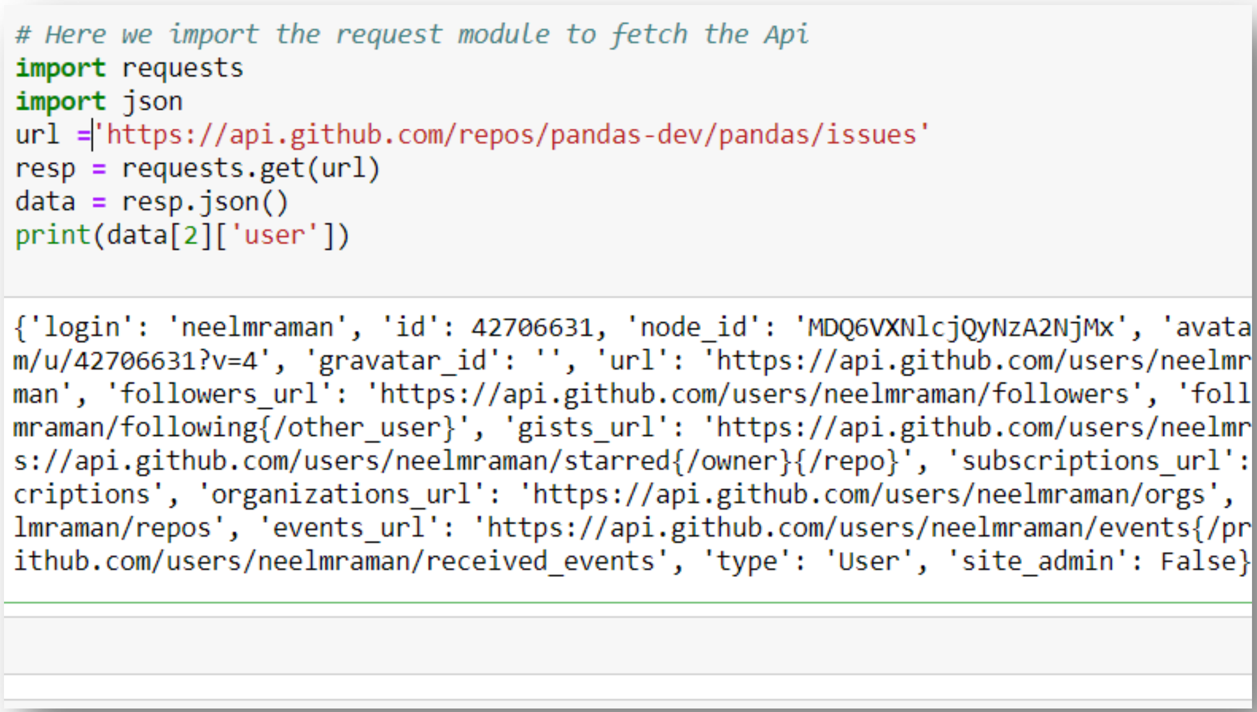
A13. PyLab is a package that contains Numpy, Scipy and Matplotlib into a single namespace.

Q14. How to convert Api into Dataframe?

A14. We can create Api into dataframe:

Code:

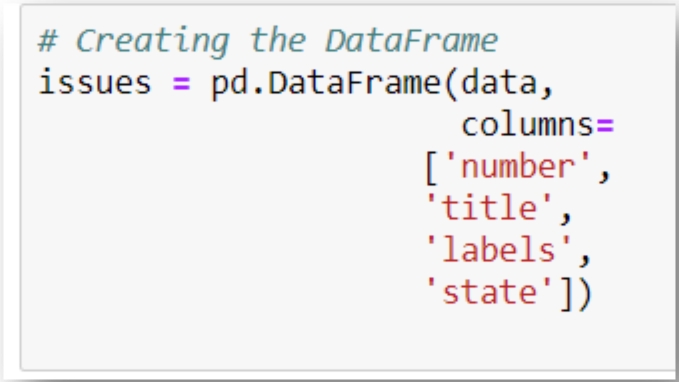
```
# Here we import the request module to fetch the Api
import requests
import json

# url for the dataframe
url = 'https://api.github.com/repos/pandas-dev/pandas/issues'
resp = requests.get(url)
data = resp.json()
print(data[2]['user'])

# Creating the DataFrame
issues = pd.DataFrame(data, columns=['number', 'title',
                        'labels', 'state'])
```



```
# Here we import the request module to fetch the Api
import requests
import json
url ='https://api.github.com/repos/pandas-dev/pandas/issues'
resp = requests.get(url)
data = resp.json()
print(data[2]['user'])
```

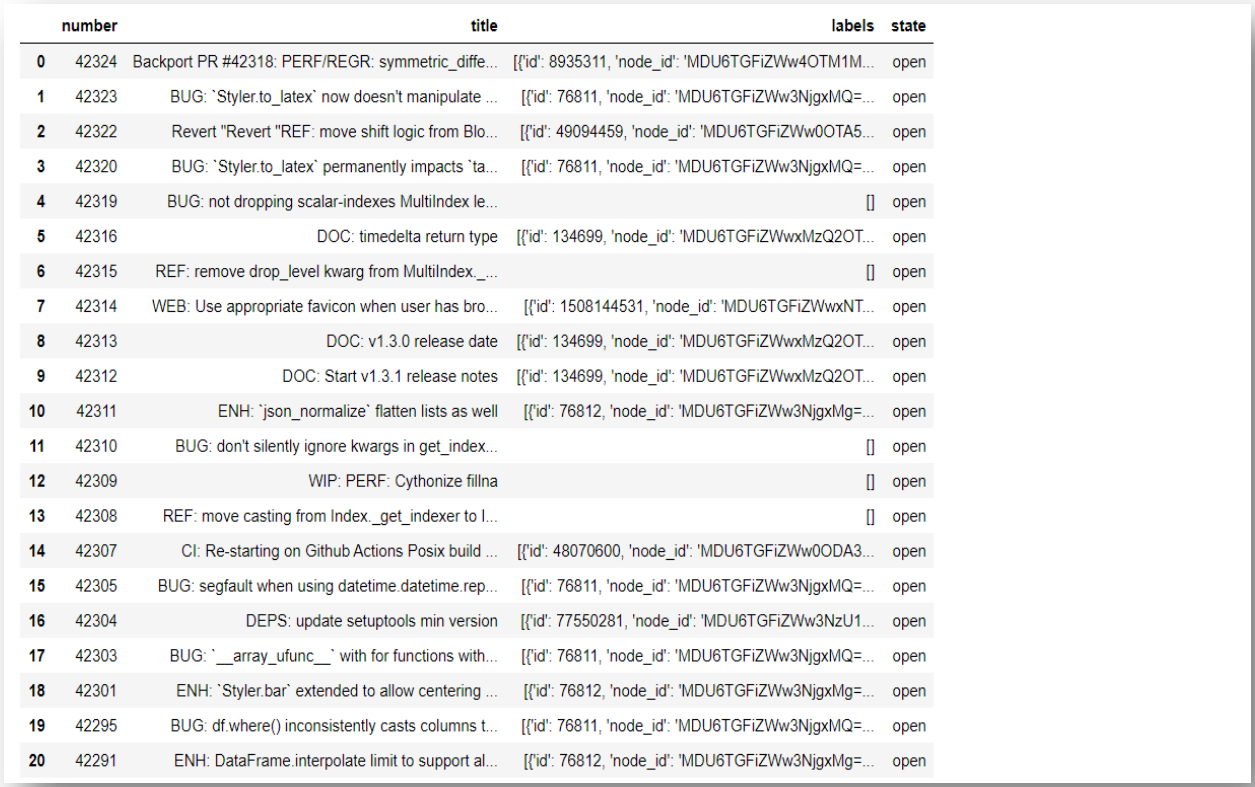
```
{'login': 'neelmraman', 'id': 42706631, 'node_id': 'MDQ6VXNlcjQyNzA2NjMx', 'avata
m/u/42706631?v=4', 'gravatar_id': '', 'url': 'https://api.github.com/users/neelmr
man', 'followers_url': 'https://api.github.com/users/neelmraman/followers', 'foll
mraman/following{/other_user}', 'gists_url': 'https://api.github.com/users/neelmr
s://api.github.com/users/neelmraman/starred{/owner}{/repo}', 'subscriptions_url':
criptions', 'organizations_url': 'https://api.github.com/users/neelmraman/orgs',
lmraman/repos', 'events_url': 'https://api.github.com/users/neelmraman/events{/pr
ithub.com/users/neelmraman/received_events', 'type': 'User', 'site_admin': False}
```

# Creating the dataframe of the output



```python
# Creating the DataFrame
issues = pd.DataFrame(data,
                      columns=
                    ['number',
                    'title',
                    'labels',
                    'state'])
```

Output :

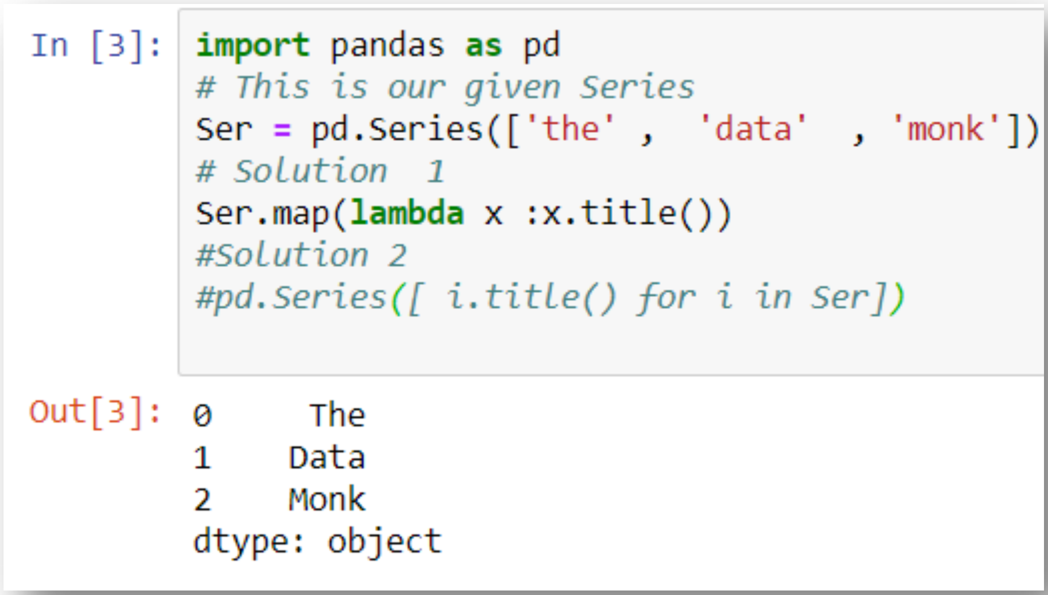


	number	title	labels	state
0	42324	Backport PR #42318: PERF/REGR: symmetric_diffe...	[{'id': 8935311, 'node_id': 'MDU6TGFiZWw4OTM1M...	open
1	42323	BUG: `Styler.to_latex` now doesn't manipulate ...	[{'id': 76811, 'node_id': 'MDU6TGFiZWw3NjgxMQ=...	open
2	42322	Revert "Revert "REF: move shift logic from Blo...	[{'id': 49094459, 'node_id': 'MDU6TGFiZWw0OTA5...	open
3	42320	BUG: `Styler.to_latex` permanently impacts `ta...	[{'id': 76811, 'node_id': 'MDU6TGFiZWw3NjgxMQ=...	open
4	42319	BUG: not dropping scalar-indexes MultiIndex le...	[]	open
5	42316	DOC: timedelta return type	[{'id': 134699, 'node_id': 'MDU6TGFiZWwxMzQ2OT...	open
6	42315	REF: remove drop_level kwarg from MultiIndex._...	[]	open
7	42314	WEB: Use appropriate favicon when user has bro...	[{'id': 1508144531, 'node_id': 'MDU6TGFiZWwxNT...	open
8	42313	DOC: v1.3.0 release date	[{'id': 134699, 'node_id': 'MDU6TGFiZWwxMzQ2OT...	open
9	42312	DOC: Start v1.3.1 release notes	[{'id': 134699, 'node_id': 'MDU6TGFiZWwxMzQ2OT...	open
10	42311	ENH: `json_normalize` flatten lists as well	[{'id': 76812, 'node_id': 'MDU6TGFiZWw3NjgxMg=...	open
11	42310	BUG: don't silently ignore kwargs in get_index...	[]	open
12	42309	WIP: PERF: Cythonize fillna	[]	open
13	42308	REF: move casting from Index._get_indexer to I...	[]	open
14	42307	CI: Re-starting on Github Actions Posix build ...	[{'id': 48070600, 'node_id': 'MDU6TGFiZWw0ODA3...	open
15	42305	BUG: segfault when using datetime.datetime.rep...	[{'id': 76811, 'node_id': 'MDU6TGFiZWw3NjgxMQ=...	open
16	42304	DEPS: update setuptools min version	[{'id': 77550281, 'node_id': 'MDU6TGFiZWw3NzU1...	open
17	42303	BUG: `__array_ufunc__` with for functions with...	[{'id': 76811, 'node_id': 'MDU6TGFiZWw3NjgxMQ=...	open
18	42301	ENH: `Styler.bar` extended to allow centering ...	[{'id': 76812, 'node_id': 'MDU6TGFiZWw3NjgxMg=...	open
19	42295	BUG: df.where() inconsistently casts columns t...	[{'id': 76811, 'node_id': 'MDU6TGFiZWw3NjgxMQ=...	open
20	42291	ENH: DataFrame.interpolate limit to support al...	[{'id': 76812, 'node_id': 'MDU6TGFiZWw3NjgxMg=...	open

Q15. How to Convert the First Character of Each element in a series to uppercase?

A15.

```
# This is our given Series

Ser = pd.Series([‘the ‘ ,  ‘data’  , ‘monk’])

# Solution  1

Ser.map(lambda x :x.title())

#Solution 2

pd.Series([ i.title() for i in Ser])
```



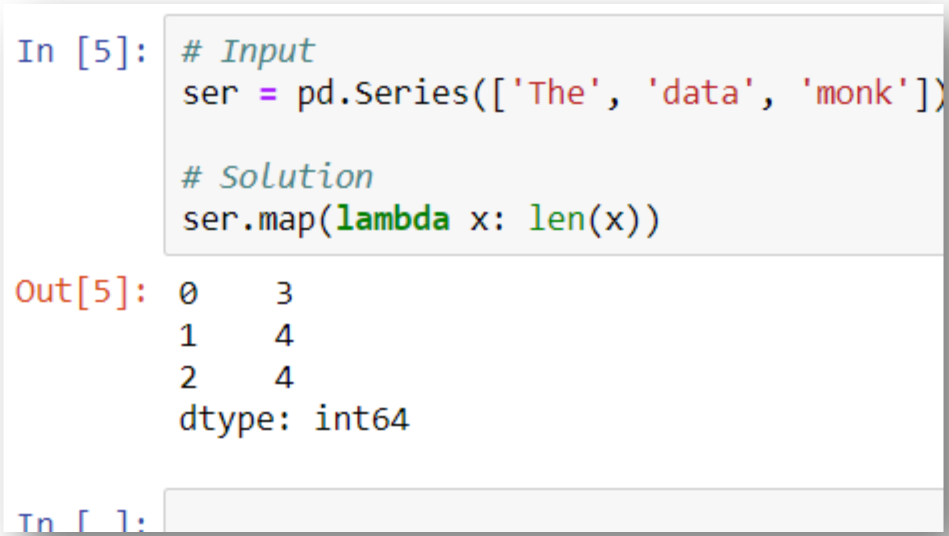
```
In [3]: import pandas as pd
        # This is our given Series
        Ser = pd.Series(['the' ,  'data'  , 'monk'])
        # Solution  1
        Ser.map(lambda x :x.title())
        #Solution 2
        #pd.Series([ i.title() for i in Ser])

Out[3]: 0     The
        1    Data
        2    Monk
        dtype: object
```

Q 16. How to Calculate the number of characters in each word in a series?

A16.

```
# Input

ser = pd.Series(['The', 'data', 'monk'])
```

```
# Solution

ser.map(lambda x: len(x))
```



```
In [5]: # Input
        ser = pd.Series(['The', 'data', 'monk'])

        # Solution
        ser.map(lambda x: len(x))

Out[5]: 0    3
        1    4
        2    4
        dtype: int64

In [ ]:
```

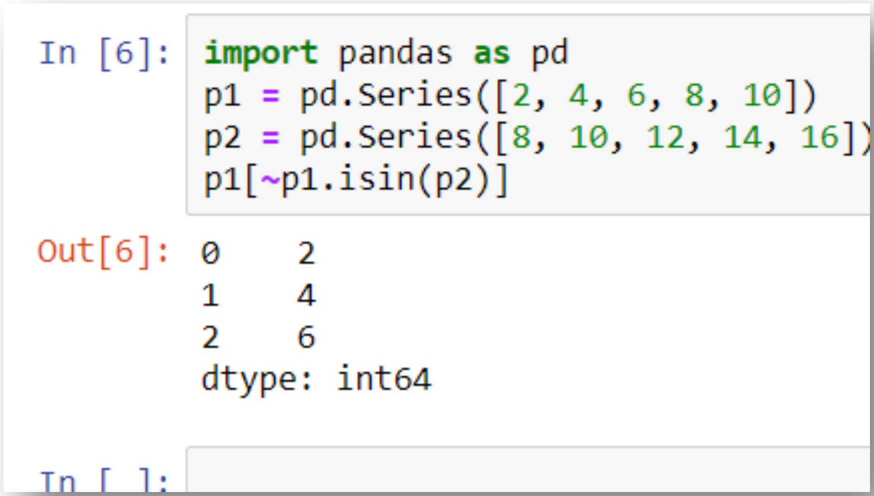
Q17 . how many minimum Number of arguments we require to pass in pandas Series ?

A17. We have to pass min 1 argument in pandas Series.

Q18. How to get the items of Series X not present in Series Y?

A18.

```
import pandas as pd

#Series 1

p1 = pd.Series([2, 4, 6, 8, 10])

# Series 2

p2 = pd.Series([8, 10, 12, 14, 16])

p1[~p1.isin(p2)]
```

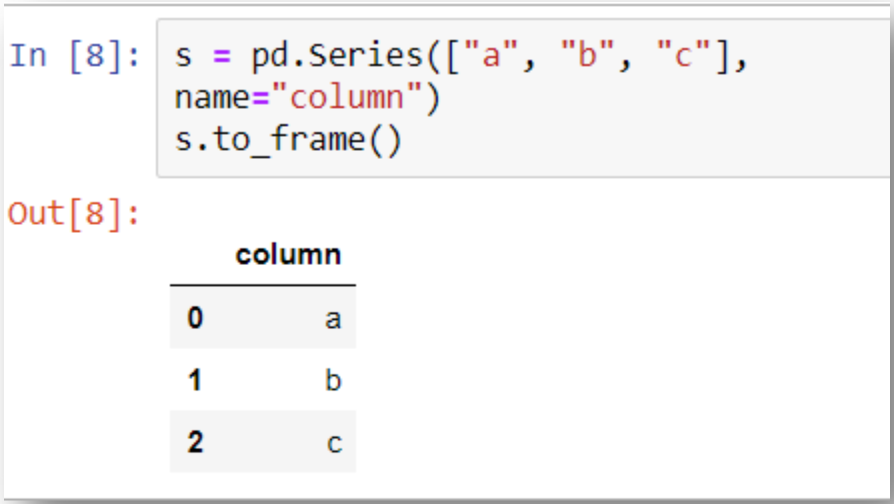


```
In [6]: import pandas as pd
        p1 = pd.Series([2, 4, 6, 8, 10])
        p2 = pd.Series([8, 10, 12, 14, 16])
        p1[~p1.isin(p2)]

Out[6]: 0    2
        1    4
        2    6
        dtype: int64

In [ ]:
```

Q19. How can we convert Series to dataframe ?

A19. We can convert Series into dataframe by using to_frame function.

```
# Input data

s = pd.Series(["a", "b", "c"],

name="column")

# Conversion of Series into dataframe

s.to_frame()
```



```
In [8]: s = pd.Series(["a", "b", "c"],
        name="column")
        s.to_frame()
```

```
Out[8]:
           column
        0       a
        1       b
        2       c
```

Q20 . If data is an ndarray , index must be the same length as data. True or False?

A20. It is always true.

Q21. What is Pandas Index?

A21. Pandas index is defined as a tool that selects particular rows and columns of data from a dataframe. Its task is to organize the data and to provide fast accessing of the data.

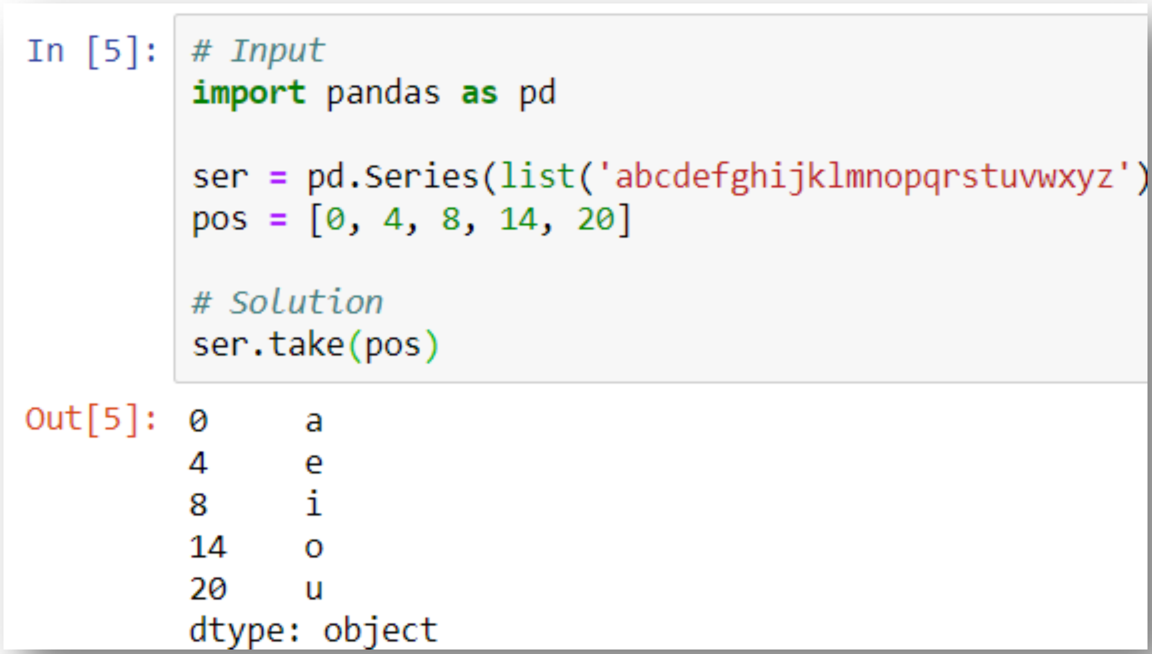
Q 22. What is Multiple Indexing?

A22. Multiple Indexing is very useful because it deals with data analysis and manipulation , especially for working with high dimensional data.

Q23. How to extract items at given positions from a series?
A23.

```
# Input
```

```
import pandas as pd

ser = pd.Series(list('abcdefghijklmnopqrstuvwxyz'))

pos = [0, 4, 8, 14, 20]

# Solution

ser.take(pos)
```



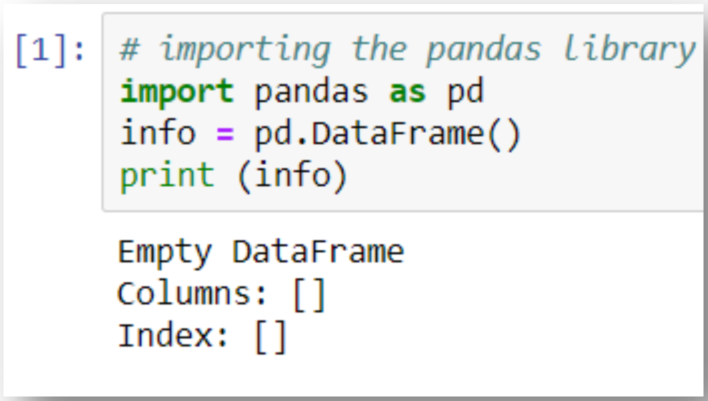
```
In [5]: # Input
        import pandas as pd

        ser = pd.Series(list('abcdefghijklmnopqrstuvwxyz')
        pos = [0, 4, 8, 14, 20]

        # Solution
        ser.take(pos)

Out[5]: 0     a
        4     e
        8     i
        14    o
        20    u
        dtype: object
```

Q24. How will you create an empty dataframe in pandas?

A24. A Dataframe is widely used data structure of pandas and works with 2 D Dimensional array with labelled axes.

```
# importing the pandas library

import pandas as pd

info = pd.DataFrame()
```

```
print(info)
```

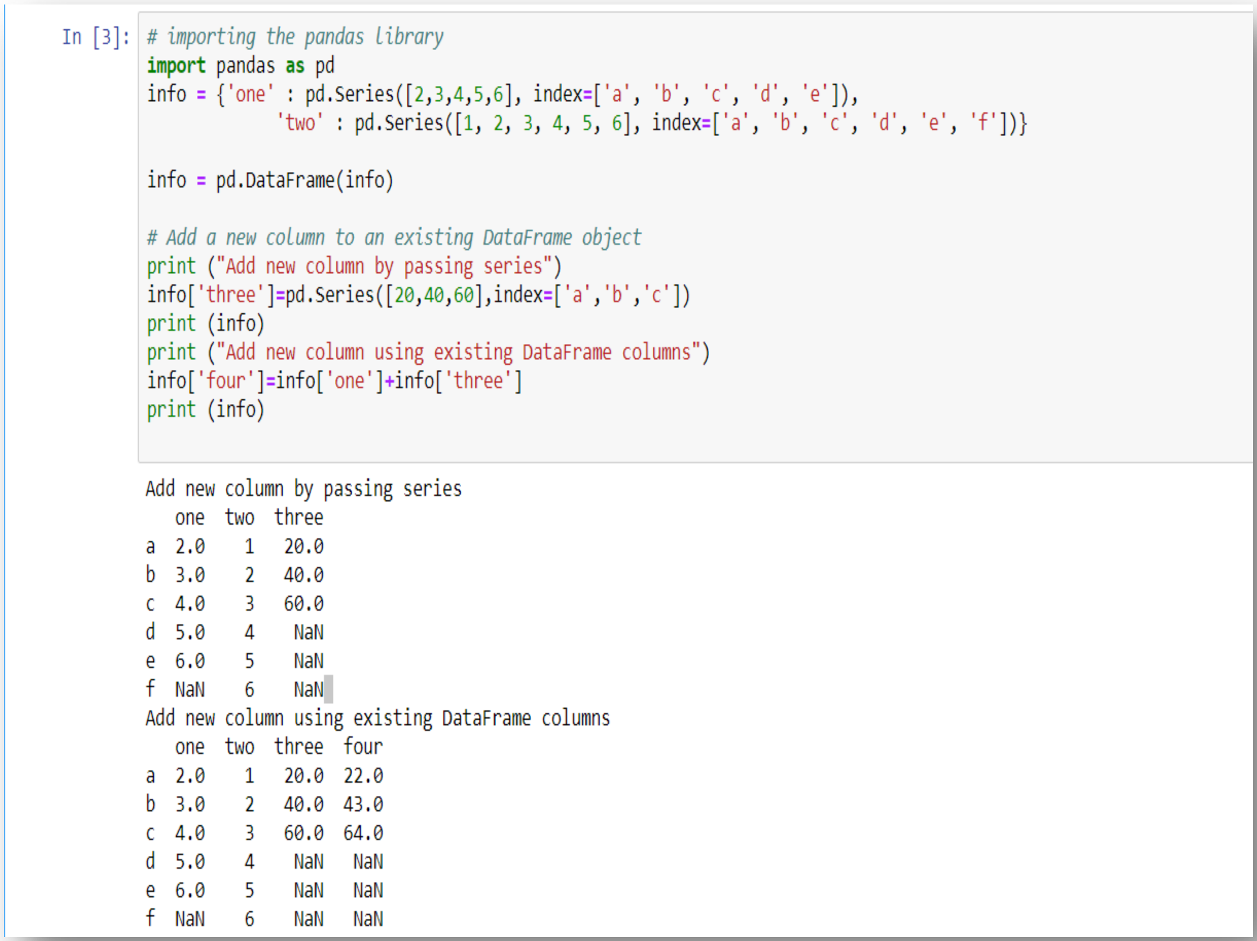


```
[1]: # importing the pandas library
     import pandas as pd
     info = pd.DataFrame()
     print (info)

     Empty DataFrame
     Columns: []
     Index: []
```

Q25. How will you add a new column to the Pandas DataFrame ?

A25. We can add new column to an existing dataframe:

```
# importing the pandas library

import pandas as pd

info = {'one' : pd.Series([2,3,4,5,6], index=['a', 'b', 'c', 'd', 'e']),

          'two' : pd.Series([1, 2, 3, 4, 5, 6], index=['a', 'b', 'c', 'd', 'e', 'f'])}

info = pd.DataFrame(info)

# Add a new column to an existing DataFrame object
print ("Add new column by passing series")
info['three']=pd.Series([20,40,60],index=['a','b','c'])
print (info)
print ("Add new column using existing DataFrame columns")
info['four']=info['one']+info['three']
print (info)
```



```
In [3]: # importing the pandas library
        import pandas as pd
        info = {'one' : pd.Series([2,3,4,5,6], index=['a', 'b', 'c', 'd', 'e']),
                 'two' : pd.Series([1, 2, 3, 4, 5, 6], index=['a', 'b', 'c', 'd', 'e', 'f'])}

        info = pd.DataFrame(info)

        # Add a new column to an existing DataFrame object
        print ("Add new column by passing series")
        info['three']=pd.Series([20,40,60],index=['a','b','c'])
        print (info)
        print ("Add new column using existing DataFrame columns")
        info['four']=info['one']+info['three']
        print (info)
```

```
Add new column by passing series
   one  two  three
a  2.0    1   20.0
b  3.0    2   40.0
c  4.0    3   60.0
d  5.0    4    NaN
e  6.0    5    NaN
f  NaN    6    NaN
Add new column using existing DataFrame columns
   one  two  three  four
a  2.0    1   20.0  22.0
b  3.0    2   40.0  43.0
c  4.0    3   60.0  64.0
d  5.0    4    NaN   NaN
e  6.0    5    NaN   NaN
f  NaN    6    NaN   NaN
```

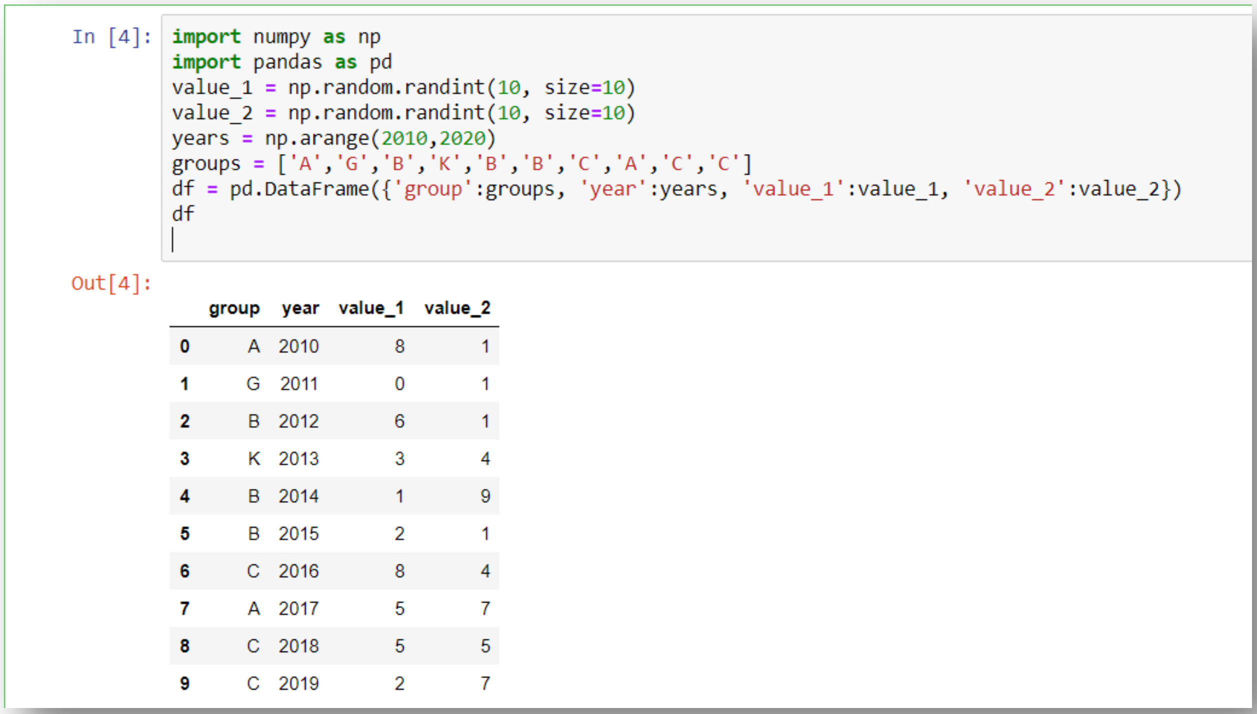
Q26. What is query function in pandas ?

A26. We sometimes need to filter a dataframe based on a condition or apply a mask to get certain values.

Let's First Create a Simple DataFrame :

```
import numpy as np

import pandas as pd

value_1 = np.random.randint(10, size=10)

value_2 = np.random.randint(10, size=10)
```

```
years = np.arange(2010,2020)

groups = ['A','G','B','K','B','B','C','A','C','C']

df = pd.DataFrame({'group':groups, 'year':years, 'value_1':value_1,
'value_2':value_2})

print(df)
```



```
In [4]: import numpy as np
        import pandas as pd
        value_1 = np.random.randint(10, size=10)
        value_2 = np.random.randint(10, size=10)
        years = np.arange(2010,2020)
        groups = ['A','G','B','K','B','B','C','A','C','C']
        df = pd.DataFrame({'group':groups, 'year':years, 'value_1':value_1, 'value_2':value_2})
        df
```

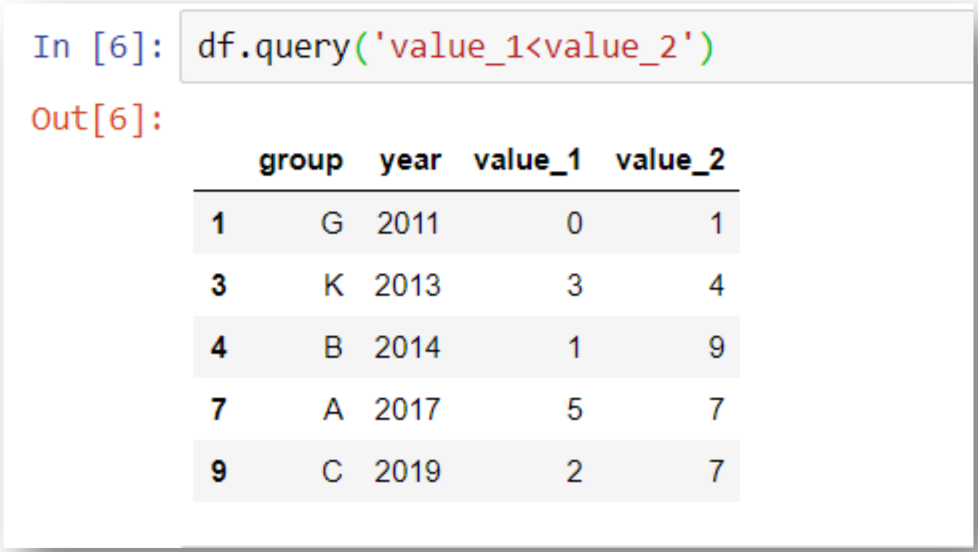
Out[4]:

	group	year	value_1	value_2
0	A	2010	8	1
1	G	2011	0	1
2	B	2012	6	1
3	K	2013	3	4
4	B	2014	1	9
5	B	2015	2	1
6	C	2016	8	4
7	A	2017	5	7
8	C	2018	5	5
9	C	2019	2	7

It is very simple to use query function. It is only require to write condition inside a query function.

```
df.query('value_1<value_2')
```

Output :



```
In [6]: df.query('value_1<value_2')
Out[6]:
```

	group	year	value_1	value_2
1	G	2011	0	1
3	K	2013	3	4
4	B	2014	1	9
7	A	2017	5	7
9	C	2019	2	7

Q27. How will you delete rows from a pandas dataframe ?

A27. For Deleting the rows from a dataframe we can use drop() method of pandas library.

Code :

```
#Import modules
import pandas as pd
#Create a dataframe
data = {'name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],
      'year': [2012, 2012, 2013, 2014, 2014],
      'reports': [4, 24, 31, 2, 3]}
df = pd.DataFrame(data, index = ['Cochice', 'Pima', 'Santa Cruz', 'Maricopa',
'Yuma'])

# Delete a row
df.drop(['Cochice', 'Pima'])
```

Q 28. How will you get the number of rows and columns of a Dataframe in pandas?

A28. We can use shape() method for find the number of rows and columns in a dataframe.

Code :

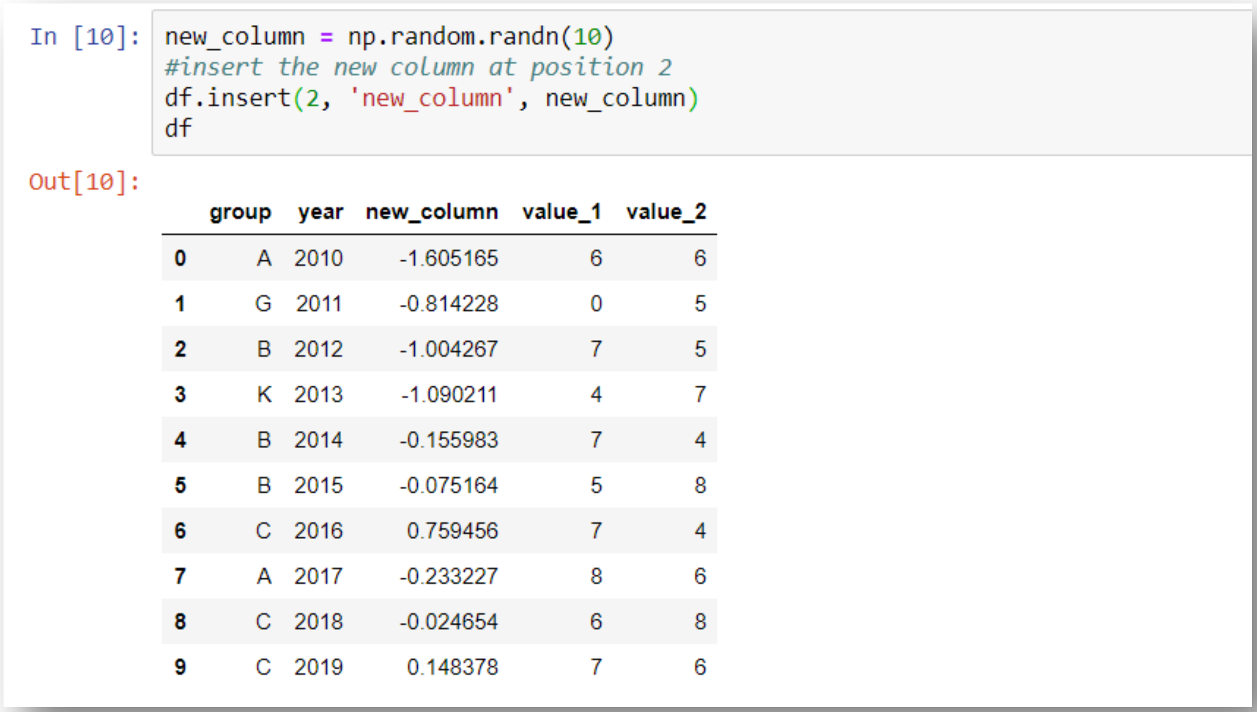
```
import pandas as pd
import numpy as np

raw_data = {'name': ['Willard Morris', 'Al Jennings', 'Omar Mullins', 'Spencer
McDaniel'],
'age': [20, 19, 22, 21],
'favorite_color': ['blue', 'red', 'yellow', "green"],
'grade': [88, 92, 95, 70]}
df = pd.DataFrame(raw_data, columns = ['name', 'age', 'favorite_color',
'grade'])
df

# get the row and column count of the df
df.shape()
```

Q29. Why do we use the insert function in pandas ?

A29. As we know whenever we want to add a column to the dataframe , it is added to the last by default. But Pandas provides us the option that we can add column at any position by using Insert Function.

We need to specify the position wherever we want to insert. Let's suppose we want to insert the column at 2nd Position.

```
new_column = np.random.randn(10)

#insert the new column at position 2

df.insert(2, 'new_column', new_column)

print(df)
```



```
In [10]: new_column = np.random.randn(10)
         #insert the new column at position 2
         df.insert(2, 'new_column', new_column)
         df
```

Out[10]:

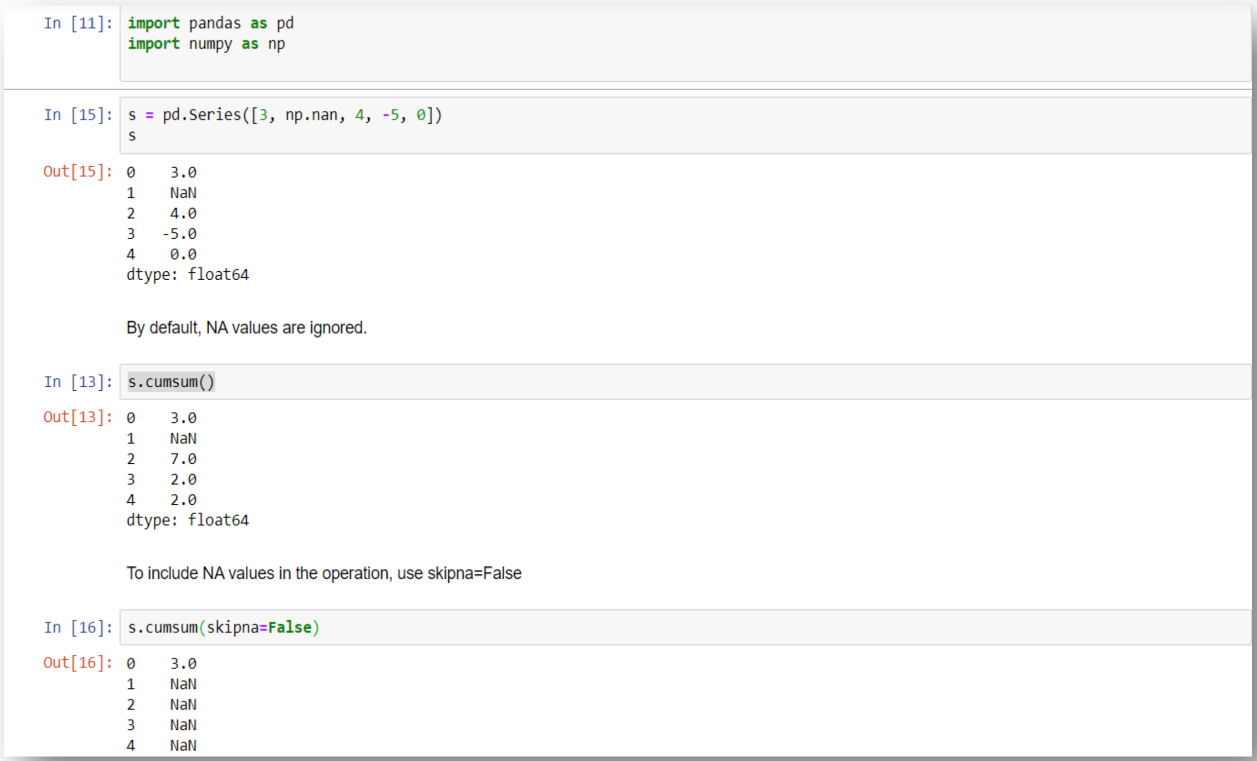
	group	year	new_column	value_1	value_2
0	A	2010	-1.605165	6	6
1	G	2011	-0.814228	0	5
2	B	2012	-1.004267	7	5
3	K	2013	-1.090211	4	7
4	B	2014	-0.155983	7	4
5	B	2015	-0.075164	5	8
6	C	2016	0.759456	7	4
7	A	2017	-0.233227	8	6
8	C	2018	-0.024654	6	8
9	C	2019	0.148378	7	6

Q30. What is Cumsum() Function in Pandas and why do we use it ?

A30. The Cumsum() Function is used to get Cumulative Sum over the dataframe.

Let's understand with the practical example.

```
import pandas as pd

import numpy as np

s = pd.Series([3, np.nan, 4, -5, 0])

s.cumsum()

s.cumsum(skipna=False)
```



```
In [11]: import pandas as pd
         import numpy as np

In [15]: s = pd.Series([3, np.nan, 4, -5, 0])
         s

Out[15]: 0    3.0
         1    NaN
         2    4.0
         3   -5.0
         4    0.0
         dtype: float64
```

By default, NA values are ignored.

```
In [13]: s.cumsum()

Out[13]: 0    3.0
         1    NaN
         2    7.0
         3    2.0
         4    2.0
         dtype: float64
```

To include NA values in the operation, use skipna=False

```
In [16]: s.cumsum(skipna=False)

Out[16]: 0    3.0
         1    NaN
         2    NaN
         3    NaN
         4    NaN
```

Q31 What is Pandas ml?

A31. Pandas_ml is a package which integrates pandas , scikit-learn , xgboost into one package for easy handling of data and creation of machine learning models.

Installation:

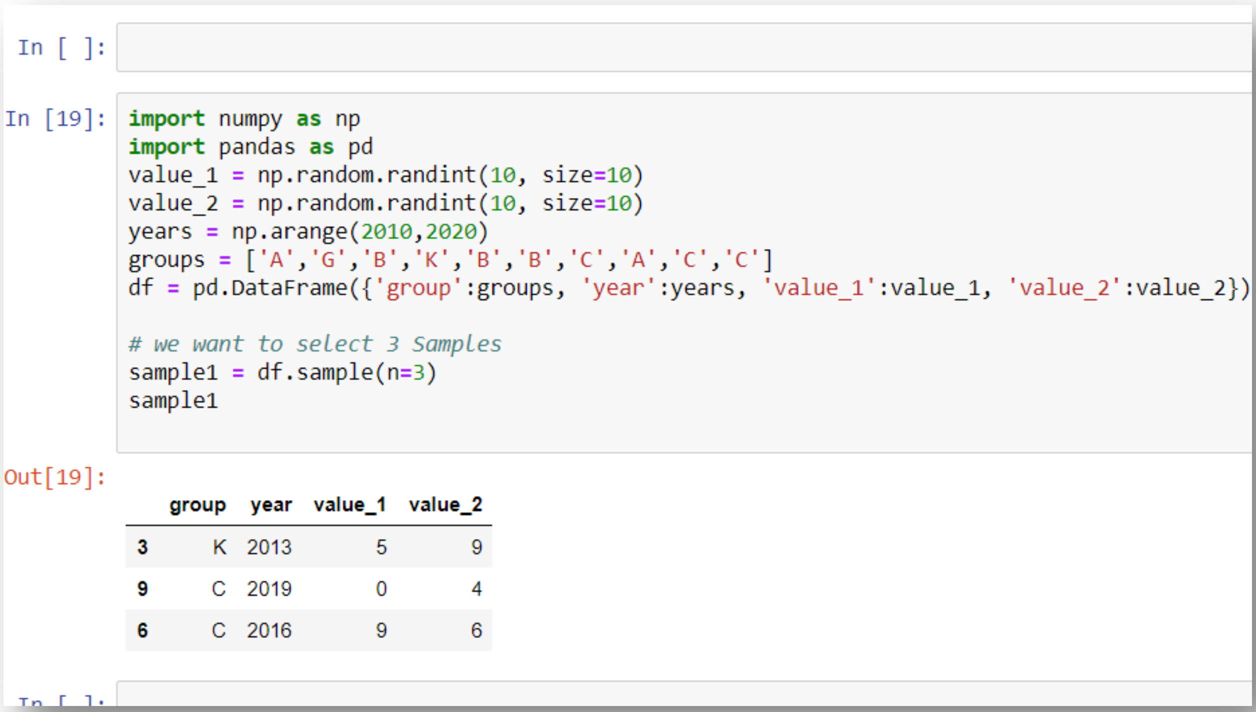
```
pip install pandas-ml
```

Q32. What is Sample Method in Pandas?

A32. Sample Method is very useful when we want to select a random sample from a distribution. Sample Method allows you to Select random number of Samples from the Series or DataFrame.

Let 's suppose this is our Dataframe.

```
import numpy as np

import pandas as pd

value_1 = np.random.randint(10, size=10)

value_2 = np.random.randint(10, size=10)

years = np.arange(2010,2020)

groups = ['A','G','B','K','B','B','C','A','C','C']

df = pd.DataFrame({'group':groups, 'year':years, 'value_1':value_1,
'value_2':value_2})

print(df)

sample1 = df.sample(n=3)
sample1
```



```
In [ ]:

In [19]: import numpy as np
         import pandas as pd
         value_1 = np.random.randint(10, size=10)
         value_2 = np.random.randint(10, size=10)
         years = np.arange(2010,2020)
         groups = ['A','G','B','K','B','B','C','A','C','C']
         df = pd.DataFrame({'group':groups, 'year':years, 'value_1':value_1, 'value_2':value_2})

         # we want to select 3 Samples
         sample1 = df.sample(n=3)
         sample1

Out[19]:
```

	group	year	value_1	value_2
3	K	2013	5	9
9	C	2019	0	4
6	C	2016	9	6

Q33. What is loc and iloc function in Pandas?

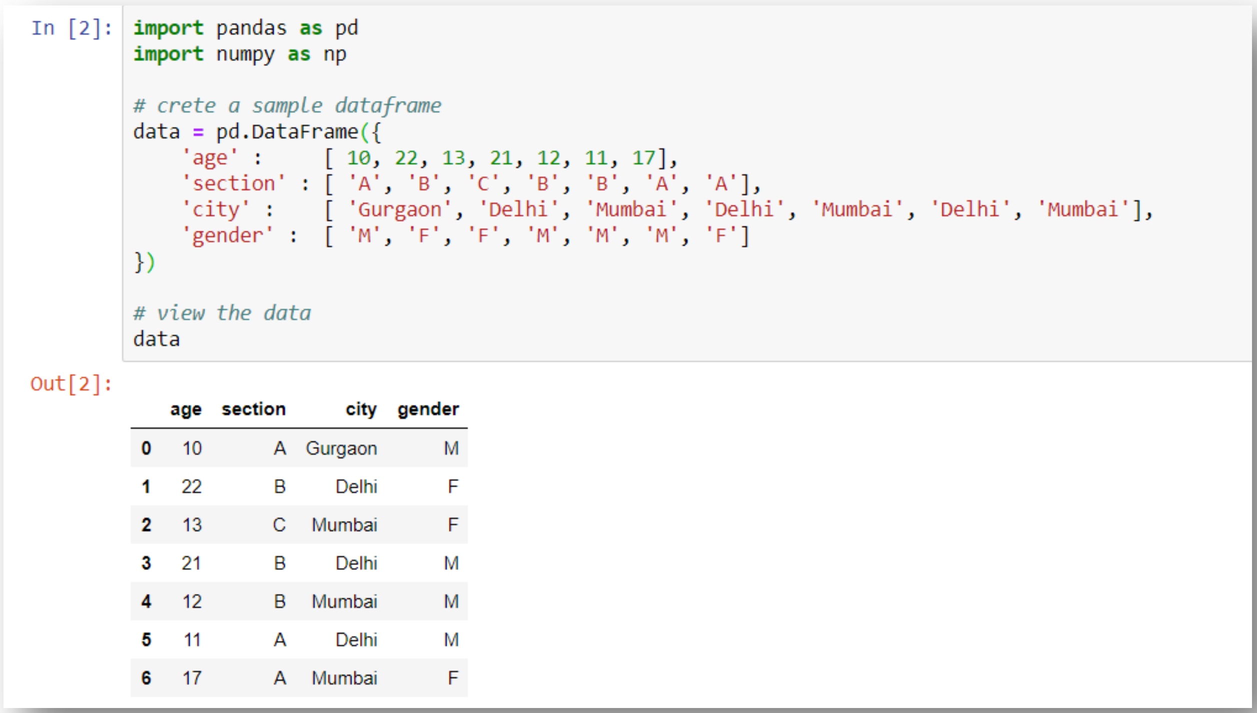
A33.

loc : loc is label-based , which means that we have to specify the name of rows and columns that we want to filter out.

iloc : iloc is interger, index-based, we have to specify the rows and columns by their interger index.

Let suppose this is our dataframe:

```
import pandas as pd

import numpy as np

# create a sample dataframe

data = pd.DataFrame({

   'age' :     [ 10, 22, 13, 21, 12, 11, 17],

   'section' : [ 'A', 'B', 'C', 'B', 'B', 'A', 'A'],

   'city' :    [ 'Gurgaon', 'Delhi', 'Mumbai', 'Delhi', 'Mumbai', 'Delhi', 'Mumbai'],

   'gender' :  [ 'M', 'F', 'F', 'M', 'M', 'M', 'F']

})

# view the data

data
```



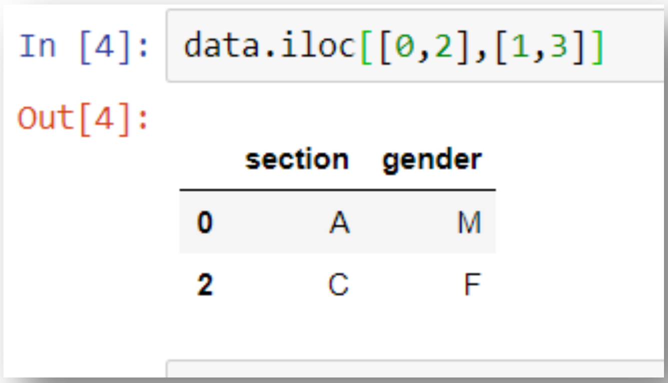
```
In [2]: import pandas as pd
        import numpy as np

        # crete a sample dataframe
        data = pd.DataFrame({
            'age' :     [ 10, 22, 13, 21, 12, 11, 17],
            'section' : [ 'A', 'B', 'C', 'B', 'B', 'A', 'A'],
            'city' :    [ 'Gurgaon', 'Delhi', 'Mumbai', 'Delhi', 'Mumbai', 'Delhi', 'Mumbai'],
            'gender' :  [ 'M', 'F', 'F', 'M', 'M', 'M', 'F']
        })

        # view the data
        data

Out[2]:
```

	age	section	city	gender
0	10	A	Gurgaon	M
1	22	B	Delhi	F
2	13	C	Mumbai	F
3	21	B	Delhi	M
4	12	B	Mumbai	M
5	11	A	Delhi	M
6	17	A	Mumbai	F

iloc Function:

# Select rows with particular indices and particular columns

data.iloc[[0,2],[1,3]]

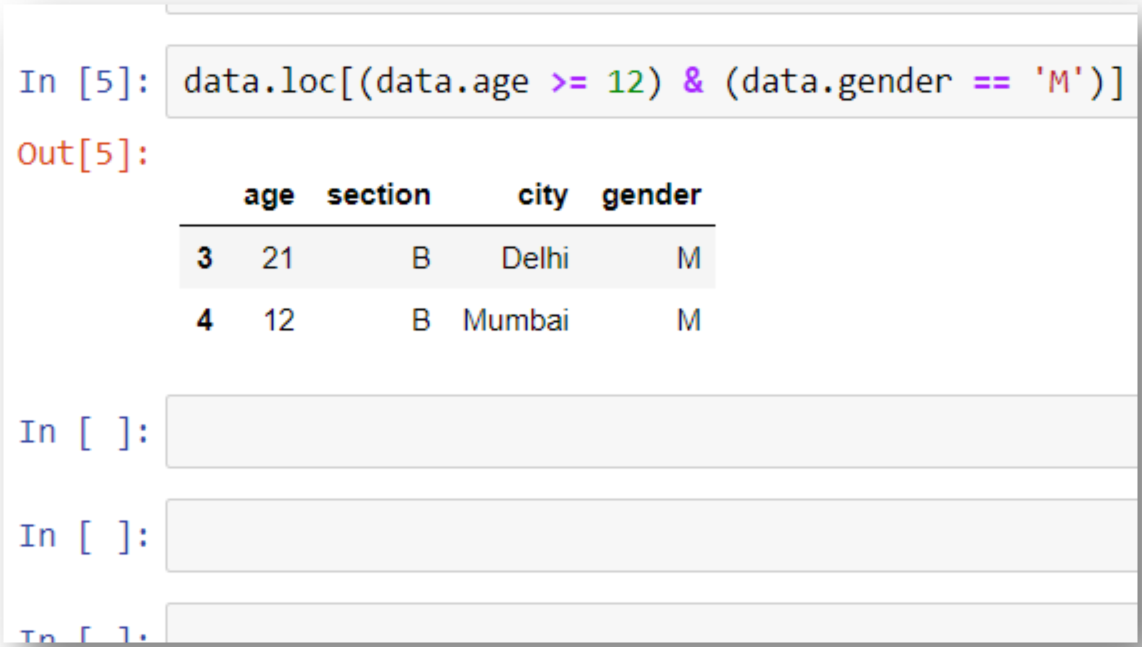


```
In [4]: data.iloc[[0,2],[1,3]]

Out[4]:
```

	section	gender
0	A	M
2	C	F

loc Function :

```
# Select using loc Function

data.loc[(data.age >= 12) & (data.gender == 'M')]
```



```
In [5]: data.loc[(data.age >= 12) & (data.gender == 'M')]
Out[5]:
```

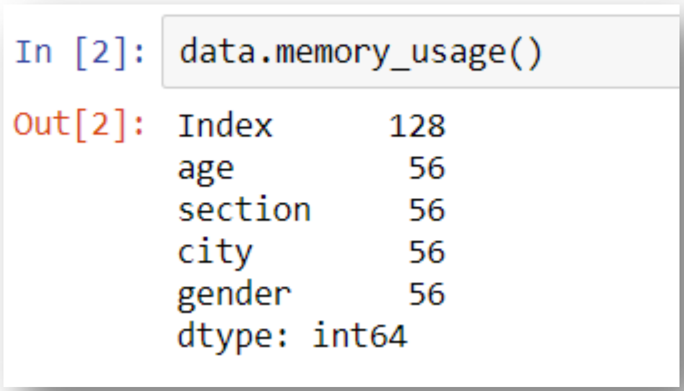
	age	section	city	gender
3	21	B	Delhi	M
4	12	B	Mumbai	M

```
In [ ]:

In [ ]:
```

Q34. What is Memory_usage function in Pandas ?

A34. Memory_usage() returns how much memory each column uses in bytes. It is very useful when we are working with large dataframes.

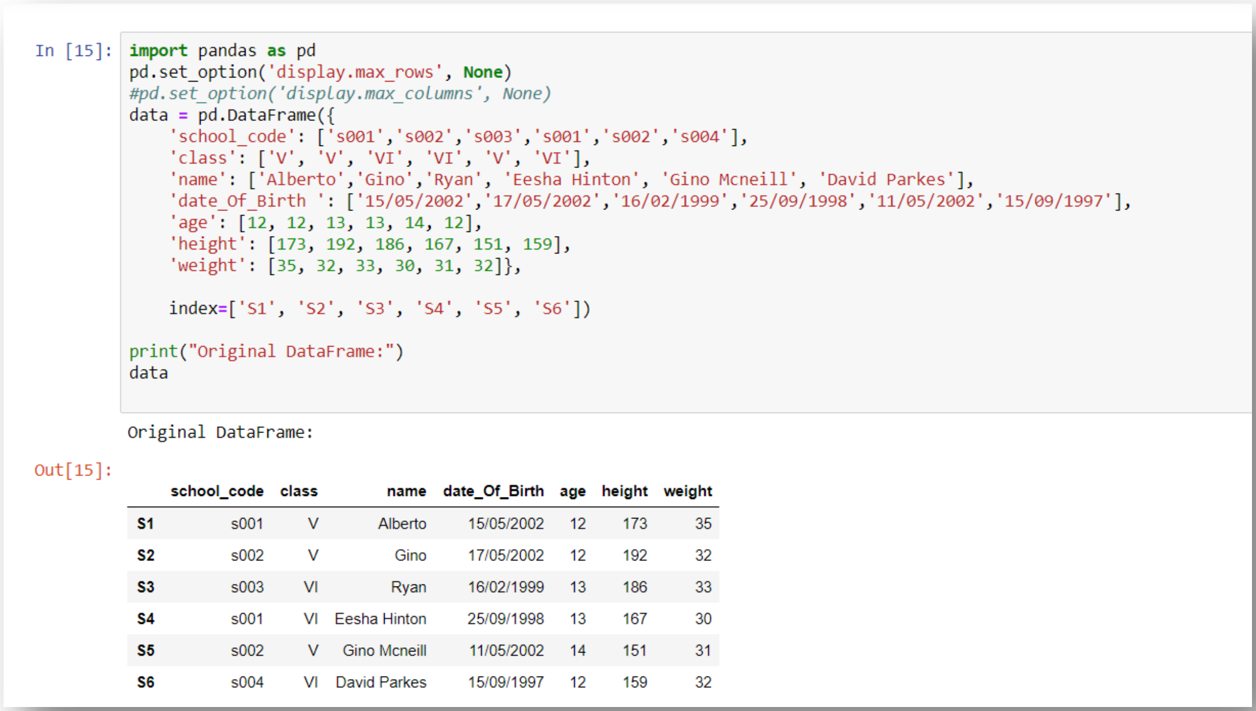
```
data.memory_usage()
```



```
In [2]: data.memory_usage()
Out[2]: Index      128
        age         56
        section     56
        city        56
        gender      56
        dtype: int64
```

For the given DataFrame answer the following questions:

```
# Importing the Pandas library

import pandas as pd

data = pd.DataFrame({

    'school_code': ['s001','s002','s003','s001','s002','s004'],

    'class': ['V', 'V', 'VI', 'VI', 'V', 'VI'],

    'name': ['Alberto','Gino','Ryan', 'Eesha Hinton', 'Gino Mcneill', 'David
Parkes'],

    'date_Of_Birth ':
['15/05/2002','17/05/2002','16/02/1999','25/09/1998','11/05/2002','15/0
9/1997'],

    'age': [12, 12, 13, 13, 14, 12],

    'height': [173, 192, 186, 167, 151, 159],

    'weight': [35, 32, 33, 30, 31, 32],

    'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']},

    index=['S1', 'S2', 'S3', 'S4', 'S5', 'S6'])

print("Original DataFrame:")

print(data)
```



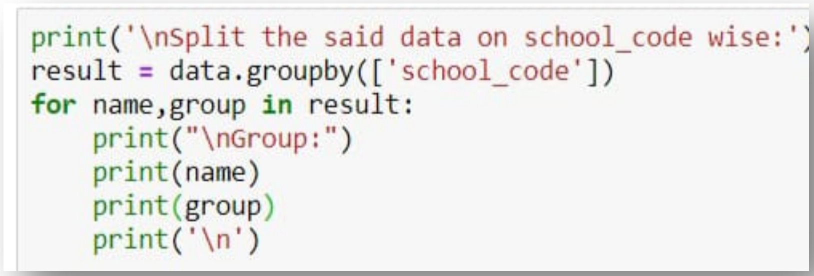
```
In [15]: import pandas as pd
         pd.set_option('display.max_rows', None)
         #pd.set_option('display.max_columns', None)
         data = pd.DataFrame({
             'school_code': ['s001','s002','s003','s001','s002','s004'],
             'class': ['V', 'V', 'VI', 'VI', 'V', 'VI'],
             'name': ['Alberto','Gino','Ryan', 'Eesha Hinton', 'Gino Mcneill', 'David Parkes'],
             'date_Of_Birth ': ['15/05/2002','17/05/2002','16/02/1999','25/09/1998','11/05/2002','15/09/1997'],
             'age': [12, 12, 13, 13, 14, 12],
             'height': [173, 192, 186, 167, 151, 159],
             'weight': [35, 32, 33, 30, 31, 32]},

             index=['S1', 'S2', 'S3', 'S4', 'S5', 'S6'])

         print("Original DataFrame:")
         data

         Original DataFrame:

Out[15]:
```

	school_code	class	name	date_Of_Birth	age	height	weight
S1	s001	V	Alberto	15/05/2002	12	173	35
S2	s002	V	Gino	17/05/2002	12	192	32
S3	s003	VI	Ryan	16/02/1999	13	186	33
S4	s001	VI	Eesha Hinton	25/09/1998	13	167	30
S5	s002	V	Gino Mcneill	11/05/2002	14	151	31
S6	s004	VI	David Parkes	15/09/1997	12	159	32

Q35. What is Group by function in Pandas and write a pandas program to split the dataframe into groups based on college code.

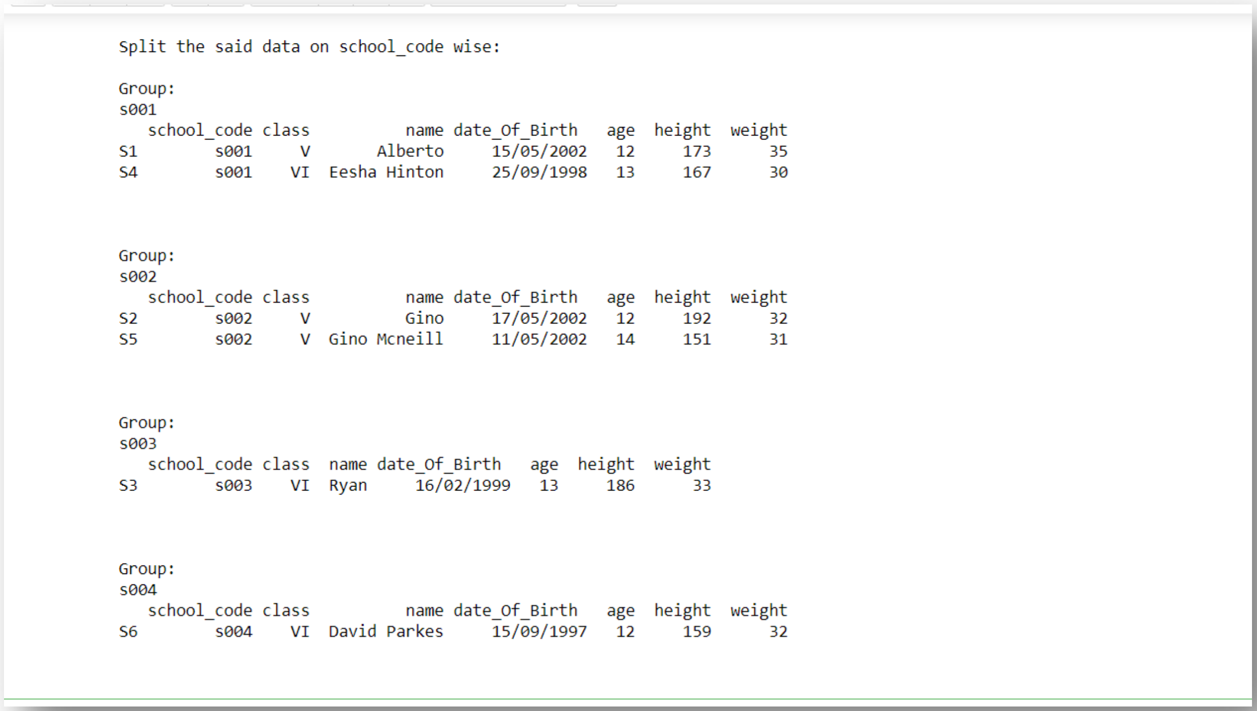
A35.

```
print('\nSplit the said data on school_code wise:')

result = data.groupby(['school_code'])

for name,group in result:

   print("\nGroup:")

   print(name)

   print(group)

   print('\n')
```



```
print('\nSplit the said data on school_code wise:'
result = data.groupby(['school_code'])
for name,group in result:
    print("\nGroup:")
    print(name)
    print(group)
    print('\n')
```

Output :



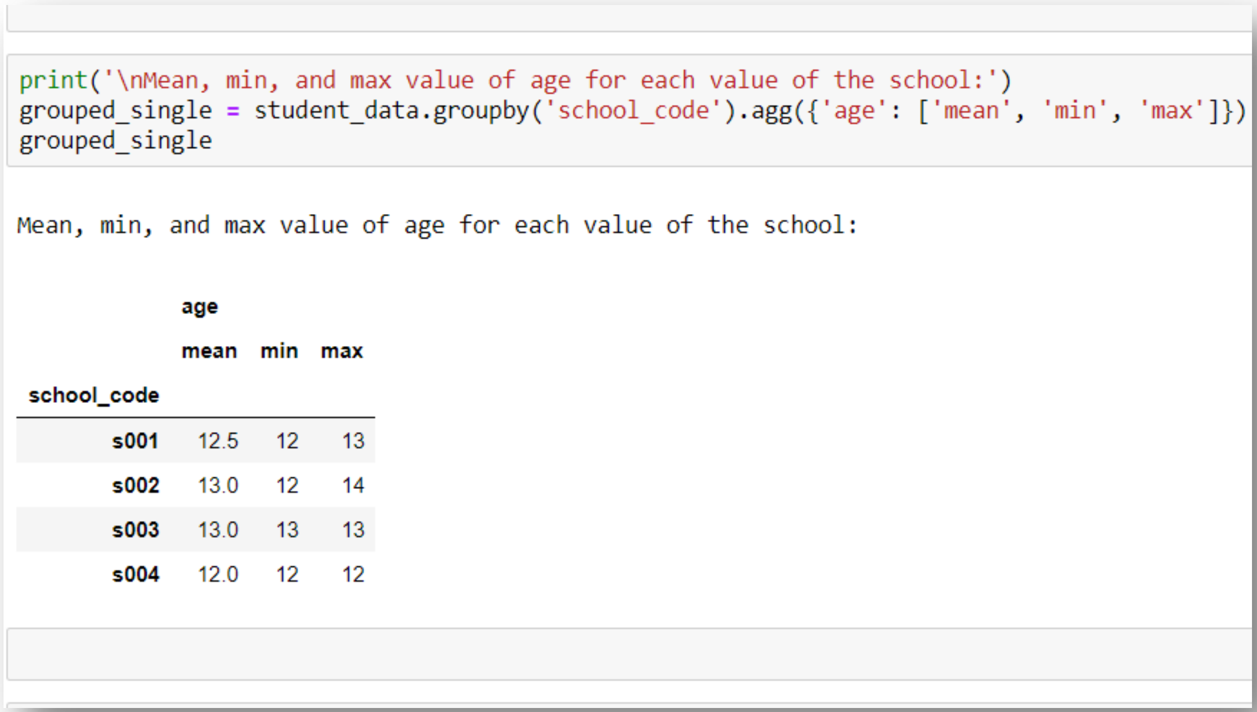
```
Split the said data on school_code wise:

Group:
s001
   school_code class          name date_Of_Birth   age  height  weight
S1        s001     V       Alberto    15/05/2002    12     173      35
S4        s001    VI  Eesha Hinton    25/09/1998    13     167      30



Group:
s002
   school_code class          name date_Of_Birth   age  height  weight
S2        s002     V          Gino    17/05/2002    12     192      32
S5        s002     V  Gino Mcneill    11/05/2002    14     151      31



Group:
s003
   school_code class  name date_Of_Birth   age  height  weight
S3        s003    VI  Ryan    16/02/1999    13     186      33



Group:
s004
   school_code class          name date_Of_Birth   age  height  weight
S6        s004    VI  David Parkes    15/09/1997    12     159      32
```

Q36. Write a Pandas program to split the following dataframe into group based on school_code and get min , max and mean value of age for each school ?

A36.

```
print('\nMean , min, and max value of age for each value of the school:')

group = data.groupby('school_code').agg({'age': ['mean', 'min', 'max']})
```

```
group
```



```
print('\nMean, min, and max value of age for each value of the school:')
grouped_single = student_data.groupby('school_code').agg({'age': ['mean', 'min', 'max']})
grouped_single
```

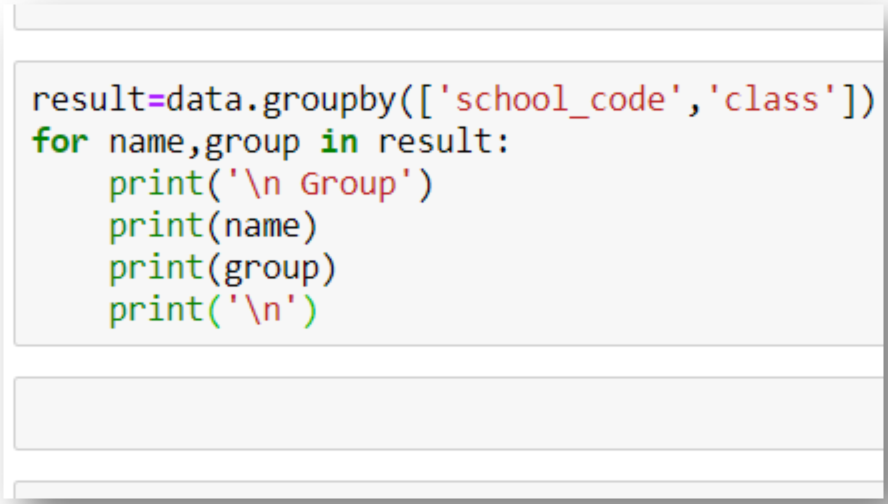
```
Mean, min, and max value of age for each value of the school:
```

	age		
	mean	min	max
school_code			
s001	12.5	12	13
s002	13.0	12	14
s003	13.0	13	13
s004	12.0	12	12

Q37. Write a Pandas Program to split the dataframe into group based on school_code and class.

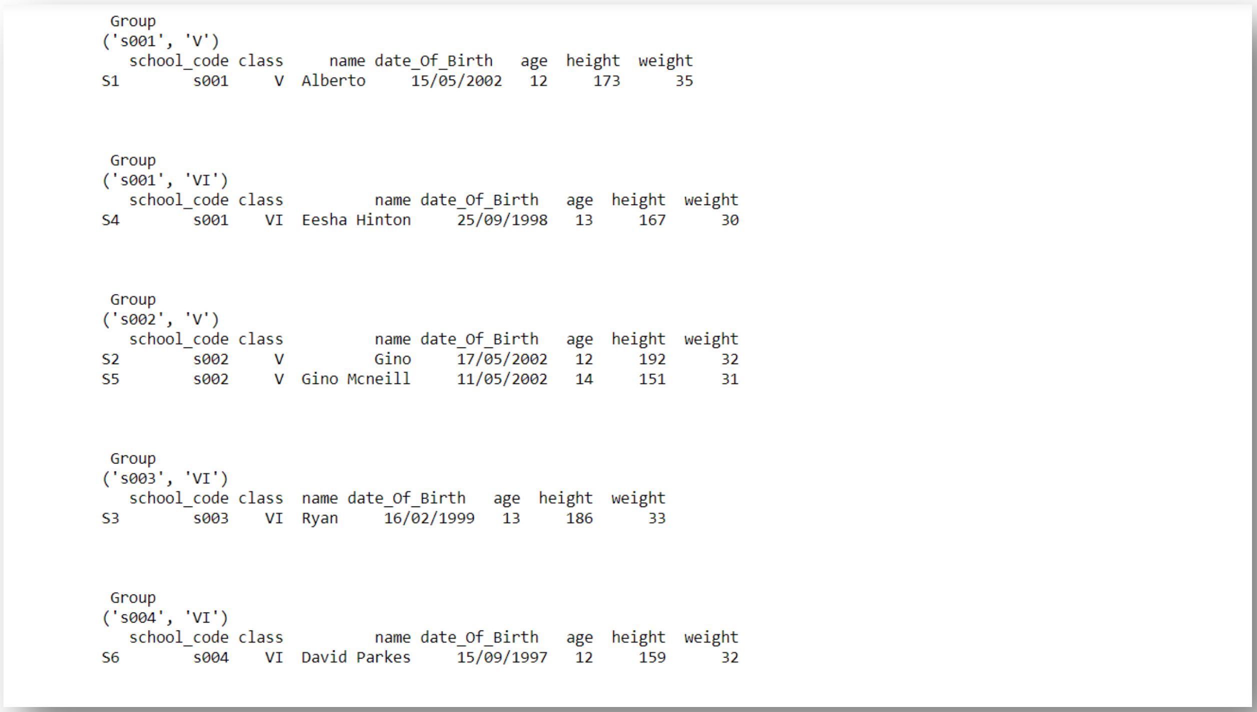
A37. Code :

```
result=data.groupby(['school_code','class'])

for name,group in result:

   print('\n Group')

   print(name)

   print(group)

   print('\n')
```



```
result=data.groupby(['school_code','class'])
for name,group in result:
    print('\n Group')
    print(name)
    print(group)
    print('\n')
```

Output :

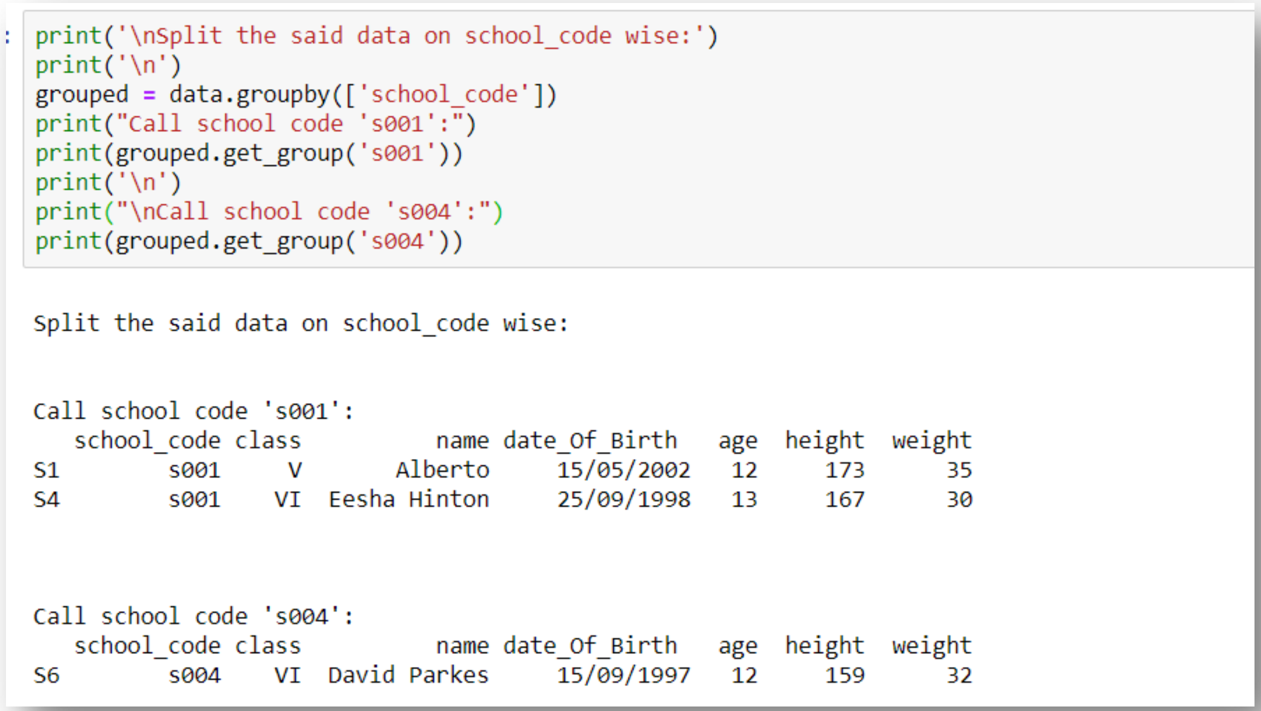


```
 Group
('s001', 'V')
   school_code class     name date_Of_Birth   age  height  weight
S1        s001     V  Alberto     15/05/2002   12     173      35


 Group
('s001', 'VI')
   school_code class          name date_Of_Birth   age  height  weight
S4        s001    VI  Eesha Hinton     25/09/1998   13     167      30


 Group
('s002', 'V')
   school_code class          name date_Of_Birth   age  height  weight
S2        s002     V          Gino     17/05/2002   12     192      32
S5        s002     V  Gino Mcneill     11/05/2002   14     151      31


 Group
('s003', 'VI')
   school_code class  name date_Of_Birth   age  height  weight
S3        s003    VI  Ryan     16/02/1999   13     186      33


 Group
('s004', 'VI')
   school_code class          name date_Of_Birth   age  height  weight
S6        s004    VI  David Parkes     15/09/1997   12     159      32
```

Q38. Write a Pandas program to split a dataframe into group based on school_code and call a specific group with the name.

A38. There is a function called get_group() in the pandas with the help of which we can call any particular group from the dataframe.
Code :

```
print('\nSplit the said data on school_code wise:')

grouped = data.groupby(['school_code'])

print("Call school code 's001':")

print(grouped.get_group('s001'))

print('\n')

print("\nCall school code 's004':")

print(grouped.get_group('s004'))
```



```
print('\nSplit the said data on school_code wise:')
print('\n')
grouped = data.groupby(['school_code'])
print("Call school code 's001':")
print(grouped.get_group('s001'))
print('\n')
print("\nCall school code 's004':")
print(grouped.get_group('s004'))
```

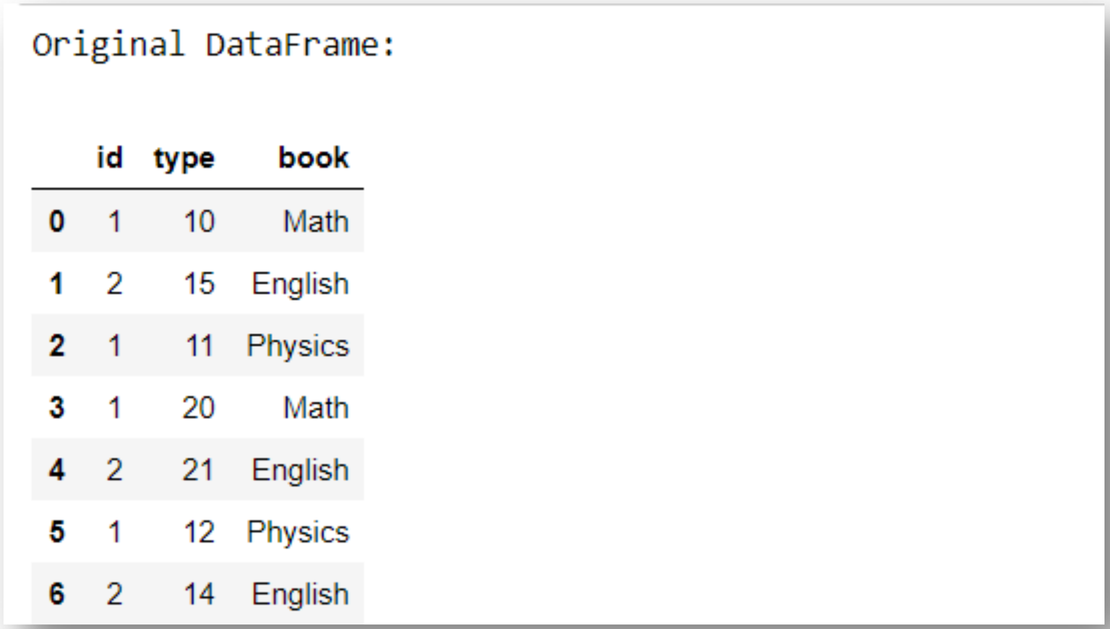
```
Split the said data on school_code wise:


Call school code 's001':
   school_code class         name date_Of_Birth   age  height  weight
S1        s001     V      Alberto     15/05/2002   12     173      35
S4        s001    VI Eesha Hinton     25/09/1998   13     167      30



Call school code 's004':
   school_code class         name date_Of_Birth   age  height  weight
S6        s004    VI  David Parkes    15/09/1997   12     159      32
```

Q39. Write a Pandas program to split the dataframe into groups based on all columns and calculate value_counts of each subject.

A39.

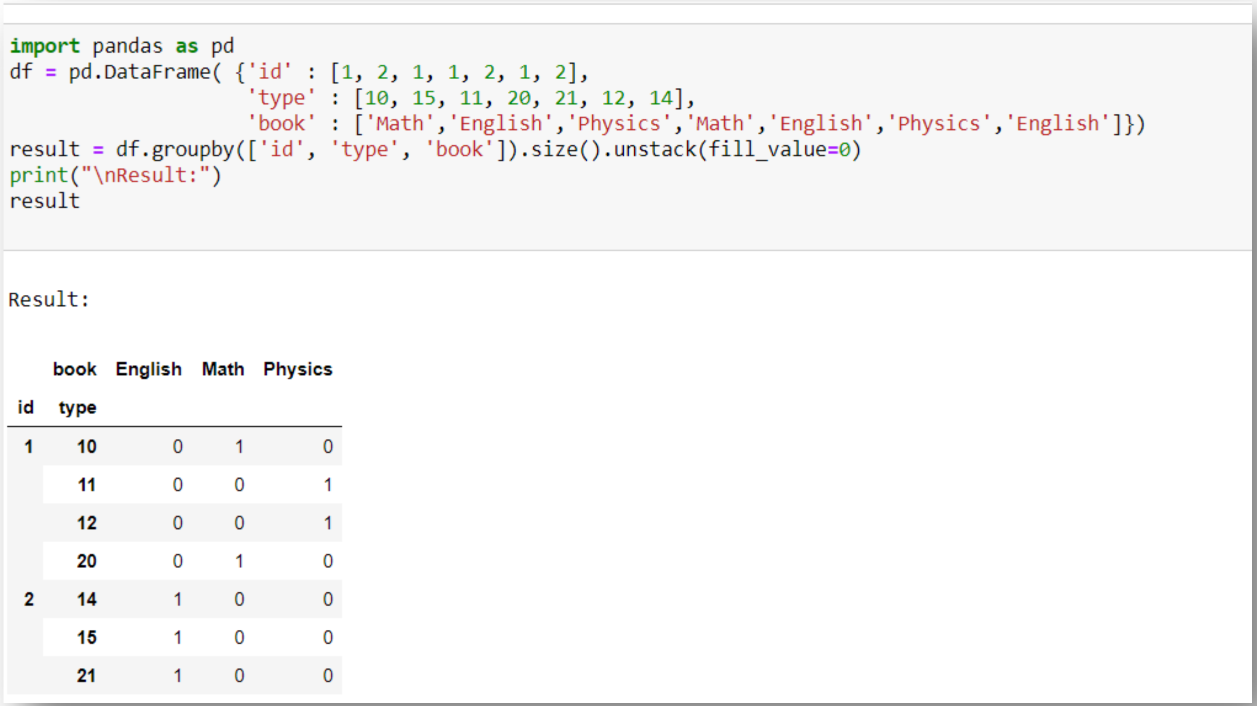
Code :

```
import pandas as pd

df = pd.DataFrame( {'id' : [1, 2, 1, 1, 2, 1, 2],

                'type' : [10, 15, 11, 20, 21, 12, 14],

                'book' :
['Math','English','Physics','Math','English','Physics','English']})

print("Original DataFrame:")

result = df.groupby(['id', 'type', 'book']).size().unstack(fill_value=0)

print("\nResult:")
```



```
Original DataFrame:
```

	id	type	book
0	1	10	Math
1	2	15	English
2	1	11	Physics
3	1	20	Math
4	2	21	English
5	1	12	Physics
6	2	14	English



```
import pandas as pd
df = pd.DataFrame( {'id' : [1, 2, 1, 1, 2, 1, 2],
                    'type' : [10, 15, 11, 20, 21, 12, 14],
                    'book' : ['Math','English','Physics','Math','English','Physics','English']})
result = df.groupby(['id', 'type', 'book']).size().unstack(fill_value=0)
print("\nResult:")
result
```

```
Result:

     book  English  Math  Physics
id  type
1   10           0     1        0
    11           0     0        1
    12           0     0        1
    20           0     1        0
2   14           1     0        0
    15           1     0        0
    21           1     0        0
```

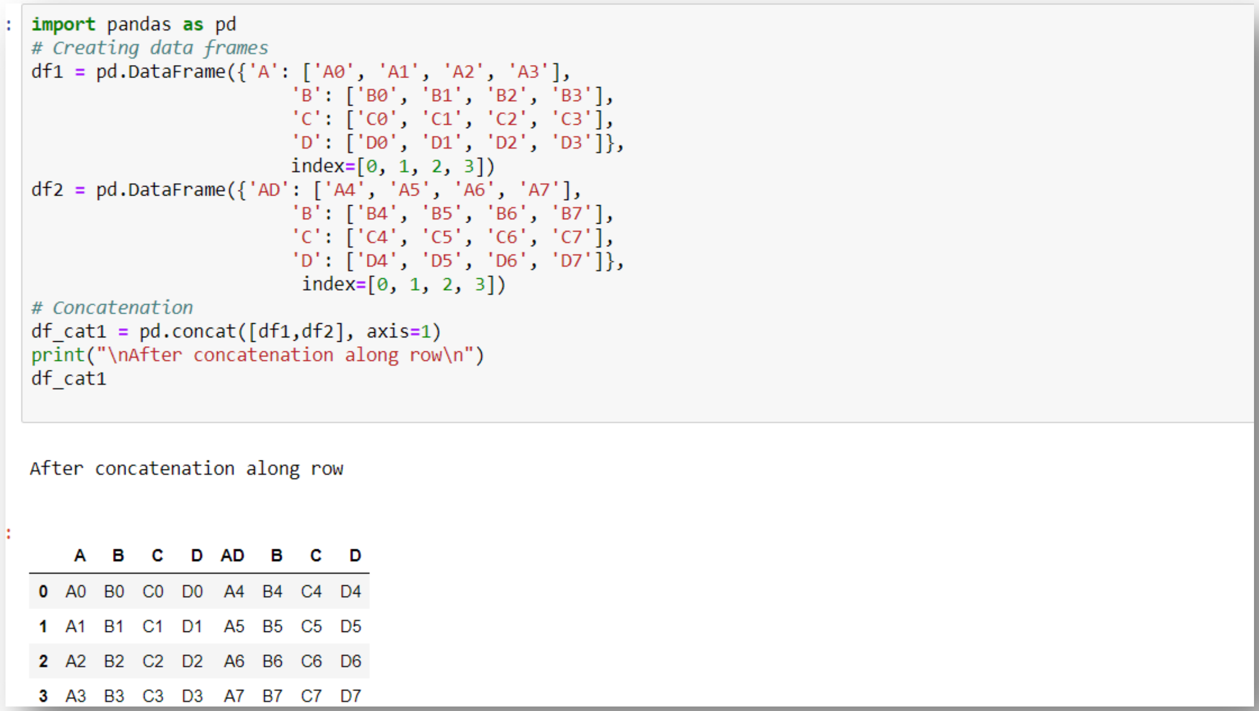
Q40. How to concatenate two or more than two dataframes. Explain with the help of Example.

A40. We can concatenate two dataframes by using concat () function. Which is a inbuilt function of pandas library.

Let's suppose we have 2 dataframes:

```
# Creating data frames

df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3'],
                    'C': ['C0', 'C1', 'C2', 'C3'],
                    'D': ['D0', 'D1', 'D2', 'D3']},
                    index=[0, 1, 2, 3])

df2 = pd.DataFrame({'AD': ['A4', 'A5', 'A6', 'A7'],
                    'B': ['B4', 'B5', 'B6', 'B7'],
                    'C': ['C4', 'C5', 'C6', 'C7'],
                    'D': ['D4', 'D5', 'D6', 'D7']},
                     index=[0, 1, 2, 3])
```

```
# Concatenation

df_cat1 = pd.concat([df1,df2], axis=1)
print("\nAfter concatenation along row\n")
df_cat1
```



```
import pandas as pd
# Creating data frames
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                        'B': ['B0', 'B1', 'B2', 'B3'],
                        'C': ['C0', 'C1', 'C2', 'C3'],
                        'D': ['D0', 'D1', 'D2', 'D3']},
                        index=[0, 1, 2, 3])
df2 = pd.DataFrame({'AD': ['A4', 'A5', 'A6', 'A7'],
                        'B': ['B4', 'B5', 'B6', 'B7'],
                        'C': ['C4', 'C5', 'C6', 'C7'],
                        'D': ['D4', 'D5', 'D6', 'D7']},
                         index=[0, 1, 2, 3])
# Concatenation
df_cat1 = pd.concat([df1,df2], axis=1)
print("\nAfter concatenation along row\n")
df_cat1
```

```
After concatenation along row
```

	A	B	C	D	AD	B	C	D
0	A0	B0	C0	D0	A4	B4	C4	D4
1	A1	B1	C1	D1	A5	B5	C5	D5
2	A2	B2	C2	D2	A6	B6	C6	D6
3	A3	B3	C3	D3	A7	B7	C7	D7

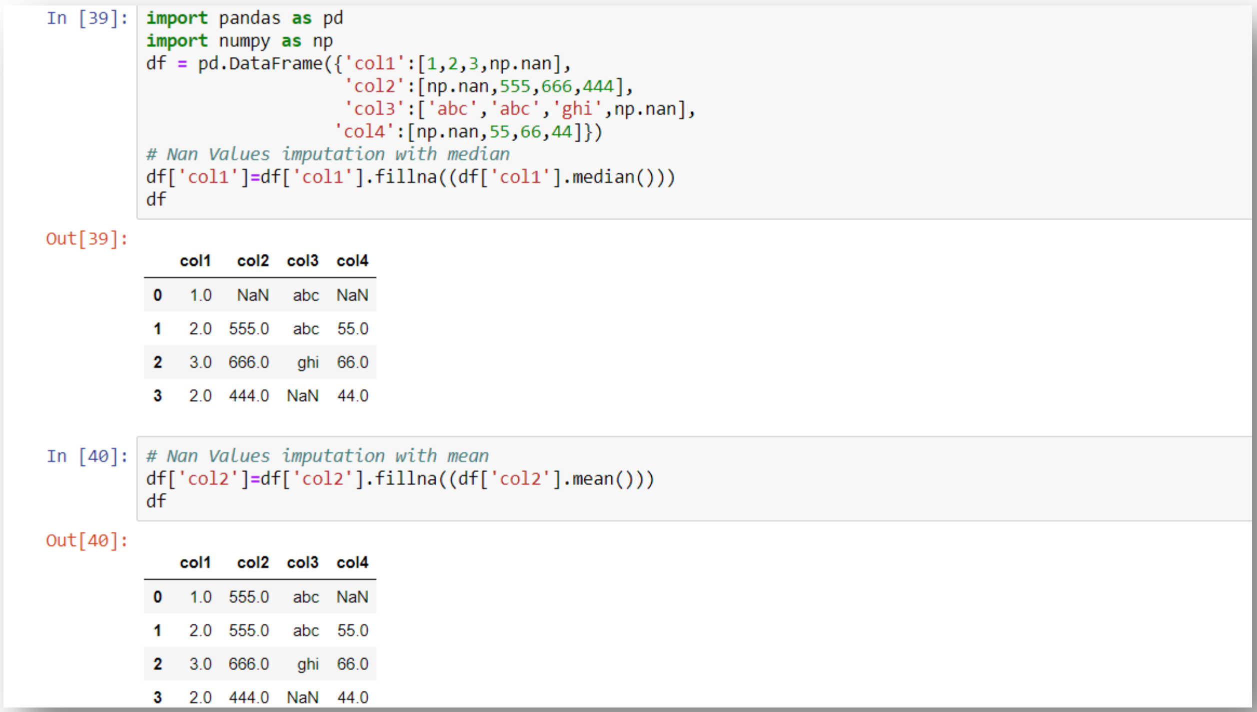
Q41. How to fill Nan Values in a dataframe ?

A41. Pandas dataframe has an inbuilt function named Fillna(). Then with the help of fillna() function we can replace Null values with mean , median ,mode or any constant value.

Example :

```
df = pd.DataFrame({'col1':[1,2,3,np.nan],
              'col2':[None,555,666,444],
              'col3':['abc','def','ghi','xyz']})

# Nan Values imputation with median
df['col1']=df['col1'].fillna((df['col1'].median()))
```

```
# Nan Values imputation with mean
df['col1']=df['col1'].fillna((df['col1'].mean()))

# Nan Values imputation with mode
df['col1']=df['col1'].fillna((df['col1'].mode()))

# Nan Values imputation with any constant
df.fillna('Fill')
```



```
In [39]: import pandas as pd
         import numpy as np
         df = pd.DataFrame({'col1':[1,2,3,np.nan],
                            'col2':[np.nan,555,666,444],
                            'col3':['abc','abc','ghi',np.nan],
                           'col4':[np.nan,55,66,44]})
         # Nan Values imputation with median
         df['col1']=df['col1'].fillna((df['col1'].median()))
         df
```

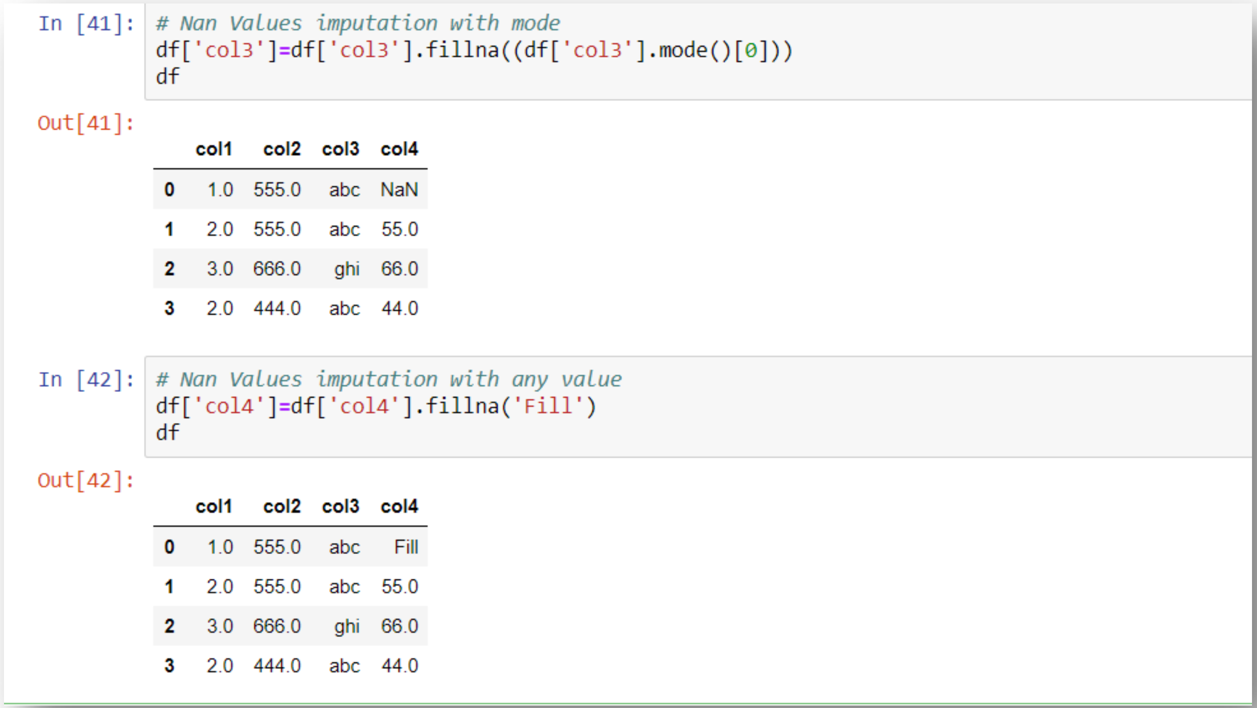
Out[39]:

	col1	col2	col3	col4
0	1.0	NaN	abc	NaN
1	2.0	555.0	abc	55.0
2	3.0	666.0	ghi	66.0
3	2.0	444.0	NaN	44.0

```
In [40]: # Nan Values imputation with mean
         df['col2']=df['col2'].fillna((df['col2'].mean()))
         df
```

Out[40]:

	col1	col2	col3	col4
0	1.0	555.0	abc	NaN
1	2.0	555.0	abc	55.0
2	3.0	666.0	ghi	66.0
3	2.0	444.0	NaN	44.0



```
In [41]: # Nan Values imputation with mode
         df['col3']=df['col3'].fillna((df['col3'].mode()[0]))
         df
```

Out[41]:

	col1	col2	col3	col4
0	1.0	555.0	abc	NaN
1	2.0	555.0	abc	55.0
2	3.0	666.0	ghi	66.0
3	2.0	444.0	abc	44.0

```
In [42]: # Nan Values imputation with any value
         df['col4']=df['col4'].fillna('Fill')
         df
```

Out[42]:

	col1	col2	col3	col4
0	1.0	555.0	abc	Fill
1	2.0	555.0	abc	55.0
2	3.0	666.0	ghi	66.0
3	2.0	444.0	abc	44.0

Q42. How can we sort the DataFrame?

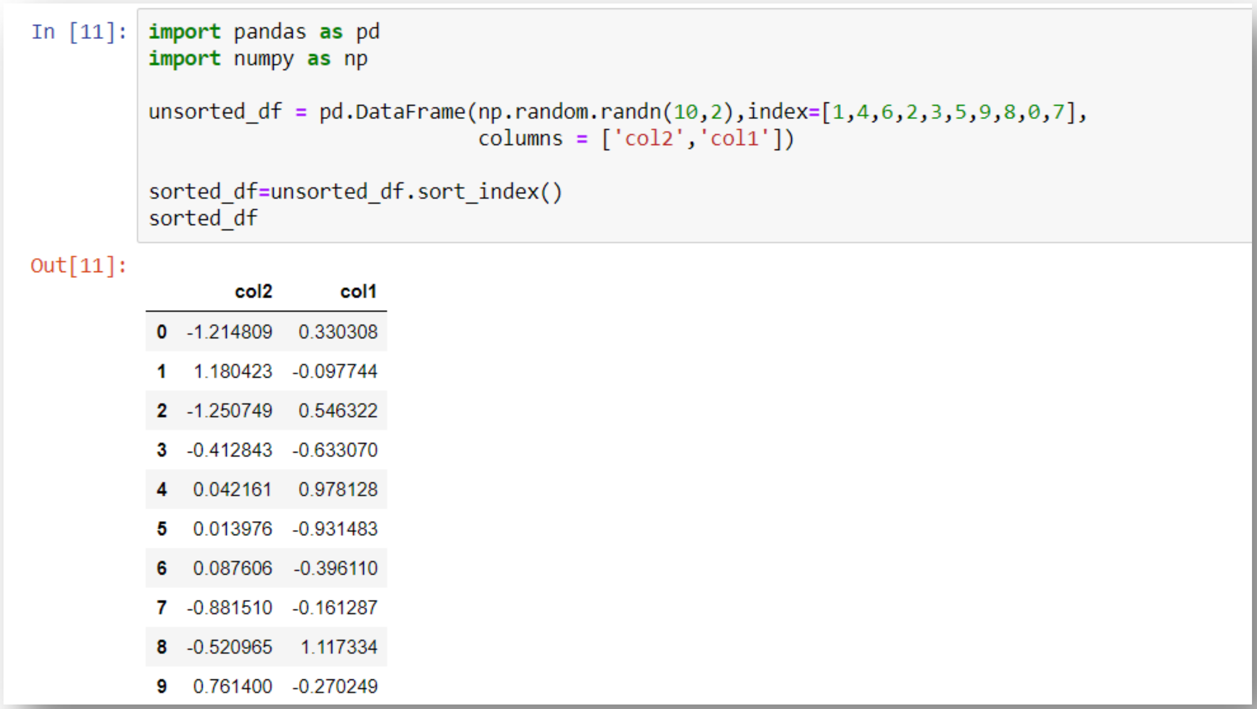
A42. We can sort the dataframe through different kinds:

- By labels
- By Actual value

By Labels:
The dataframe can be sorted using sort_index method. It can be done by passing the axis argument and the order of sorting.

Code :

```
import pandas as pd
import numpy as np

unsorted_df=pd.DataFrame(np.random.randn(10,2),index=[1,4,6,2,3,5,9,8,0,7]
,columns = ['col2','col1'])
sorted_df=unsorted_df.sort_index()
sorted_df
```



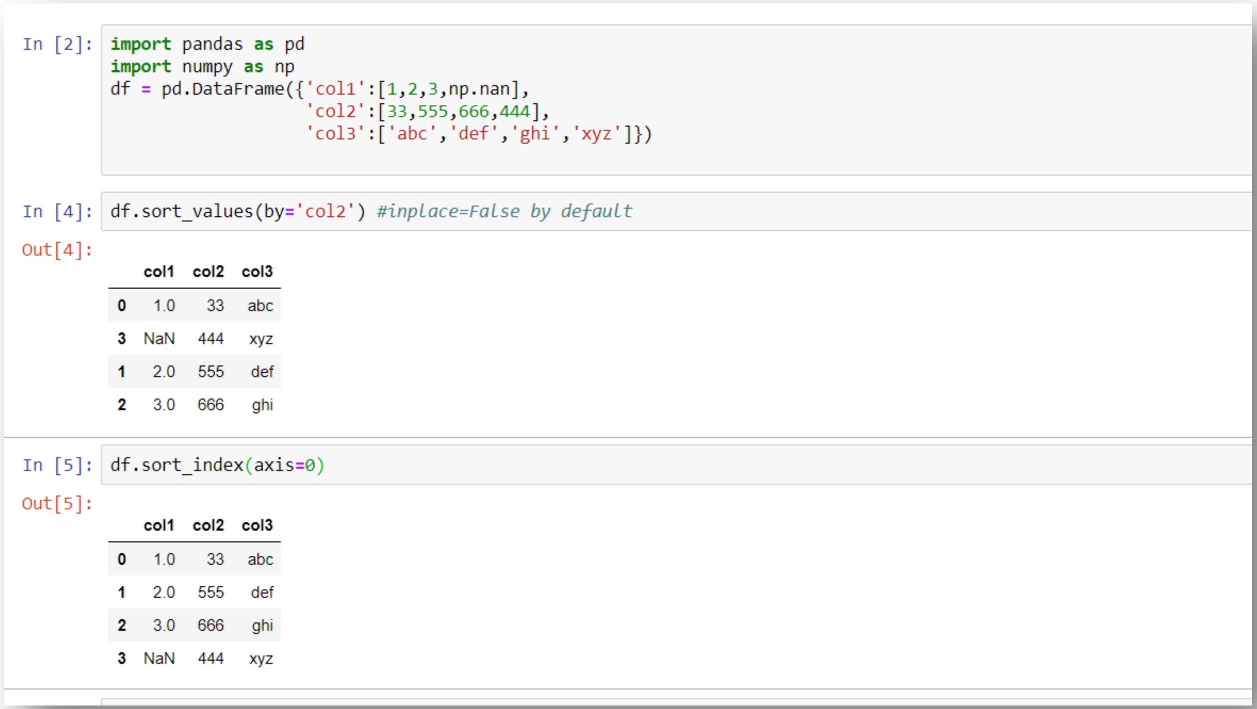
```
In [11]: import pandas as pd
         import numpy as np

         unsorted_df = pd.DataFrame(np.random.randn(10,2),index=[1,4,6,2,3,5,9,8,0,7],
                                    columns = ['col2','col1'])

         sorted_df=unsorted_df.sort_index()
         sorted_df
```

Out[11]:

	col2	col1
0	-1.214809	0.330308
1	1.180423	-0.097744
2	-1.250749	0.546322
3	-0.412843	-0.633070
4	0.042161	0.978128
5	0.013976	-0.931483
6	0.087606	-0.396110
7	-0.881510	-0.161287
8	-0.520965	1.117334
9	0.761400	-0.270249

By Actual value :
It is another kind of sorting the dataframe. sort_values() is a method for sorting the values.

```
df = pd.DataFrame({'col1':[1,2,3,np.nan],
             'col2':[33,555,666,444],
             'col3':['abc','def','ghi','xyz']})

# Sorting by actual values

df.sort_values(by='col2')
```



```
In [2]: import pandas as pd
        import numpy as np
        df = pd.DataFrame({'col1':[1,2,3,np.nan],
                           'col2':[33,555,666,444],
                           'col3':['abc','def','ghi','xyz']})

In [4]: df.sort_values(by='col2') #inplace=False by default
Out[4]:
```

	col1	col2	col3
0	1.0	33	abc
3	NaN	444	xyz
1	2.0	555	def
2	3.0	666	ghi

```
In [5]: df.sort_index(axis=0)
Out[5]:
```

	col1	col2	col3
0	1.0	33	abc
1	2.0	555	def
2	3.0	666	ghi
3	NaN	444	xyz

Q43. How can we convert Dataframe into excel file?

A43. We can convert the Dataframe into excel file by using to_excel() function.

Code:

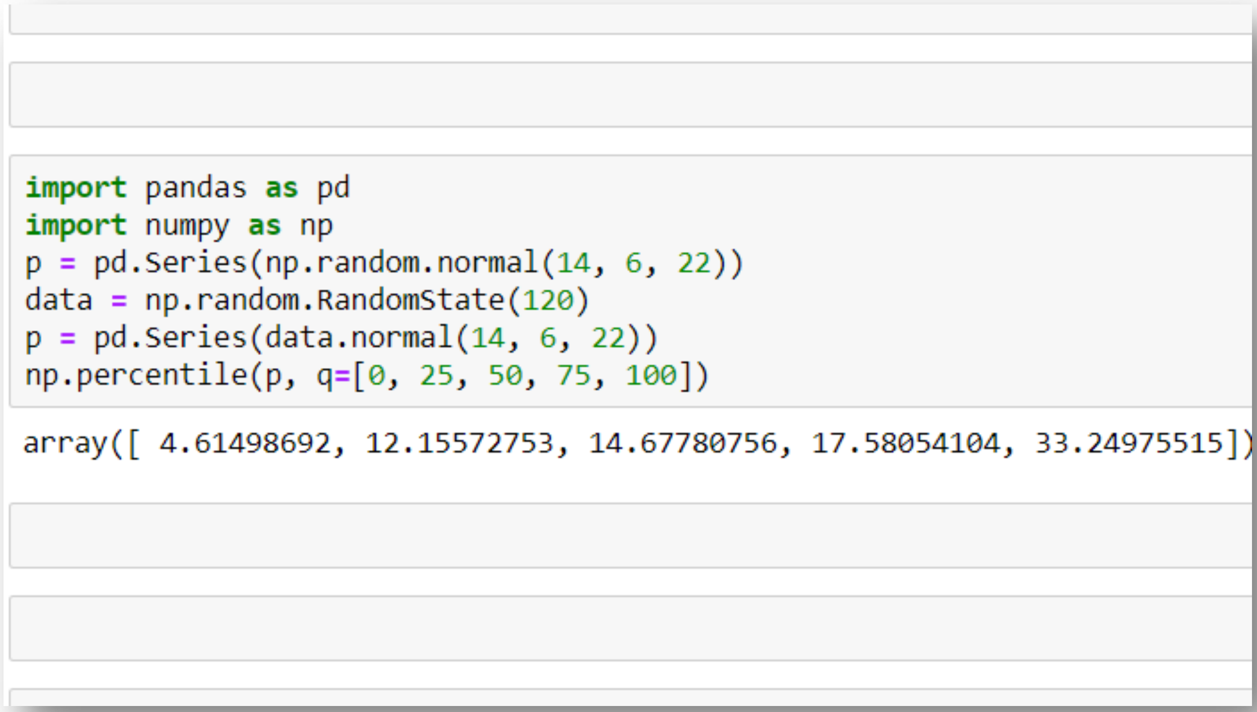
```
file_name="Xyz.xlsx"

df.to_excel(file_name)
```

Q44. How to get minimum, 25 percentile, median, 75 percentile and max of a numeric values?

A44 . We can compute all these values by using percentile method.

Code:

```
import pandas as pd
import numpy as np
p = pd.Series(np.random.normal(14, 6, 22))
data = np.random.RandomState(120)
p = pd.Series(data.normal(14, 6, 22))
np.percentile(p, q=[0, 25, 50, 75, 100])
```



```
import pandas as pd
import numpy as np
p = pd.Series(np.random.normal(14, 6, 22))
data = np.random.RandomState(120)
p = pd.Series(data.normal(14, 6, 22))
np.percentile(p, q=[0, 25, 50, 75, 100])
```

```
array([ 4.61498692, 12.15572753, 14.67780756, 17.58054104, 33.24975515])
```

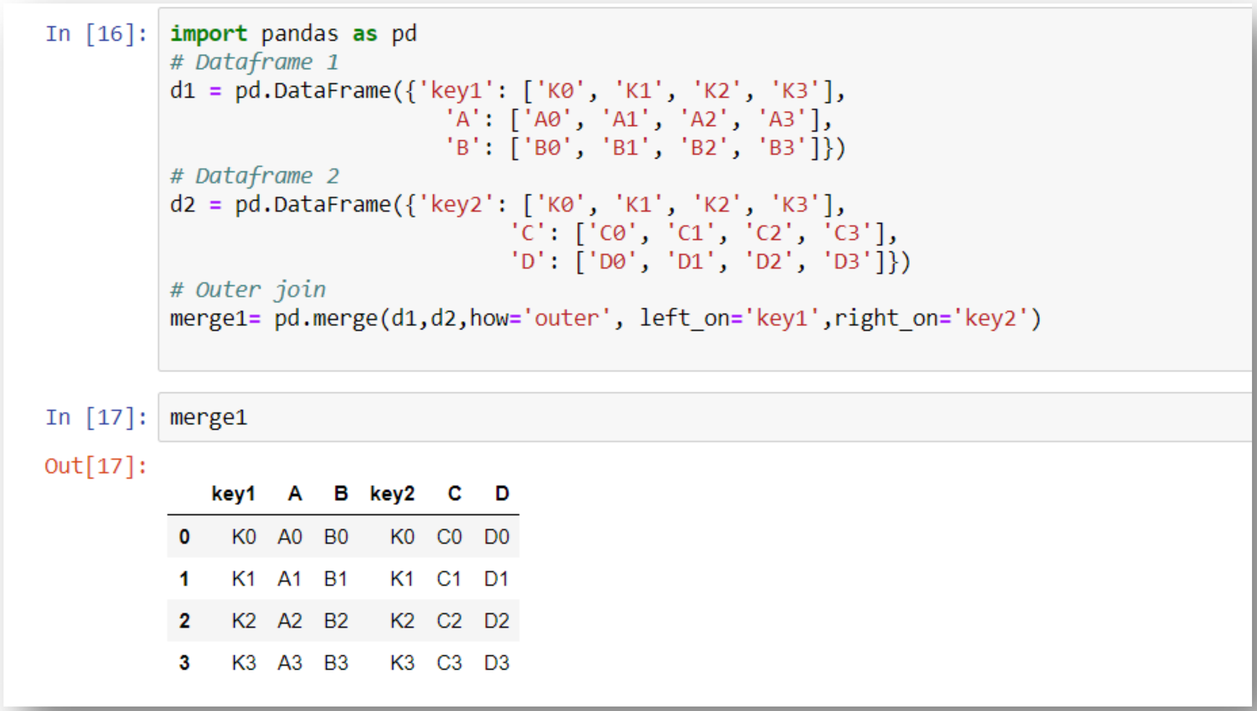
Q45. How to perform join operation by using merge function in the dataframe?

A45. When we need to combine the large dataframes, join serves a very powerful way to perform these operations swiftly. Pandas provide a single function, merge(), as the entry point for all standard database join operations between DataFrame objects.

Inside The merge function there is an argument “How”, with the help of which we can do the type of join, we want to perform.

Code :

```
import pandas as pd

# Dataframe 1
d1 = pd.DataFrame({'key1': ['K0', 'K1', 'K2', 'K3'],
                  'A': ['A0', 'A1', 'A2', 'A3'],
                  'B': ['B0', 'B1', 'B2', 'B3']})

# Dataframe 2

d2 = pd.DataFrame({'key2': ['K0', 'K1', 'K2', 'K3'],
                      'C': ['C0', 'C1', 'C2', 'C3'],
                      'D': ['D0', 'D1', 'D2', 'D3']})

#Inner join

pd.merge(df1, df2, how='inner')

# Outer join

merge1= pd.merge(d1,d2,how='outer', left_on='key1',right_on='key2')

print(merge1)
```



```
In [16]: import pandas as pd
         # Dataframe 1
         d1 = pd.DataFrame({'key1': ['K0', 'K1', 'K2', 'K3'],
                              'A': ['A0', 'A1', 'A2', 'A3'],
                              'B': ['B0', 'B1', 'B2', 'B3']})
         # Dataframe 2
         d2 = pd.DataFrame({'key2': ['K0', 'K1', 'K2', 'K3'],
                                 'C': ['C0', 'C1', 'C2', 'C3'],
                                 'D': ['D0', 'D1', 'D2', 'D3']})
         # Outer join
         merge1= pd.merge(d1,d2,how='outer', left_on='key1',right_on='key2')

In [17]: merge1

Out[17]:
```

	key1	A	B	key2	C	D
0	K0	A0	B0	K0	C0	D0
1	K1	A1	B1	K1	C1	D1
2	K2	A2	B2	K2	C2	D2
3	K3	A3	B3	K3	C3	D3

Q46. What is the difference between merge() and concat() function?

A46. The Difference between merge() and concat() function is :

- Merge() function is used to combine two(or more) dataframes on the basis of values of common columns.
- Concat() function is used to append one (or more) dataframes .

Q47. How can we create the copy of the series in pandas?

A47. We can create a copy of series by using the following syntax.

```
pandas.Series.copy ()
Series.copy(deep=True)
```

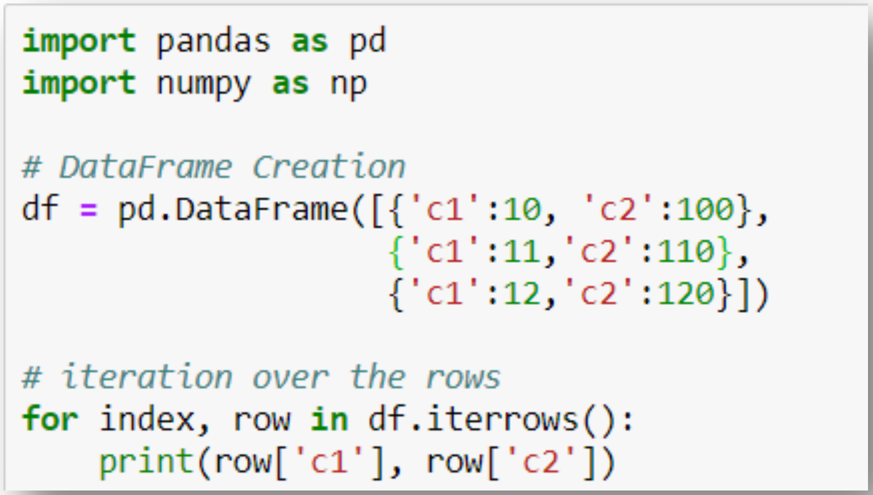
The above code make a deep copy that includes a copy of the data and the indices. If we set the value of deep to false, then it will neither copy the indices nor the data.

Q48. How to iterate over a pandas dataframe?

A48. We can iterate over the rows of the dataframe by using for loop in combination with iterrows () call on the dataframe.

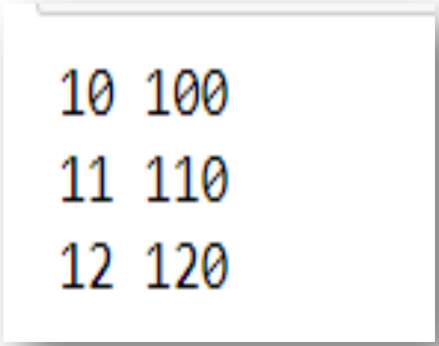
Code:

```
import pandas as pd
import numpy as np

# Dataframe creation
df = pd.DataFrame([{'c1':10, 'c2':100}, {'c1':11,'c2':110}, {'c1':12,'c2':120}])

# Iterate over the rows
for index, row in df.iterrows():
   print(row['c1'], row['c2'])
```



```
import pandas as pd
import numpy as np

# DataFrame Creation
df = pd.DataFrame([{'c1':10, 'c2':100},
                   {'c1':11,'c2':110},
                   {'c1':12,'c2':120}])

# iteration over the rows
for index, row in df.iterrows():
    print(row['c1'], row['c2'])
```

Output :



```
10 100
11 110
12 120
```

Q49. What is Data Aggregation?

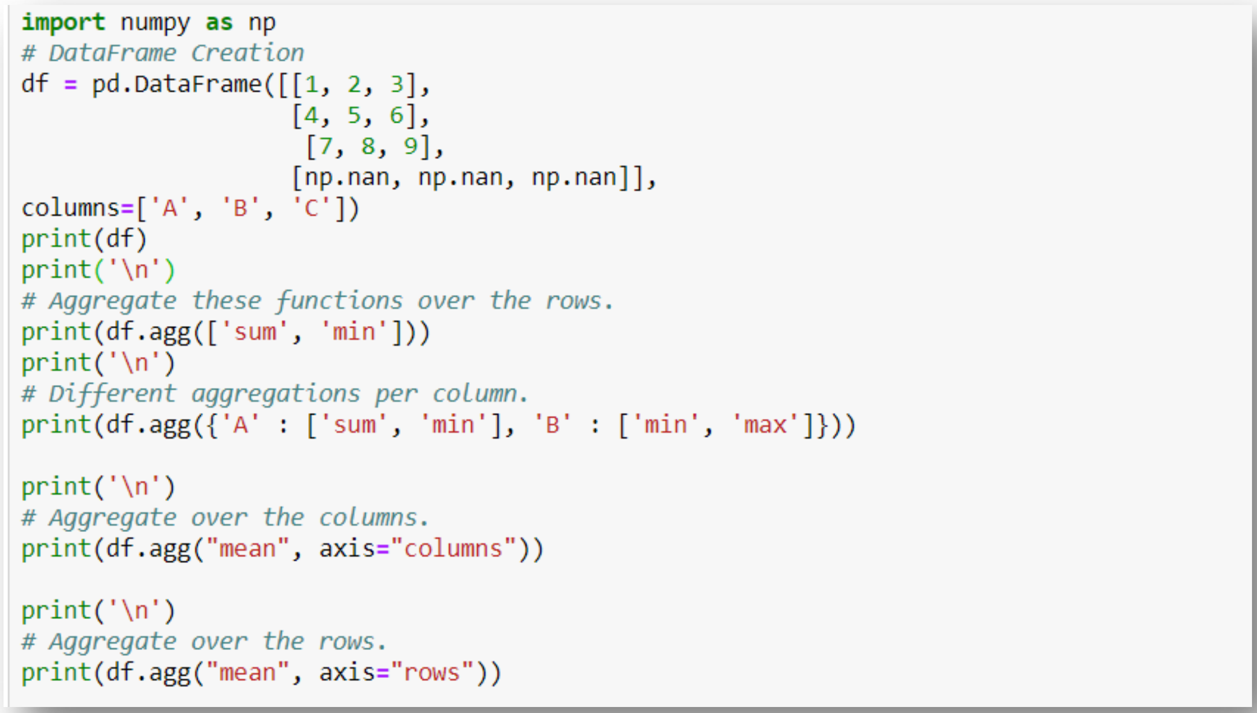
A49. The main task of Data Aggregation is to apply some aggregation to one or more columns.

- **sum:** It is used to return the sum of the values for the requested axis.
- **min:** It is used to return a minimum of the values for the requested axis.
- **max:** It is used to return a maximum values for the requested axis.

Code :

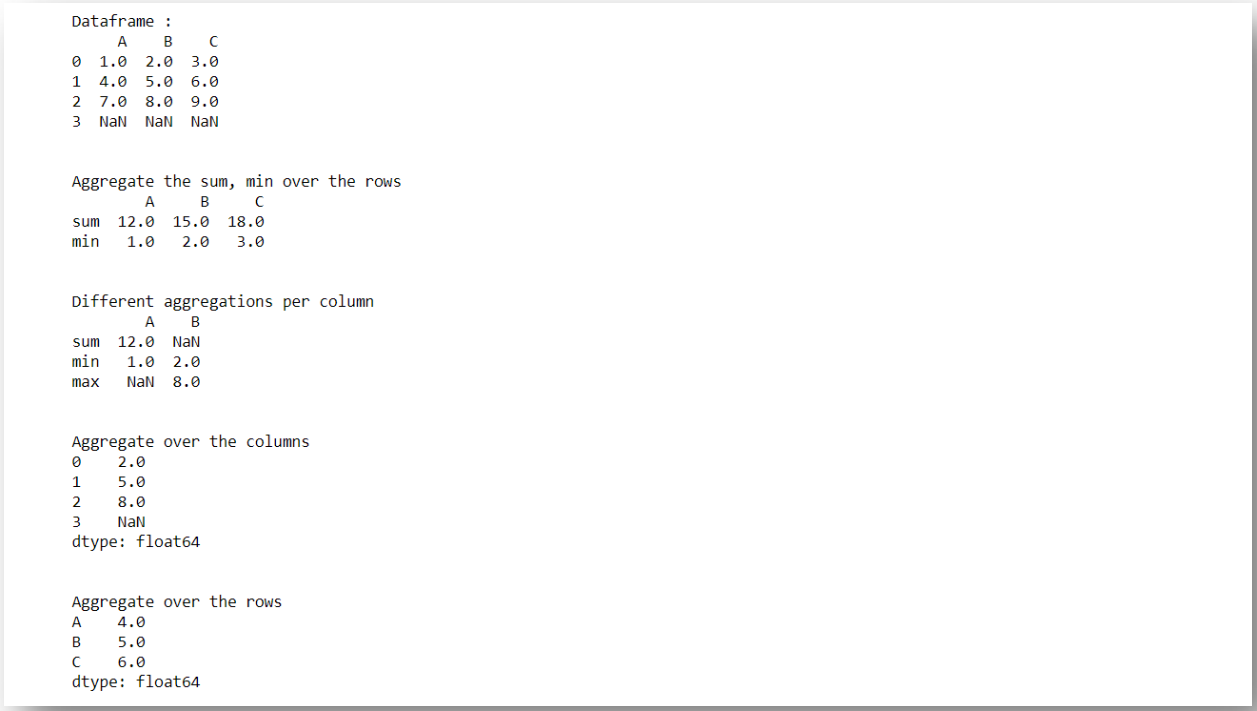
```
import pandas as pd
import numpy as np

# DataFrame Creation
df = pd.DataFrame([[1, 2, 3],
               [4, 5, 6],
                [7, 8, 9],
               [np.nan, np.nan, np.nan]],
columns=['A', 'B', 'C'])
print(df)

# Aggregate these functions over the rows.
print(df.agg(['sum', 'min']))

# Different aggregations per column.
print(df.agg({'A' : ['sum', 'min'], 'B' : ['min', 'max']}))

# Aggregate over the columns.
print(df.agg("mean", axis="columns"))
```

```
# Aggregate over the rows.
print(df.agg("mean", axis="rows"))
```



```
import numpy as np
# DataFrame Creation
df = pd.DataFrame([[1, 2, 3],
                   [4, 5, 6],
                    [7, 8, 9],
                   [np.nan, np.nan, np.nan]],
columns=['A', 'B', 'C'])
print(df)
print('\n')
# Aggregate these functions over the rows.
print(df.agg(['sum', 'min']))
print('\n')
# Different aggregations per column.
print(df.agg({'A' : ['sum', 'min'], 'B' : ['min', 'max']}))

print('\n')
# Aggregate over the columns.
print(df.agg("mean", axis="columns"))

print('\n')
# Aggregate over the rows.
print(df.agg("mean", axis="rows"))
```

Output :



```
Dataframe :
     A    B    C
0  1.0  2.0  3.0
1  4.0  5.0  6.0
2  7.0  8.0  9.0
3  NaN  NaN  NaN


Aggregate the sum, min over the rows
        A     B     C
sum  12.0  15.0  18.0
min   1.0   2.0   3.0


Different aggregations per column
        A    B
sum  12.0  NaN
min   1.0  2.0
max   NaN  8.0


Aggregate over the columns
0    2.0
1    5.0
2    8.0
3    NaN
dtype: float64


Aggregate over the rows
A    4.0
B    5.0
C    6.0
dtype: float64
```

Q**50. How can we calculate the standard deviation from the Series?**

A50. Pandas provides a inbuilt function named std(). With the help of which we can calculate the standard deviation of the dataframe or given set of numbers.

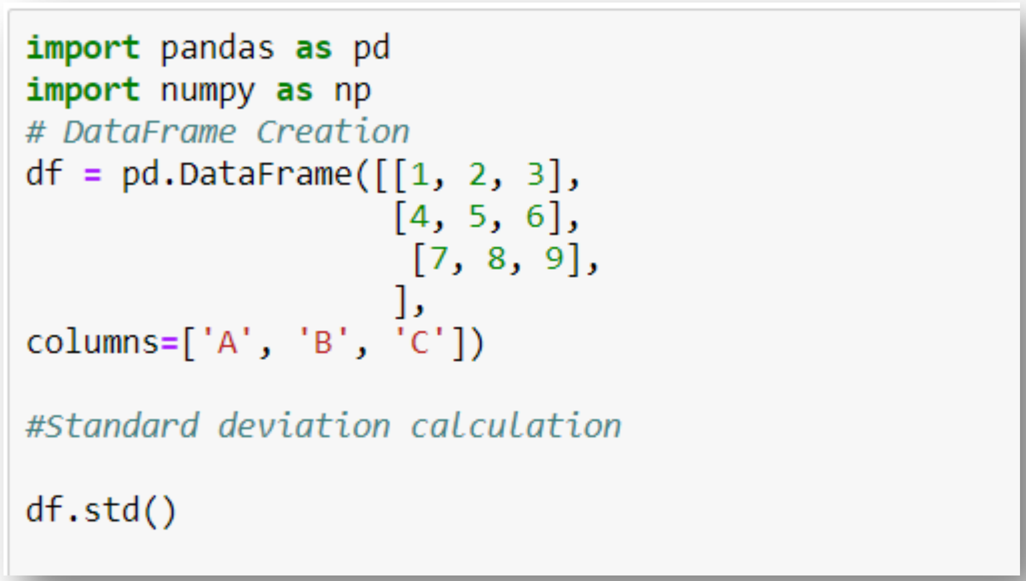
Syntax:

Series.std(axis=None, skipna=None, level=None, ddof=1, numeric_only=None, **kwargs)

Code :

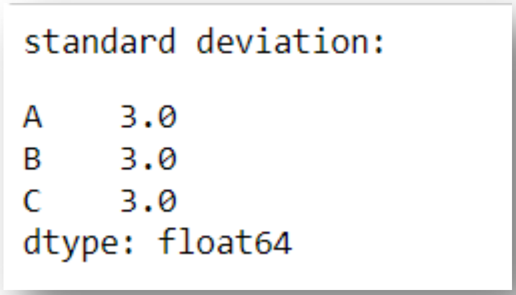
```
import pandas as pd
import numpy as np

# DataFrame Creation
df = pd.DataFrame([[1, 2, 3],
                   [4, 5, 6],
                    [7, 8,9 ],],
columns=['A', 'B', 'C'])
```

```
#Standard deviation calculation

df.std()
```



```
import pandas as pd
import numpy as np
# DataFrame Creation
df = pd.DataFrame([[1, 2, 3],
                   [4, 5, 6],
                    [7, 8, 9],
                   ],
columns=['A', 'B', 'C'])

#Standard deviation calculation

df.std()
```

Output :



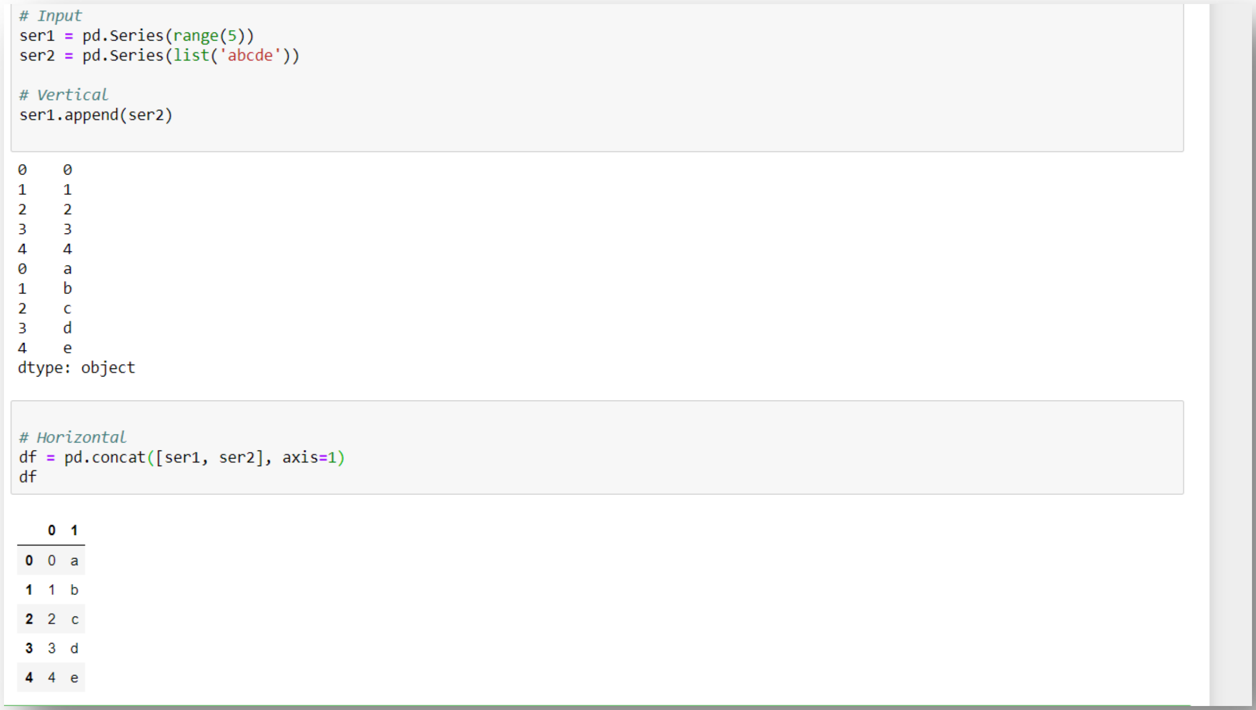
```
standard deviation:

A    3.0
B    3.0
C    3.0
dtype: float64
```

Q51. How to stack two series Vertically and horizontally ?

A51. Stack ser1 and ser2 vertically and horizontally (to form a dataframe)

Code :

```
# Input

ser1 = pd.Series(range(5))
ser2 = pd.Series(list('abcde'))

# Vertical
ser1.append(ser2)

# Horizontal
df = pd.concat([ser1, ser2], axis=1)
df
```



```
# Input
ser1 = pd.Series(range(5))
ser2 = pd.Series(list('abcde'))

# Vertical
ser1.append(ser2)
```

```
0    0
1    1
2    2
3    3
4    4
0    a
1    b
2    c
3    d
4    e
dtype: object
```

```
# Horizontal
df = pd.concat([ser1, ser2], axis=1)
df
```

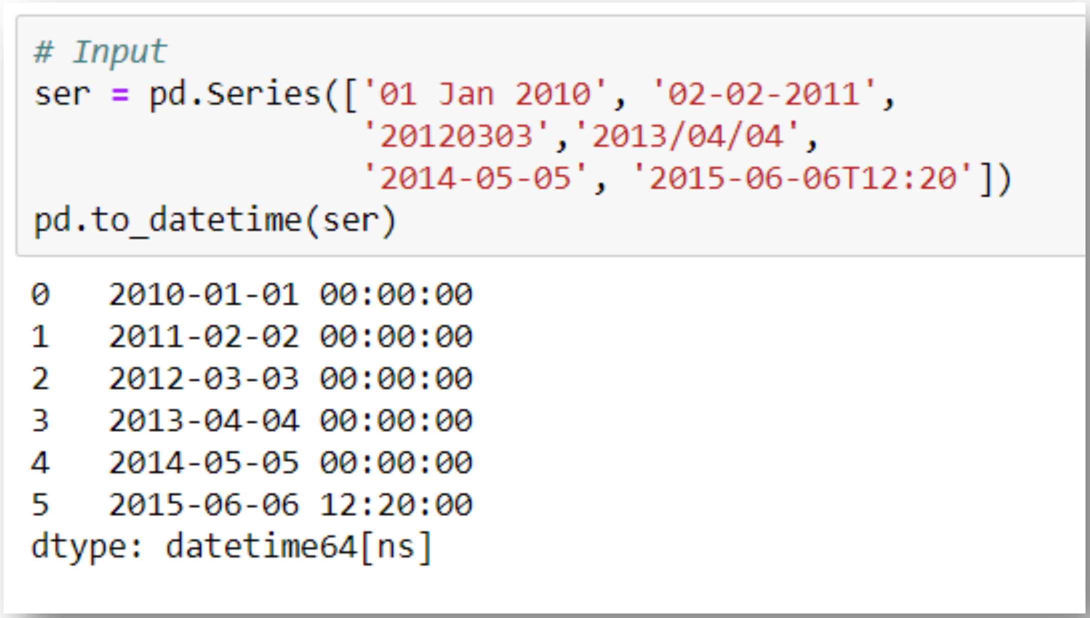
	0	1
0	0	a
1	1	b
2	2	c
3	3	d
4	4	e

Q52. How to convert a series of date-strings to a timeseries?

A52. We can convert a series to date-strings to a timeseries using pandas.

Code :

```
# Input

ser = pd.Series(['01 Jan 2010',  '02-02-2011',
                 '20120303',    '2013/04/04',
                    '2014-05-05',   '2015-06-06T12:20'])

# Conversion
pd.to_datetime(ser)
```



```
# Input
ser = pd.Series(['01 Jan 2010', '02-02-2011',
                 '20120303','2013/04/04',
                 '2014-05-05', '2015-06-06T12:20'])
pd.to_datetime(ser)
```

```
0   2010-01-01 00:00:00
1   2011-02-02 00:00:00
2   2012-03-03 00:00:00
3   2013-04-04 00:00:00
4   2014-05-05 00:00:00
5   2015-06-06 12:20:00
dtype: datetime64[ns]
```

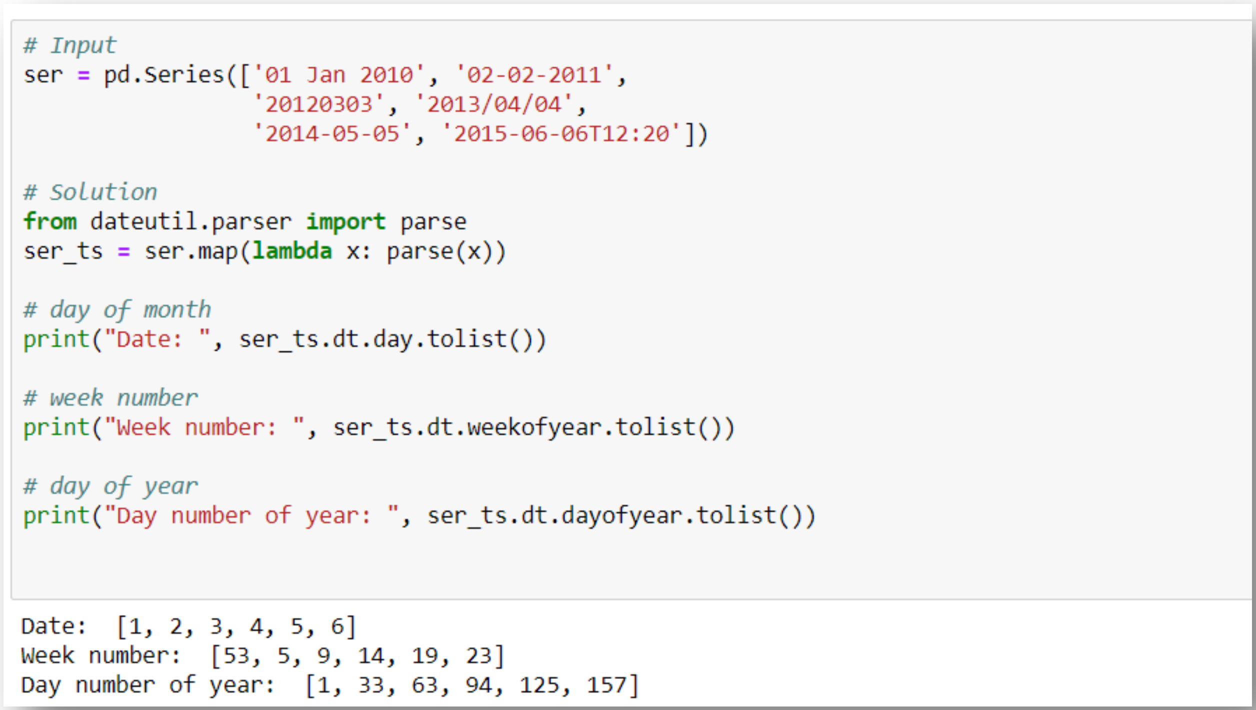
Q53. How to get the day of month, week number, day of year and day of week from a series of date strings?

A53. We can extract month , year, date from the date strings.

Code:

```
# Input
ser = pd.Series(['01 Jan 2010', '02-02-2011',
          '20120303', '2013/04/04',

          '2014-05-05', '2015-06-06T12:20'])


# Solution
from dateutil.parser import parse
ser_ts = ser.map(lambda x: parse(x))

# day of month
```

```
print("Date: ", ser_ts.dt.day.tolist())

# week number
print("Week number: ", ser_ts.dt.weekofyear.tolist())

# day of year
print("Day number of year: ", ser_ts.dt.dayofyear.tolist())
```



```
# Input
ser = pd.Series(['01 Jan 2010', '02-02-2011',
                 '20120303', '2013/04/04',
                 '2014-05-05', '2015-06-06T12:20'])

# Solution
from dateutil.parser import parse
ser_ts = ser.map(lambda x: parse(x))

# day of month
print("Date: ", ser_ts.dt.day.tolist())

# week number
print("Week number: ", ser_ts.dt.weekofyear.tolist())

# day of year
print("Day number of year: ", ser_ts.dt.dayofyear.tolist())
```

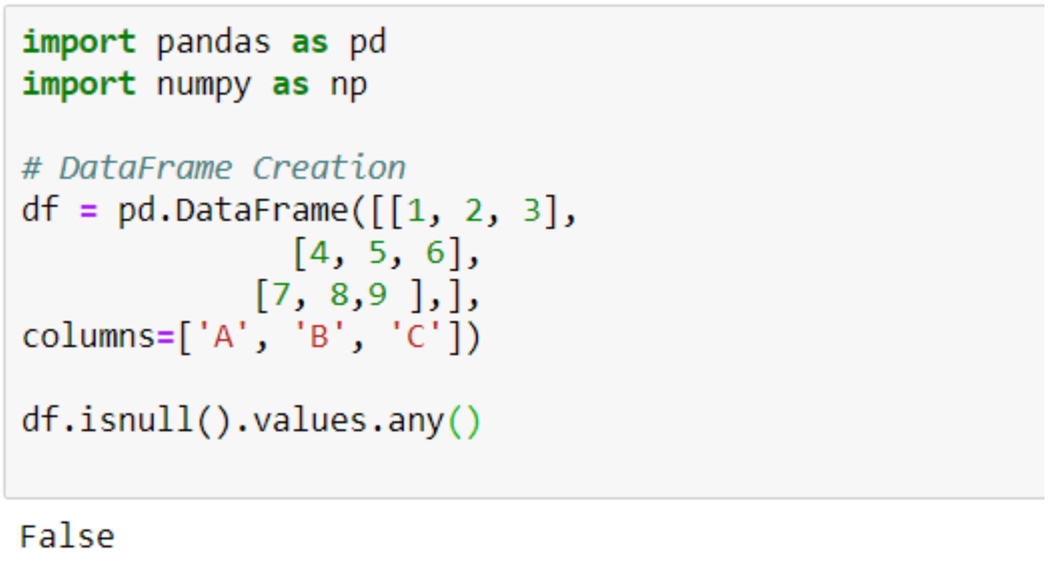
```
Date:  [1, 2, 3, 4, 5, 6]
Week number:  [53, 5, 9, 14, 19, 23]
Day number of year:  [1, 33, 63, 94, 125, 157]
```

Q54. How to check if a dataframe has any missing values ?

A54. We can check missing values in the dataframe with the help of isnull() function. The function returns True if dataframe contains any missing value otherwise it will return False.

Code :

```
import pandas as pd
import numpy as np

# DataFrame Creation
df = pd.DataFrame([[1, 2, 3],
                   [4, 5, 6],
                  [7, 8,9 ],],
columns=['A', 'B', 'C'])

# Missing values checking
df.isnull().values.any()
```



```
import pandas as pd
import numpy as np

# DataFrame Creation
df = pd.DataFrame([[1, 2, 3],
              [4, 5, 6],
            [7, 8,9 ],],
columns=['A', 'B', 'C'])

df.isnull().values.any()
```

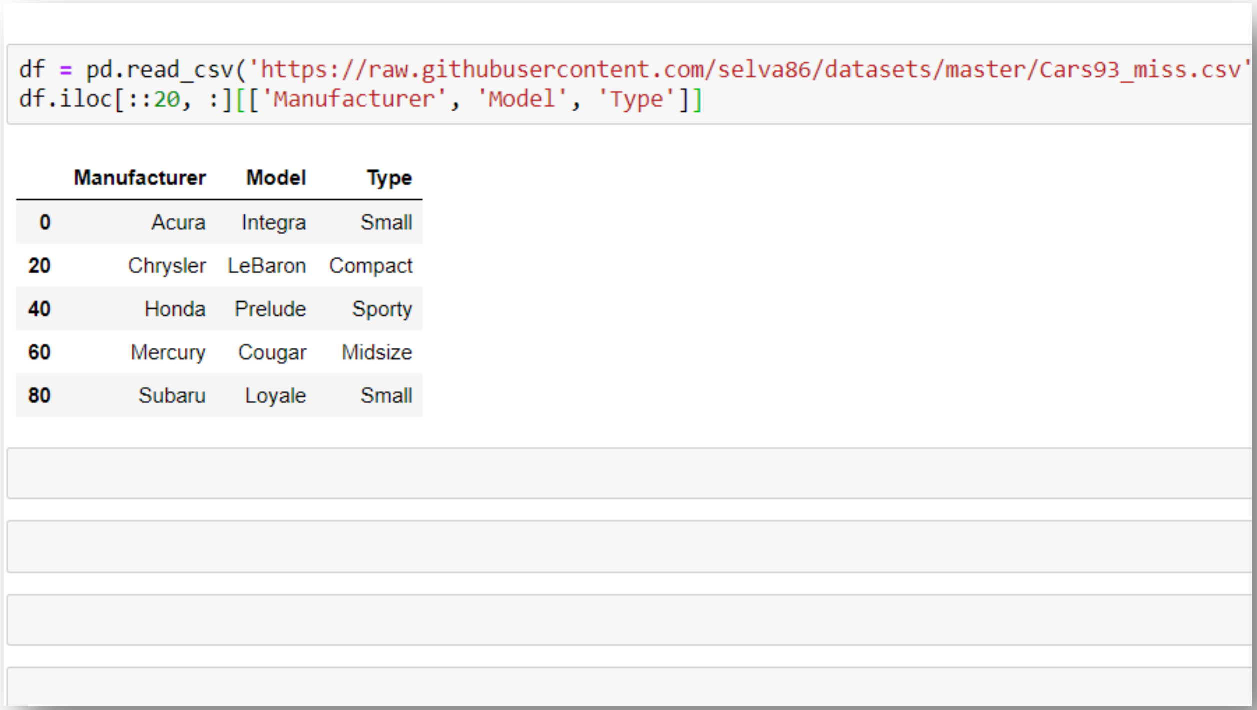
```
False
```

Q55. How to filter every nth row in a dataframe ?

A55. From the given dataframe , filter the ‘Manufacturer’, ‘Model’ and ‘type’ for every 20th row starting from 1st (row 0).

Code :

```
# Given dataframe

df=pd.read_csv(‘https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv’)

#Extraction of every 20th row data

df.iloc[::20, :][['Manufacturer', 'Model', 'Type']]
```



```
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv'
df.iloc[::20, :][['Manufacturer', 'Model', 'Type']]
```

	Manufacturer	Model	Type
0	Acura	Integra	Small
20	Chrysler	LeBaron	Compact
40	Honda	Prelude	Sporty
60	Mercury	Cougar	Midsize
80	Subaru	Loyale	Small

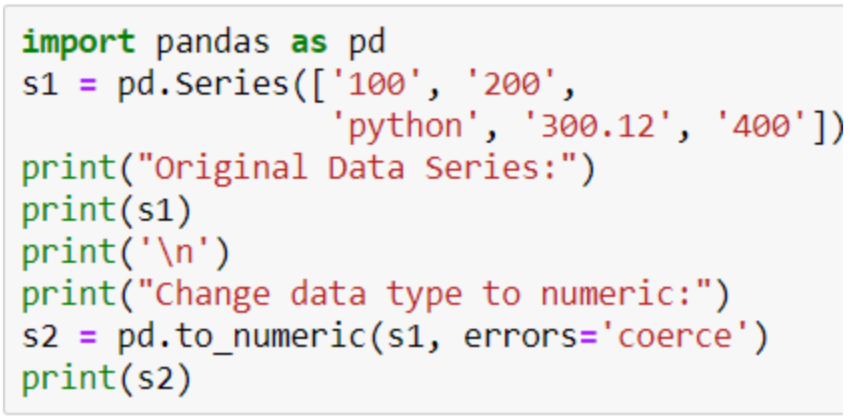
Q56. Write a Pandas program to change a data type (object to int) of a Column or a Series.

Sample Series:
Original Data Series:
0 100
1 200
2 python
3 300.12
4 400
dtype: object

A56. Code:

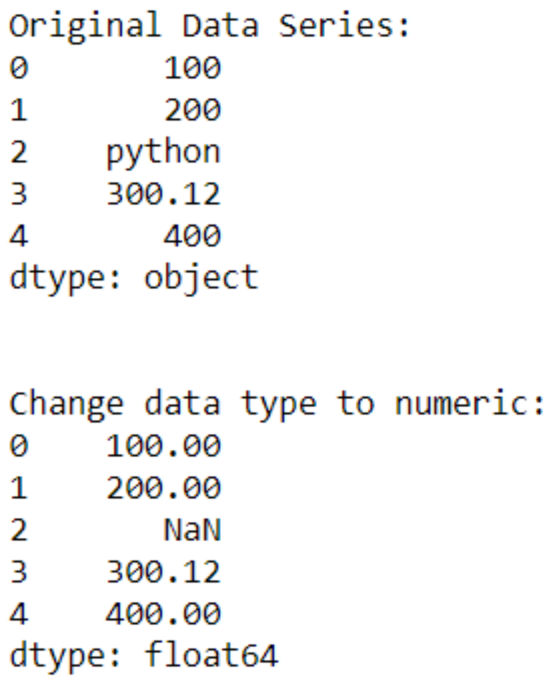
```
import pandas as pd
s1 = pd.Series(['100', '200', 'python', '300.12', '400'])
print("Original Data Series:")
print(s1)
print('\n')
print("Change the said data type to numeric:")
```

```
s2 = pd.to_numeric(s1, errors='coerce')
print(s2)
```



```
import pandas as pd
s1 = pd.Series(['100', '200',
                'python', '300.12', '400'])
print("Original Data Series:")
print(s1)
print('\n')
print("Change data type to numeric:")
s2 = pd.to_numeric(s1, errors='coerce')
print(s2)
```

Output:

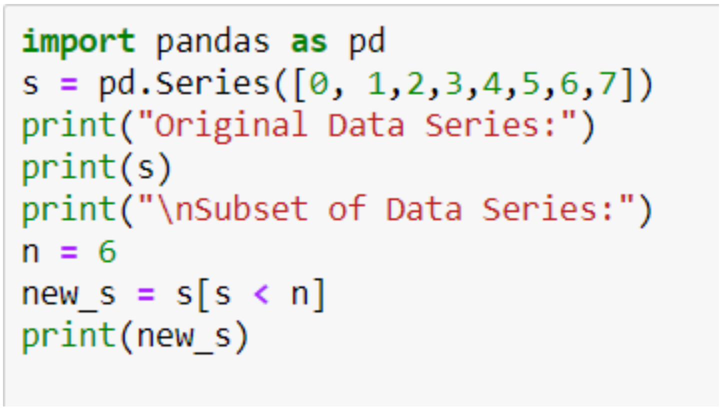


```
Original Data Series:
0        100
1        200
2     python
3     300.12
4        400
dtype: object


Change data type to numeric:
0    100.00
1    200.00
2       NaN
3    300.12
4    400.00
dtype: float64
```

Q57. Write a Pandas program to create a subset of a given series based on value and condition.
Original Data Series:
0 0
1 1
2 2
3 3

4 4
5 5
6 6
7 7
dtype: int64
Code:

```
import pandas as pd
s = pd.Series([0, 1,2,3,4,5,6,7])
print("Original Data Series:")
print(s)
print("\nSubset of the above Data Series:")
n = 6
new_s = s[s < n]
print(new_s)
```



```
import pandas as pd
s = pd.Series([0, 1,2,3,4,5,6,7])
print("Original Data Series:")
print(s)
print("\nSubset of Data Series:")
n = 6
new_s = s[s < n]
print(new_s)
```

Output:

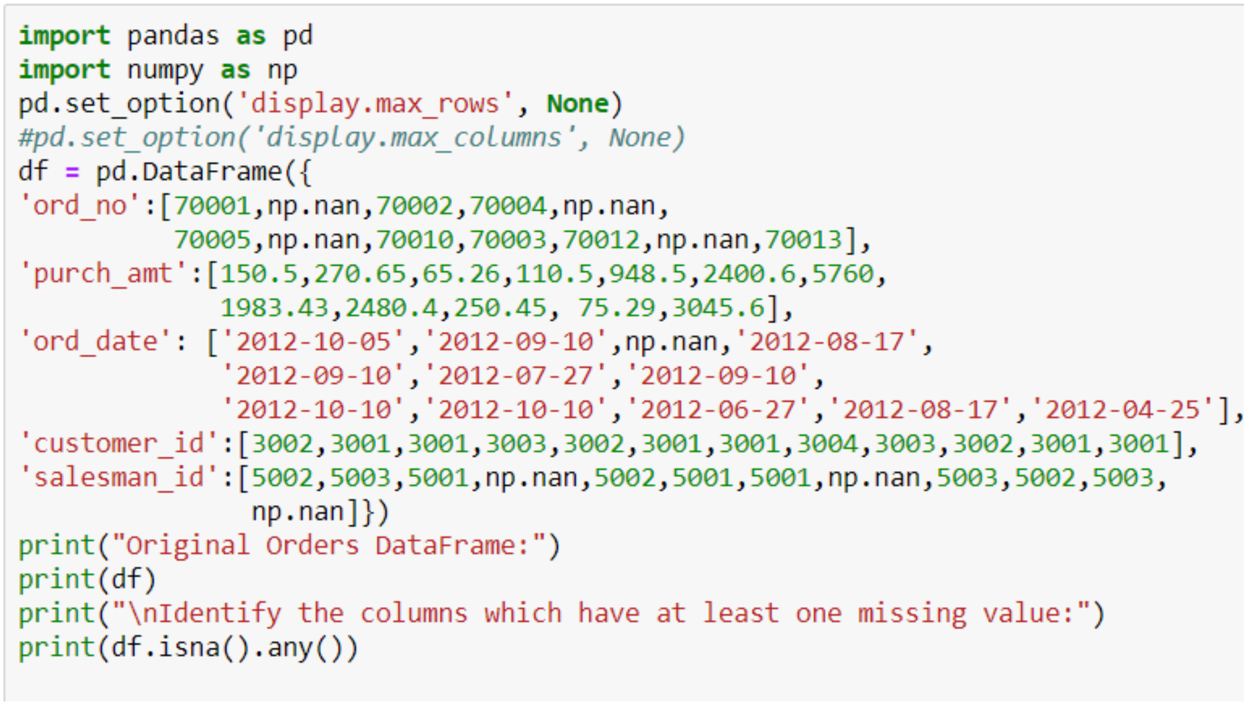
```
Original Data Series:
0    0
1    1
2    2
3    3
4    4
5    5
6    6
7    7
dtype: int64

Subset of Data Series:
0    0
1    1
2    2
3    3
4    4
5    5
dtype: int64
```

Q58. Write a pandas program to identify those columns of the given dataframe which have atleast 1 missing value.

A58. For filtering the missing values we can use isnull() function.

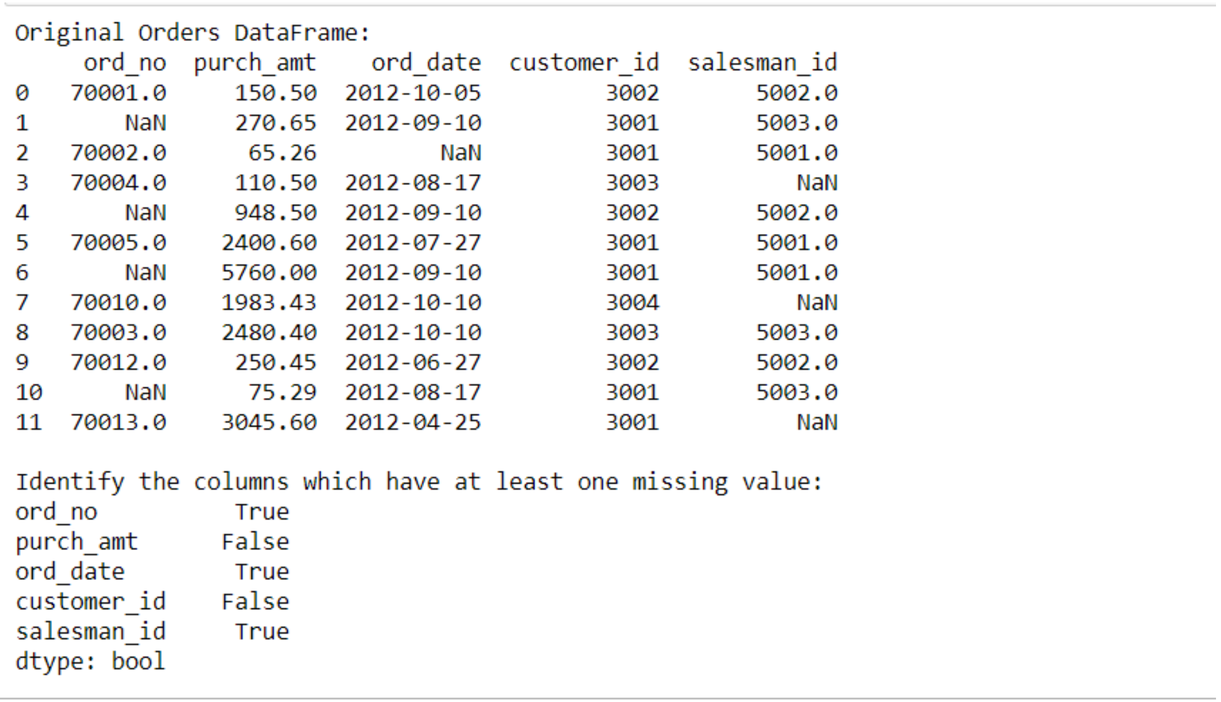
Code :

```
import pandas as pd
import numpy as np
pd.set_option('display.max_rows', None)
#pd.set_option('display.max_columns', None)

# Data Frame Creation
df = pd.DataFrame({
'ord_no':[70001,np.nan,70002,70004,np.nan,70005,np.nan,70010,70003,700
12,np.nan,70013],
'purch_amt':[150.5,270.65,65.26,110.5,948.5,2400.6,5760,1983.43,2480.4,25
0.45, 75.29,3045.6],
'ord_date': ['2012-10-05','2012-09-10',np.nan,'2012-08-17','2012-09-
10','2012-07-27','2012-09-10','2012-10-10','2012-10-10','2012-06-27','2012-
08-17','2012-04-25'],
'customer_id':[3002,3001,3001,3003,3002,3001,3001,3004,3003,3002,3001,
3001],
'salesman_id':[5002,5003,5001,np.nan,5002,5001,5001,np.nan,5003,5002,50
03,np.nan]})

# Print all dataframes
print("Original Orders DataFrame:")
print(df)
print("\nIdentify the columns which have at least one missing value:")
print(df.isna().any())
```



```
import pandas as pd
import numpy as np
pd.set_option('display.max_rows', None)
#pd.set_option('display.max_columns', None)
df = pd.DataFrame({
'ord_no':[70001,np.nan,70002,70004,np.nan,
          70005,np.nan,70010,70003,70012,np.nan,70013],
'purch_amt':[150.5,270.65,65.26,110.5,948.5,2400.6,5760,
             1983.43,2480.4,250.45, 75.29,3045.6],
'ord_date': ['2012-10-05','2012-09-10',np.nan,'2012-08-17',
             '2012-09-10','2012-07-27','2012-09-10',
             '2012-10-10','2012-10-10','2012-06-27','2012-08-17','2012-04-25'],
'customer_id':[3002,3001,3001,3003,3002,3001,3001,3004,3003,3002,3001,3001],
'salesman_id':[5002,5003,5001,np.nan,5002,5001,5001,np.nan,5003,5002,5003,
               np.nan]})
print("Original Orders DataFrame:")
print(df)
print("\nIdentify the columns which have at least one missing value:")
print(df.isna().any())
```

Output:



```
Original Orders DataFrame:
     ord_no  purch_amt    ord_date  customer_id  salesman_id
0   70001.0     150.50  2012-10-05         3002       5002.0
1       NaN     270.65  2012-09-10         3001       5003.0
2   70002.0      65.26         NaN         3001       5001.0
3   70004.0     110.50  2012-08-17         3003          NaN
4       NaN     948.50  2012-09-10         3002       5002.0
5   70005.0    2400.60  2012-07-27         3001       5001.0
6       NaN    5760.00  2012-09-10         3001       5001.0
7   70010.0    1983.43  2012-10-10         3004          NaN
8   70003.0    2480.40  2012-10-10         3003       5003.0
9   70012.0     250.45  2012-06-27         3002       5002.0
10      NaN      75.29  2012-08-17         3001       5003.0
11  70013.0    3045.60  2012-04-25         3001          NaN

Identify the columns which have at least one missing value:
ord_no          True
purch_amt      False
ord_date        True
customer_id    False
salesman_id     True
dtype: bool
```

Q59. Write a Pandas program to drop the rows where atleast one atleast one element is missing in a given dataframe.

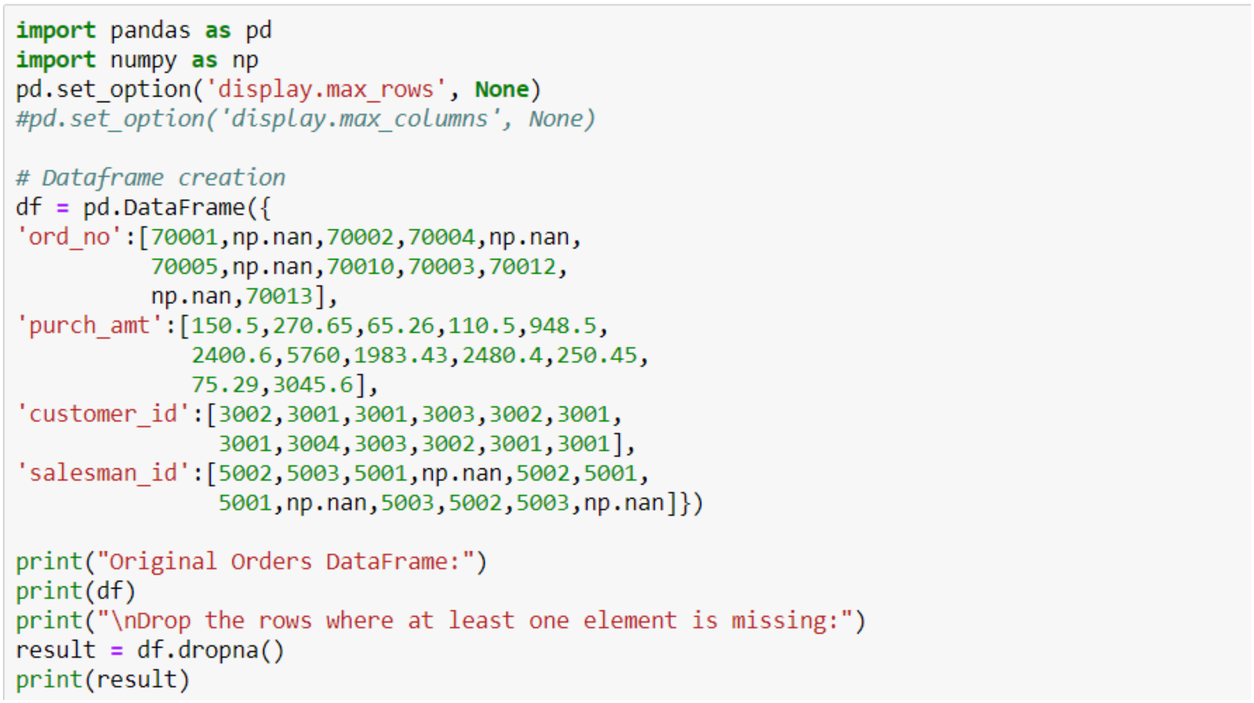
A59. We can use dropna() function. dropna() function is used to remove rows and columns with Null/Nan values. By default dropna() function returns a new dataframe and the original dataframe remains unchanged.

Syntax : DataFrame.dropna(self, axis=0, how='any', thresh=None, inplace = False)

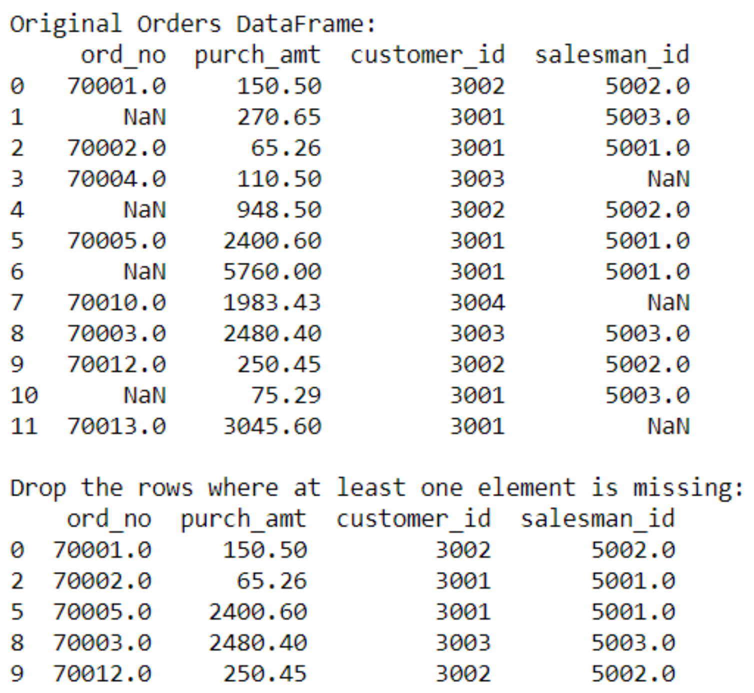
If inplace =True then, the change will be permanent in the dataframe.
axis=0 for row
axis =1 for columns.
Code :

```
import pandas as pd
import numpy as np
pd.set_option('display.max_rows', None)
#pd.set_option('display.max_columns', None)

# Dataframe creation
df = pd.DataFrame({
```

```
'ord_no':[70001,np.nan,70002,70004,np.nan,70005,np.nan,70010,70003,700
12,np.nan,70013],
'purch_amt':[150.5,270.65,65.26,110.5,948.5,2400.6,5760,1983.43,2480.4,25
0.45, 75.29,3045.6],
'customer_id':[3002,3001,3001,3003,3002,3001,3001,3004,3003,3002,3001,
3001],
'salesman_id':[5002,5003,5001,np.nan,5002,5001,5001,np.nan,5003,5002,50
03,np.nan]})

print("Original Orders DataFrame:")
print(df)
print("\nDrop the rows where at least one element is missing:")
result = df.dropna()
print(result)
```



```python
import pandas as pd
import numpy as np
pd.set_option('display.max_rows', None)
#pd.set_option('display.max_columns', None)

# Dataframe creation
df = pd.DataFrame({
'ord_no':[70001,np.nan,70002,70004,np.nan,
          70005,np.nan,70010,70003,70012,
          np.nan,70013],
'purch_amt':[150.5,270.65,65.26,110.5,948.5,
             2400.6,5760,1983.43,2480.4,250.45,
             75.29,3045.6],
'customer_id':[3002,3001,3001,3003,3002,3001,
               3001,3004,3003,3002,3001,3001],
'salesman_id':[5002,5003,5001,np.nan,5002,5001,
               5001,np.nan,5003,5002,5003,np.nan]})

print("Original Orders DataFrame:")
print(df)
print("\nDrop the rows where at least one element is missing:")
result = df.dropna()
print(result)
```



```
Original Orders DataFrame:
     ord_no  purch_amt  customer_id  salesman_id
0   70001.0     150.50         3002       5002.0
1       NaN     270.65         3001       5003.0
2   70002.0      65.26         3001       5001.0
3   70004.0     110.50         3003          NaN
4       NaN     948.50         3002       5002.0
5   70005.0    2400.60         3001       5001.0
6       NaN    5760.00         3001       5001.0
7   70010.0    1983.43         3004          NaN
8   70003.0    2480.40         3003       5003.0
9   70012.0     250.45         3002       5002.0
10      NaN      75.29         3001       5003.0
11  70013.0    3045.60         3001          NaN

Drop the rows where at least one element is missing:
    ord_no  purch_amt  customer_id  salesman_id
0  70001.0     150.50         3002       5002.0
2  70002.0      65.26         3001       5001.0
5  70005.0    2400.60         3001       5001.0
8  70003.0    2480.40         3003       5003.0
9  70012.0     250.45         3002       5002.0
```

Q60. What is pivot table and how can we create the pivot table using pandas? Explain with a suitable Example.

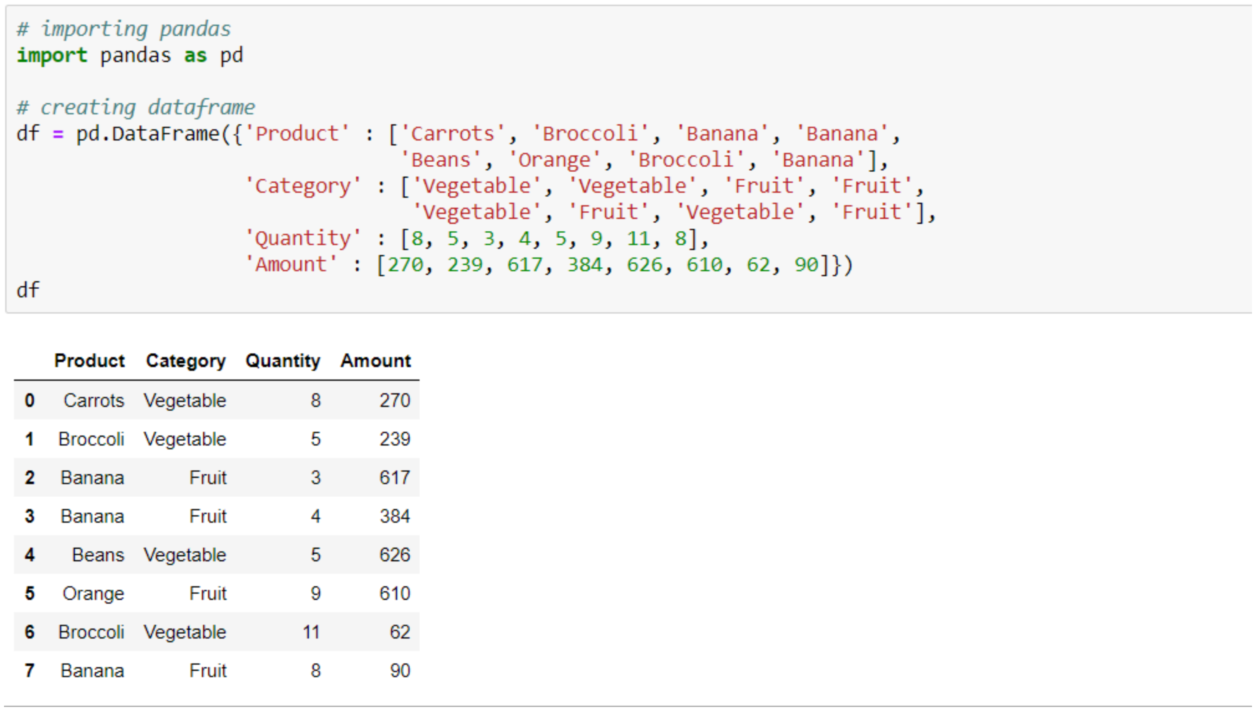
A60. Pivot table is a summary of the data. It is a part of data processing. This Summary in pivot table may include sum, mean, median or other statistical terms.
We can create pivot table using pandas. There is an inbuilt method in pandas named dataframe.pivot() with the help of which we can create the pivot table.

Syntax : dataframe.pivot(self, index=None, columns=None, values=None, aggfunc)
Parameters :
Parameters –
index: Column for making new frame's index.
columns: Column for new frame's columns.
values: Column(s) for populating new frame's values.
aggfunc: function, list of functions, dict, default numpy.mean

Let's understand with an example:

Code :

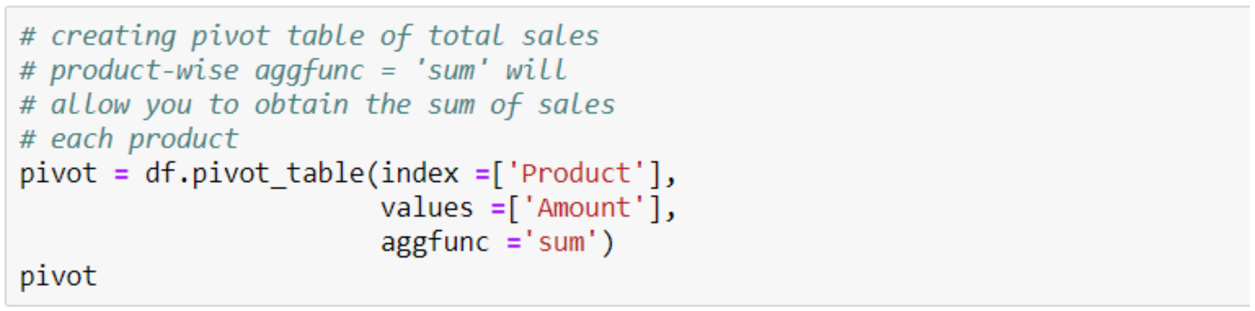
```
# importing pandas
import pandas as pd

# creating dataframe
df = pd.DataFrame({'Product' : ['Carrots', 'Broccoli', 'Banana', 'Banana',
                               'Beans', 'Orange', 'Broccoli', 'Banana'],
                  'Category' : ['Vegetable', 'Vegetable', 'Fruit', 'Fruit',
                               'Vegetable', 'Fruit', 'Vegetable', 'Fruit'],
                  'Quantity' : [8, 5, 3, 4, 5, 9, 11, 8],
                  'Amount' : [270, 239, 617, 384, 626, 610, 62, 90]})
df
```



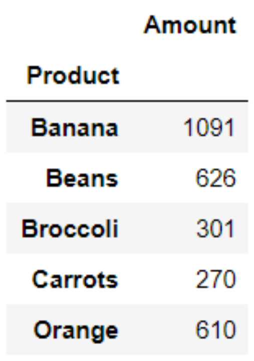
```
# importing pandas
import pandas as pd

# creating dataframe
df = pd.DataFrame({'Product' : ['Carrots', 'Broccoli', 'Banana', 'Banana',
                               'Beans', 'Orange', 'Broccoli', 'Banana'],
                  'Category' : ['Vegetable', 'Vegetable', 'Fruit', 'Fruit',
                                'Vegetable', 'Fruit', 'Vegetable', 'Fruit'],
                  'Quantity' : [8, 5, 3, 4, 5, 9, 11, 8],
                  'Amount' : [270, 239, 617, 384, 626, 610, 62, 90]})
df
```

	Product	Category	Quantity	Amount
0	Carrots	Vegetable	8	270
1	Broccoli	Vegetable	5	239
2	Banana	Fruit	3	617
3	Banana	Fruit	4	384
4	Beans	Vegetable	5	626
5	Orange	Fruit	9	610
6	Broccoli	Vegetable	11	62
7	Banana	Fruit	8	90

```
# creating pivot table of total sales
# product-wise aggfunc = 'sum' will
# allow you to obtain the sum of sales
# each product

pivot = df.pivot_table(index =['Product'],values =['Amount'],aggfunc ='sum')
pivot
```



```
# creating pivot table of total sales
# product-wise aggfunc = 'sum' will
# allow you to obtain the sum of sales
# each product
pivot = df.pivot_table(index =['Product'],
                       values =['Amount'],
                       aggfunc ='sum')
pivot
```



```
          Amount
Product
Banana      1091
Beans        626
Broccoli     301
Carrots      270
Orange       610
```

Q61. What method will you use to rename the index or columns of the pandas dataframe ?

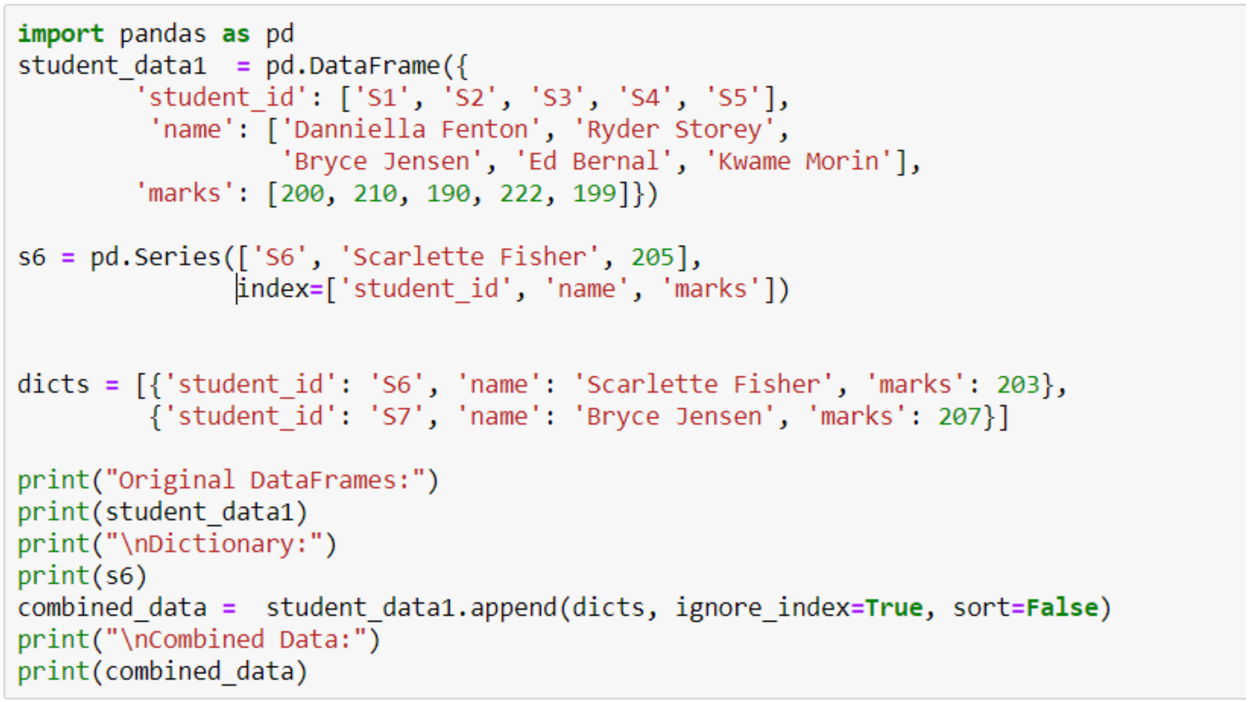
A61. We can use .rename() method to rename(give different values of columns) of Dataframe.

Q62. Write a Pandas program to append a list of dictionaries or series to a existing dataframe and display the combined data.

A62. We can use append method.
Code :

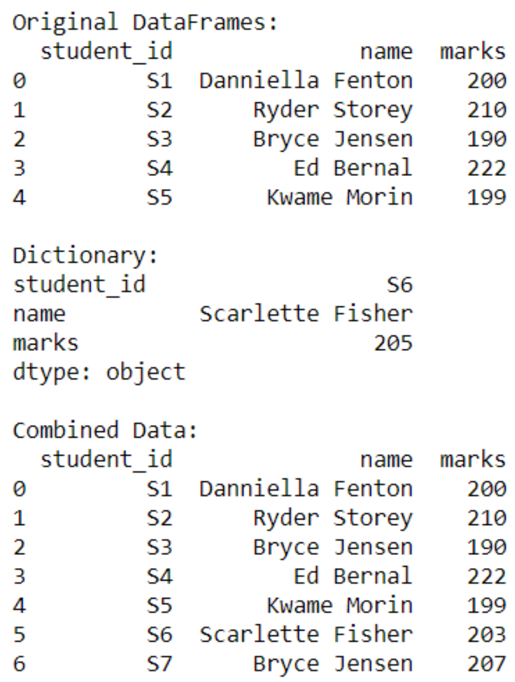
```
import pandas as pd

#  DataFrame Creation
student_data1  = pd.DataFrame({
                        'student_id': ['S1', 'S2', 'S3', 'S4', 'S5'],
                       'name': ['Danniella Fenton', 'Ryder Storey',  'Bryce Jensen', 'Ed
                                    Bernal', 'Kwame Morin'],
                        'marks': [200, 210, 190, 222, 199]})
```

```
s6 = pd.Series(['S6', 'Scarlette Fisher', 205], index=['student_id', 'name',
'marks'])


dicts = [{'student_id': 'S6', 'name': 'Scarlette Fisher', 'marks': 203},
        {'student_id': 'S7', 'name': 'Bryce Jensen', 'marks': 207}]

print("Original DataFrames:")
print(student_data1)
print("\nDictionary:")
print(s6)
combined_data =  student_data1.append(dicts, ignore_index=True, sort=False)
print("\nCombined Data:")
print(combined_data)
```



```
import pandas as pd
student_data1  = pd.DataFrame({
        'student_id': ['S1', 'S2', 'S3', 'S4', 'S5'],
         'name': ['Danniella Fenton', 'Ryder Storey',
              'Bryce Jensen', 'Ed Bernal', 'Kwame Morin'],
        'marks': [200, 210, 190, 222, 199]})

s6 = pd.Series(['S6', 'Scarlette Fisher', 205],
               index=['student_id', 'name', 'marks'])


dicts = [{'student_id': 'S6', 'name': 'Scarlette Fisher', 'marks': 203},
         {'student_id': 'S7', 'name': 'Bryce Jensen', 'marks': 207}]

print("Original DataFrames:")
print(student_data1)
print("\nDictionary:")
print(s6)
combined_data =  student_data1.append(dicts, ignore_index=True, sort=False)
print("\nCombined Data:")
print(combined_data)
```

Output :



```
Original DataFrames:
  student_id              name  marks
0         S1  Danniella Fenton    200
1         S2      Ryder Storey    210
2         S3      Bryce Jensen    190
3         S4         Ed Bernal    222
4         S5       Kwame Morin    199

Dictionary:
student_id                  S6
name          Scarlette Fisher
marks                      205
dtype: object

Combined Data:
  student_id              name  marks
0         S1  Danniella Fenton    200
1         S2      Ryder Storey    210
2         S3      Bryce Jensen    190
3         S4         Ed Bernal    222
4         S5       Kwame Morin    199
5         S6  Scarlette Fisher    203
6         S7      Bryce Jensen    207
```

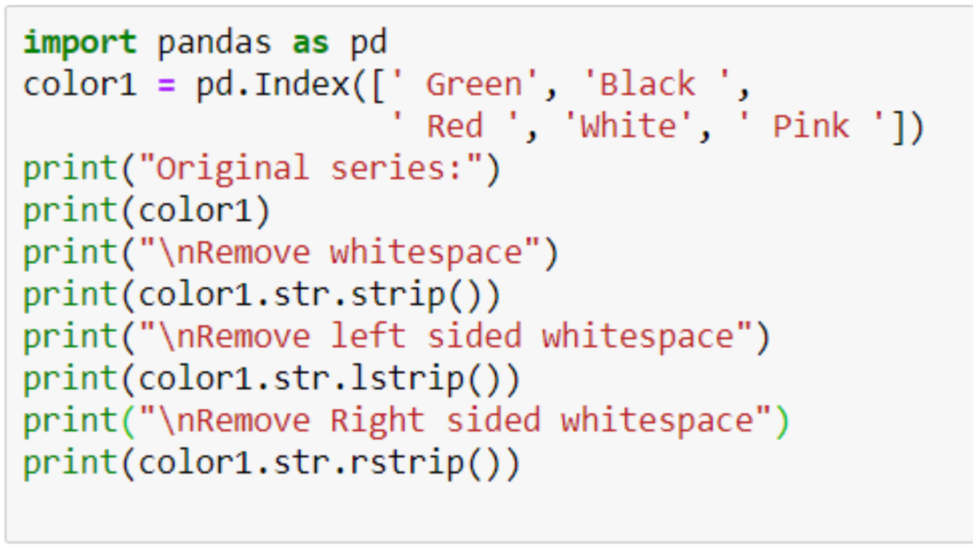
Q63. Write the Pandas program to remove whitespaces , right side whitespaces and left whitespaces of the string values of a given pandas series.

A63. We can use string methods (strip, lstrip, rstrip) with pandas dataframe.

Let us understand with an example.

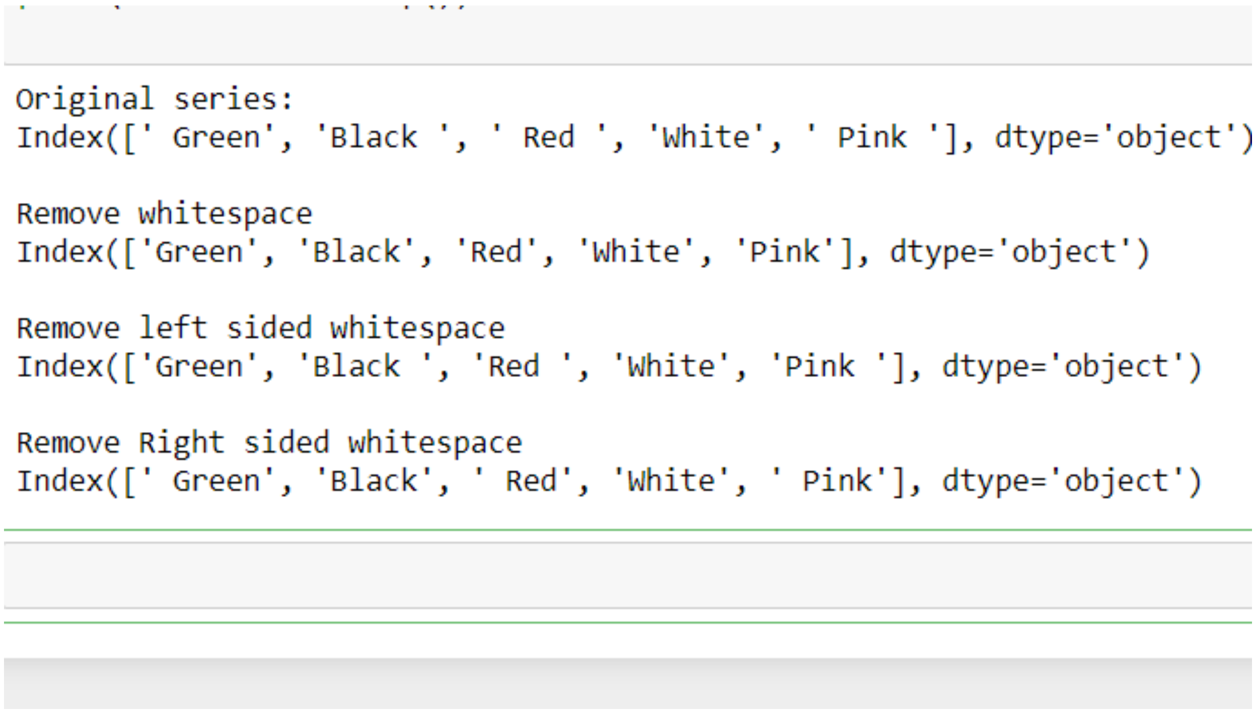
Code:

```
import pandas as pd

# dataframe creation
color1 = pd.Index([' Green', 'Black ', ' Red ', 'White', ' Pink '])

print("Original series:")
print(color1)
print("\nRemove whitespace")
print(color1.str.strip())
print("\nRemove left sided whitespace")
print(color1.str.lstrip())
print("\nRemove Right sided whitespace")
print(color1.str.rstrip())
```



```
import pandas as pd
color1 = pd.Index([' Green', 'Black ',
                   ' Red ', 'White', ' Pink '])
print("Original series:")
print(color1)
print("\nRemove whitespace")
print(color1.str.strip())
print("\nRemove left sided whitespace")
print(color1.str.lstrip())
print("\nRemove Right sided whitespace")
print(color1.str.rstrip())
```

Output :



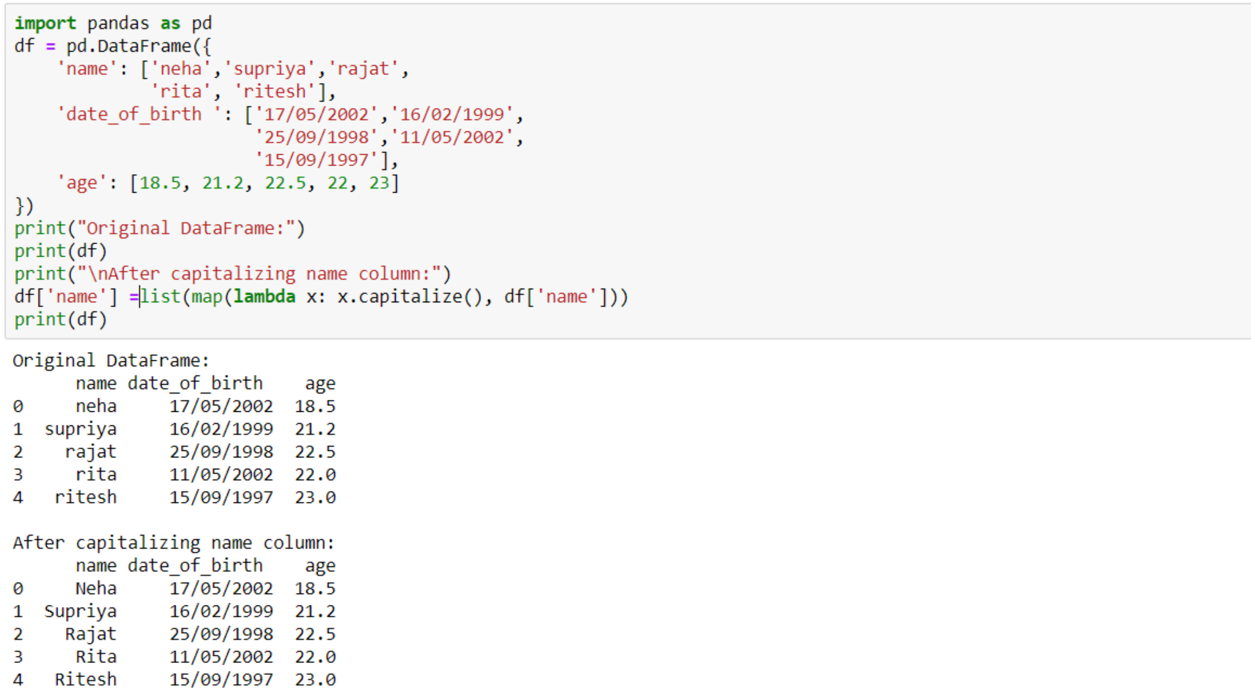
```
Original series:
Index([' Green', 'Black ', ' Red ', 'White', ' Pink '], dtype='object')

Remove whitespace
Index(['Green', 'Black', 'Red', 'White', 'Pink'], dtype='object')

Remove left sided whitespace
Index(['Green', 'Black ', 'Red ', 'White', 'Pink '], dtype='object')

Remove Right sided whitespace
Index([' Green', 'Black', ' Red', 'White', ' Pink'], dtype='object')
```

Q64. Write a Pandas program to capitalize all the string values of the specified column of the dataframe.

A64. We can do all of this operations on the dataframe. here we will use map function(for mapping string methods to dataframe).

Code :

```
import pandas as pd
df = pd.DataFrame({
      'name': ['neha','supriya','rajat', 'rita', 'ritesh'],
    'date_of_birth’:['17/05/2002','16/02/1999','25/09/1998','11/05/2002',
                                                '15/0 9/1997'],
     'age': [18.5, 21.2, 22.5, 22, 23]
                 })

print("Original DataFrame:")
print(df)
print("\nAfter capitalizing name column:")
df['name'] = list(map(lambda x: x.capitalize(), df['name']))
print(df)
```



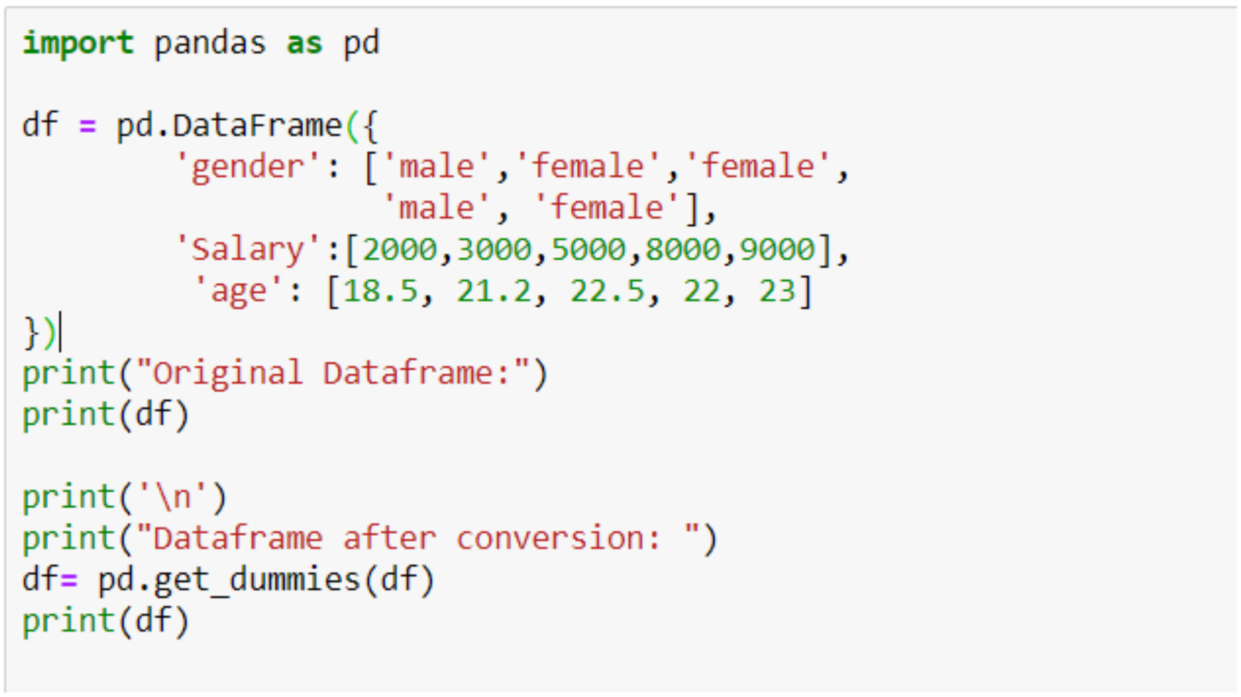
```
import pandas as pd
df = pd.DataFrame({
    'name': ['neha','supriya','rajat',
             'rita', 'ritesh'],
    'date_of_birth ': ['17/05/2002','16/02/1999',
                       '25/09/1998','11/05/2002',
                       '15/09/1997'],
    'age': [18.5, 21.2, 22.5, 22, 23]
})
print("Original DataFrame:")
print(df)
print("\nAfter capitalizing name column:")
df['name'] =list(map(lambda x: x.capitalize(), df['name']))
print(df)
```

```
Original DataFrame:
      name date_of_birth    age
0     neha     17/05/2002  18.5
1  supriya     16/02/1999  21.2
2    rajat     25/09/1998  22.5
3     rita     11/05/2002  22.0
4   ritesh     15/09/1997  23.0

After capitalizing name column:
      name date_of_birth    age
0     Neha     17/05/2002  18.5
1  Supriya     16/02/1999  21.2
2    Rajat     25/09/1998  22.5
3     Rita     11/05/2002  22.0
4   Ritesh     15/09/1997  23.0
```

Q65. What is a Categorical variable in Dataset and how to handle it using pandas?

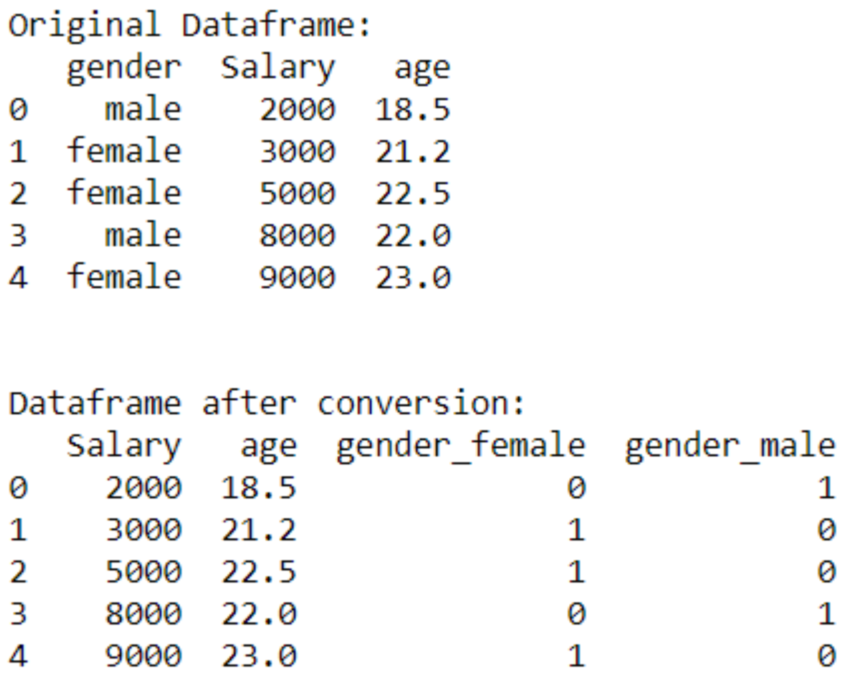
A65. A Categorical variable is one that has two or more categories.
Example : Black, brown, red
As we know Machine learning models are unable to handle categorical variables. So we have to convert them into a discreate variable.
Create Dummies:
Create Dummies for each category in the object(categorical) feature. The value for each row is 1 if that category is available in that row else 0. To create dummies we can use pandas get_dummies() function.

Code:
```
import pandas as pd

#Dataframe creation
df = pd.DataFrame({
	'gender': ['male','female','female','male', 'female'],
	'Salary':[2000,3000,5000,8000,9000],
	 'age': [18.5, 21.2, 22.5, 22, 23]
})
print("Original Dataframe:")
print(df)

print('\n')
print("Dataframe after conversion: ")
df= pd.get_dummies(df)
print(df)
```



```python
import pandas as pd

df = pd.DataFrame({
        'gender': ['male','female','female',
                   'male', 'female'],
        'Salary':[2000,3000,5000,8000,9000],
         'age': [18.5, 21.2, 22.5, 22, 23]
})
print("Original Dataframe:")
print(df)

print('\n')
print("Dataframe after conversion: ")
df= pd.get_dummies(df)
print(df)
```

Output:



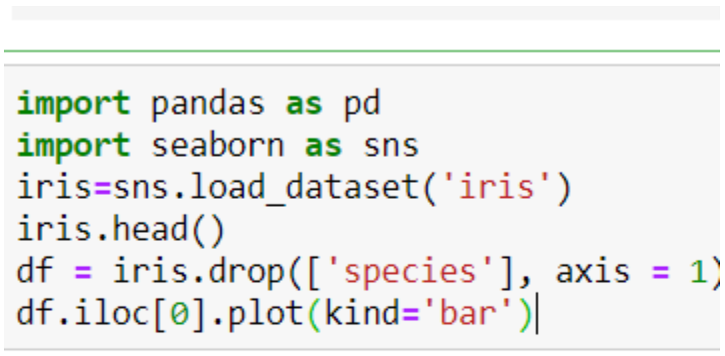
```
Original Dataframe:
   gender  Salary   age
0    male    2000  18.5
1  female    3000  21.2
2  female    5000  22.5
3    male    8000  22.0
4  female    9000  23.0


Dataframe after conversion:
   Salary   age  gender_female  gender_male
0    2000  18.5              0            1
1    3000  21.2              1            0
2    5000  22.5              1            0
3    8000  22.0              0            1
4    9000  23.0              1            0
```

Q66. How to plot bar graph using pandas ?

A66. Pandas is not a Data Visualization library but it is capable of creating basic plots. we can easily plot bar graph using pandas.

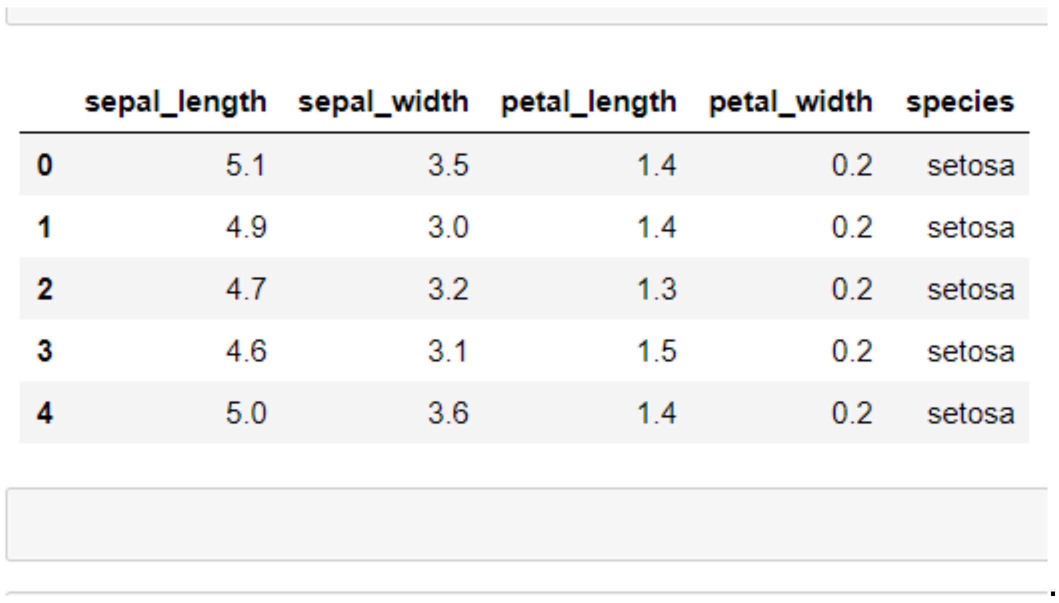
Code :

```
import pandas as pd
import seaborn as sns
iris=sns.load_dataset('iris')
iris.head()

# Code for bar graph

df = iris.drop(['species'], axis = 1)
df.iloc[0].plot(kind='bar')
```



```
import pandas as pd
import seaborn as sns
iris=sns.load_dataset('iris')
iris.head()
df = iris.drop(['species'], axis = 1)
df.iloc[0].plot(kind='bar')
```

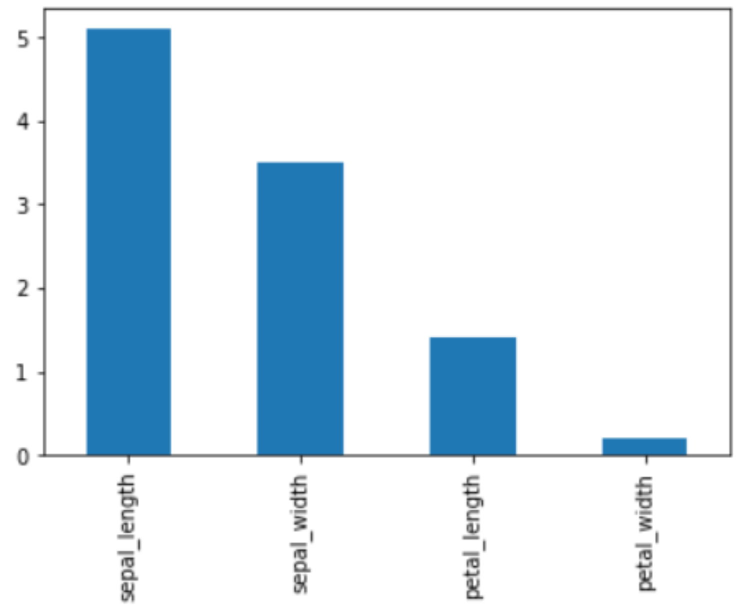
Dataframe:



	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Output:

```
<AxesSubplot:>
```



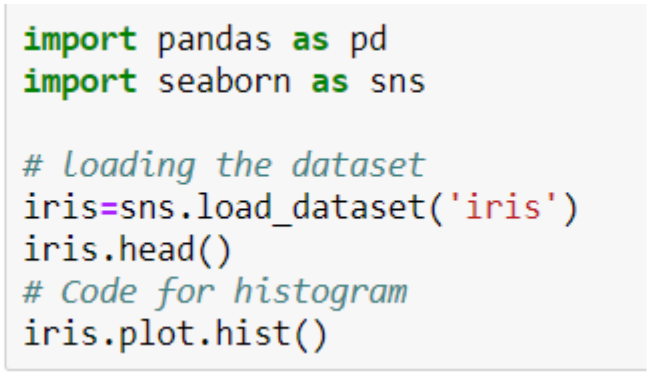
Q67. How to plot histograms using pandas?

A67. A histogram is a bar graph like representation of data.
The histogram is used for features whose values are numerical and measured

on an interval scale. It is generally used when dealing with large data sets (greater than 200 observations)

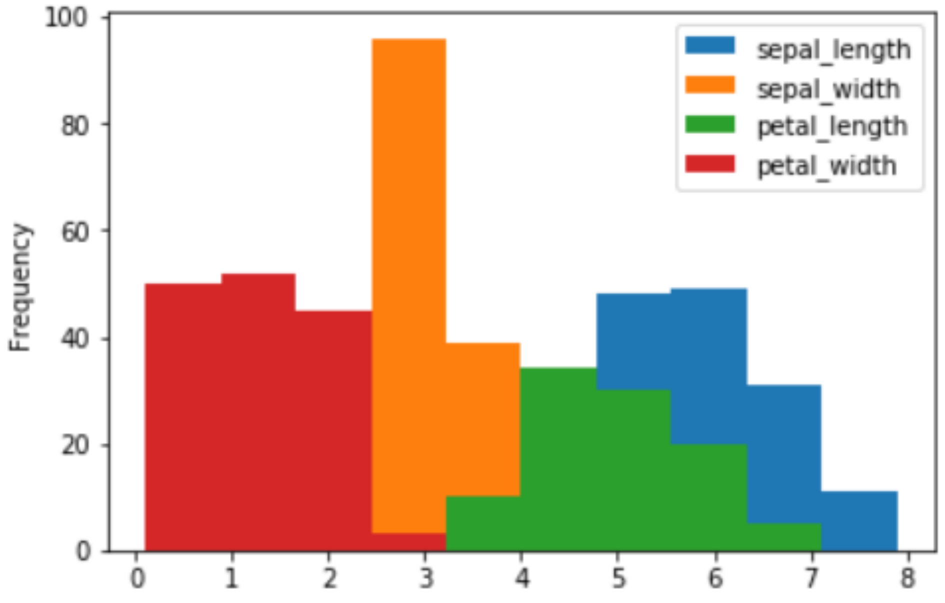
Code:

```
import pandas as pd
import seaborn as sns

# loading the dataset using seaborn library
iris=sns.load_dataset('iris')
iris.head()

# Code for histogram

iris.plot.hist()
```



```
import pandas as pd
import seaborn as sns

# loading the dataset
iris=sns.load_dataset('iris')
iris.head()
# Code for histogram
iris.plot.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2c1e9c18>
```



Q68. When , why and how you should reshape your pandas dataframe ?

A 68. Reshaping your dataframe is basically transforming it so that the resulting structure makes it more suitable for data analysis.

Q69. In pandas, Index values must be?
A. unique
B. hashable
C. Both A and B
D. None of the above

A69. C
Explanation: Index values must be unique and hashable, same length as data. Default np.arange(n) if no index is passed.

Q70. A panel is a ___ container of data
A. 1D
B. 2D

C. 3D
D. Infinite

A70. C
Explanation: A panel is a 3D container of data. The term Panel data is derived from econometrics and is partially responsible for the name pandas : pan(el)-da(ta)-s.

Q71. Which of the following takes a dict of dicts or a dict of array-like sequences and returns a DataFrame

A DataFrame.from_items
B. DataFrame.from_records
C. DataFrame.from_dict
D. All of the above

A71. A
Explanation: DataFrame.from_dict operates like the DataFrame constructor except for the orient parameter which is 'columns' by default.

Q72. What will be the output of the following code?

```
import pandas as pd
import numpy as np
s = pd.Series(np.random.randn(4))
print s.ndim
```

A72. B
Explanation: Returns the number of dimensions of the object. By definition, a Series is a 1D data structure, so it returns 1.

Q73. Which of the following indexing capabilities is used as a concise means of selecting data from a pandas object?

A. In
B. ix
C. ipy
D. iy

A73. B
Explanation: ix and reindex are 100% equivalent.

Q74. Which of the following makes use of pandas and returns data in a series or dataFrame?

A. pandaSDMX
B. freedapi
C. OutPy
D. Inpy

A74. B
Explanation: freedapi module requires a FRED API key that you can obtain for free on the FRED website.

Q75. Which of the following thing can be data in Pandas?
a) a python dict
b) an ndarray
c) a scalar value
d) all of the mentioned

A75. d
Explanation: The passed index is a list of axis labels.

Q76. Which function can read the dataset from a large text file?

A76. read_csv

Q77. Which function in the library of Pandas allows you to manipulate data and create new variables.

A77. Apply function

Q78. Which of these is an invalid writer function in Pandas?

a) to_clipboard
b) to_text
c) to_stata
d) to_msgpack

A78. to_text

Summary

This Article contains the most frequently asked interview questions of pandas. As we all know pandas is most important popular library in the field of data science and Machine learning. That's why it is most favourite topic for interviewers. This article will cover the whole concept of pandas starting from the basics to advance level of pandas. This article will surely help you when you are going for any Python Interview, Data Science Interview, and Data Analyst Interview. These are the common questions asked by almost every Multi-National Companies.

In this Article I have covered the all functions of pandas with explanation and several variety of questions. The level of questions is from basics to advance. This can be revision for an interviewee or else you can also start from these questions as all the questions are of easy to hard level which is easy to understand as well as the code is in the easiest way.

If you're interested in a Machine Learning or Data Science Domain then it is must to have a good knowledge of pandas. Pandas is an open-source Python library for Data Manipulation and Data Analytics works.