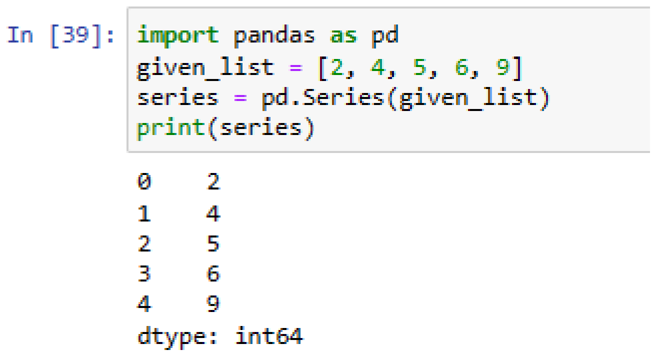
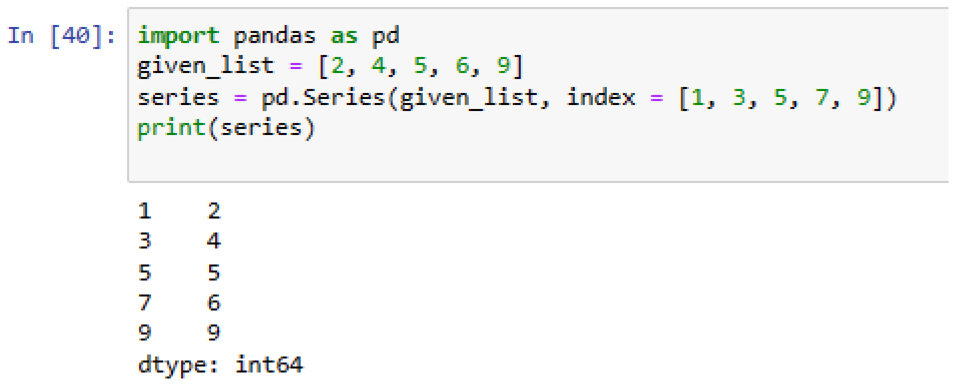
MACHINE LEARNING ASSIGNMENT-3

1. Given a list, output the corresponding pandas series.



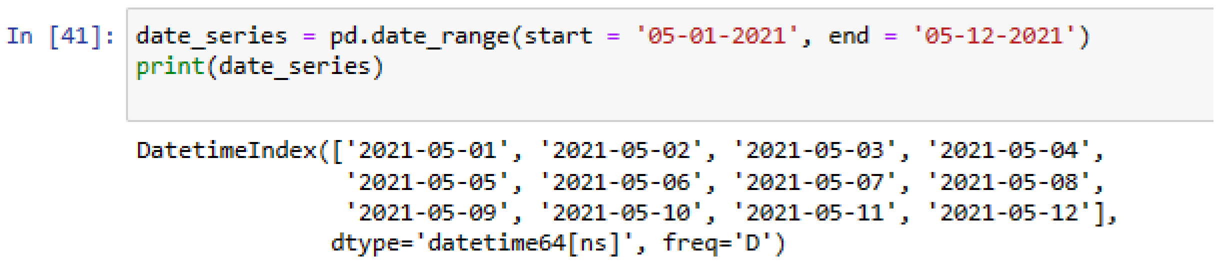
```
In [39]: import pandas as pd
         given_list = [2, 4, 5, 6, 9]
         series = pd.Series(given_list)
         print(series)

0    2
1    4
2    5
3    6
4    9
dtype: int64
```

2. Given a list, output the corresponding pandas series.



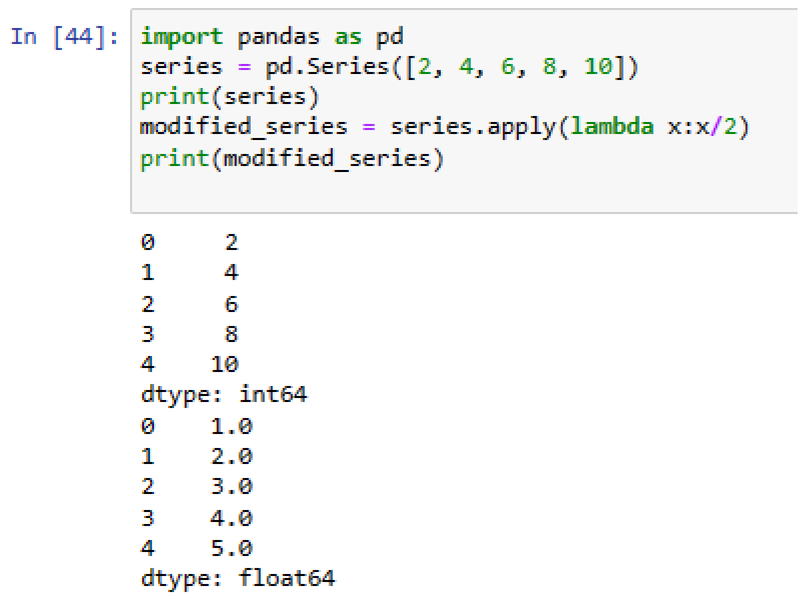
```
In [40]: import pandas as pd
         given_list = [2, 4, 5, 6, 9]
         series = pd.Series(given_list, index = [1, 3, 5, 7, 9])
         print(series)

1    2
3    4
5    5
7    6
9    9
dtype: int64
```

3. Generate the series of dates from 1st May, 2021 to 12th May, 2021 (both inclusive)



```
In [41]: date_series = pd.date_range(start = '05-01-2021', end = '05-12-2021')
         print(date_series)

DatetimeIndex(['2021-05-01', '2021-05-02', '2021-05-03', '2021-05-04',
               '2021-05-05', '2021-05-06', '2021-05-07', '2021-05-08',
               '2021-05-09', '2021-05-10', '2021-05-11', '2021-05-12'],
              dtype='datetime64[ns]', freq='D')
```

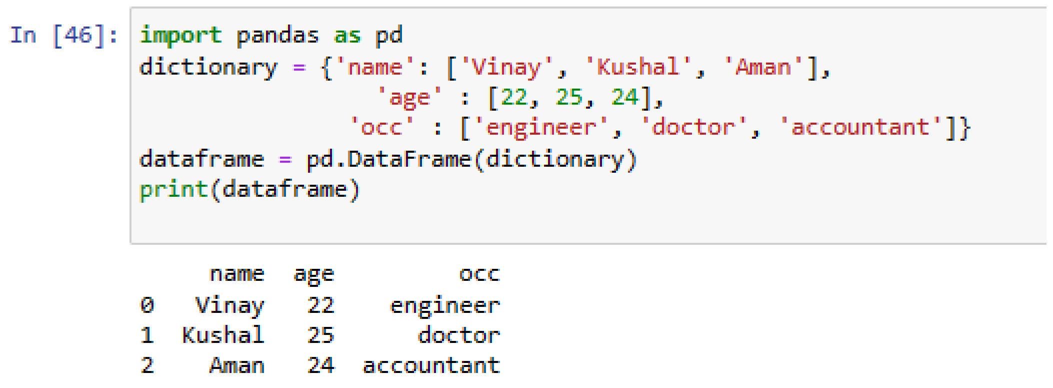
4. Apply the function, f(x) = x/2 on each and every element of a given pandas series



```
In [44]: import pandas as pd
         series = pd.Series([2, 4, 6, 8, 10])
         print(series)
         modified_series = series.apply(lambda x:x/2)
         print(modified_series)
```

```
0     2
1     4
2     6
3     8
4    10
dtype: int64
0    1.0
1    2.0
2    3.0
3    4.0
4    5.0
dtype: float64
```

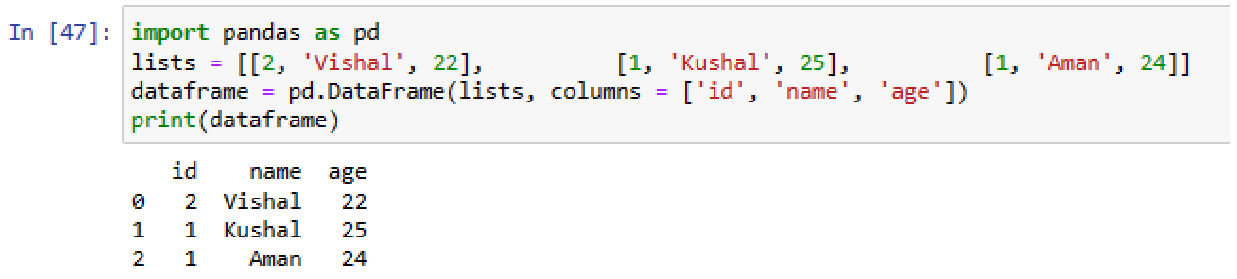
5. Given a dictionary, convert it into corresponding dataframe and display it



```
In [46]: import pandas as pd
         dictionary = {'name': ['Vinay', 'Kushal', 'Aman'],
                          'age' : [22, 25, 24],
                         'occ' : ['engineer', 'doctor', 'accountant']}
         dataframe = pd.DataFrame(dictionary)
         print(dataframe)
```

```
     name  age         occ
0   Vinay   22    engineer
1  Kushal   25      doctor
2    Aman   24  accountant
```

6. Given a 2D List, convert it into corresponding dataframe and display it.



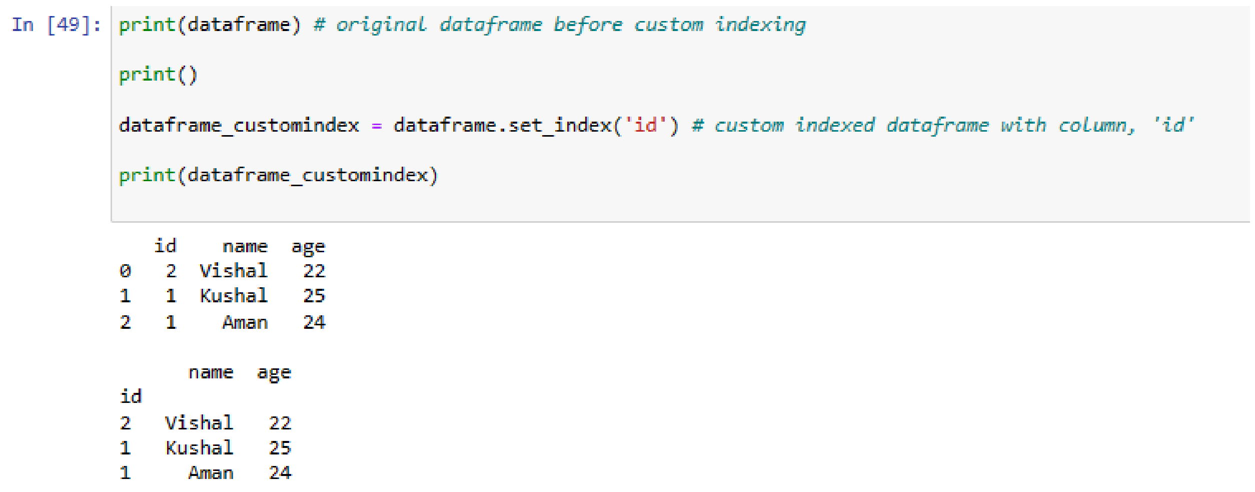
```
In [47]: import pandas as pd
         lists = [[2, 'Vishal', 22],          [1, 'Kushal', 25],          [1, 'Aman', 24]]
         dataframe = pd.DataFrame(lists, columns = ['id', 'name', 'age'])
         print(dataframe)
```

```
   id    name  age
0   2  Vishal   22
1   1  Kushal   25
2   1    Aman   24
```

7. Given a CSV file, read it into a dataframe and display it.

```
dataframe = pd.read_csv('data.csv')

print(dataframe)

   id    name  age         occ
0   1   Vinay   22    engineer
1   2  Kushal   25      doctor
2   3    Aman   24  accountant
```

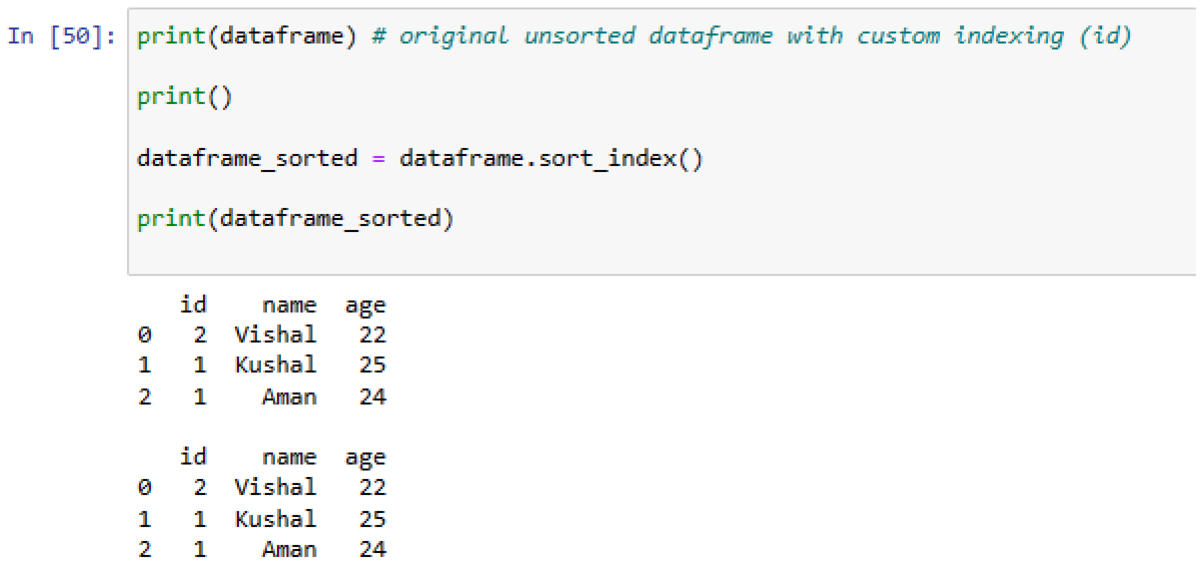
8. Given a dataframe, change the index of a dataframe from the default indexes to a particular column.



```
In [49]: print(dataframe) # original dataframe before custom indexing

         print()

         dataframe_customindex = dataframe.set_index('id') # custom indexed dataframe with column, 'id'

         print(dataframe_customindex)
```

```
   id    name  age
0   2  Vishal   22
1   1  Kushal   25
2   1    Aman   24

      name  age
id
2   Vishal   22
1   Kushal   25
1     Aman   24
```

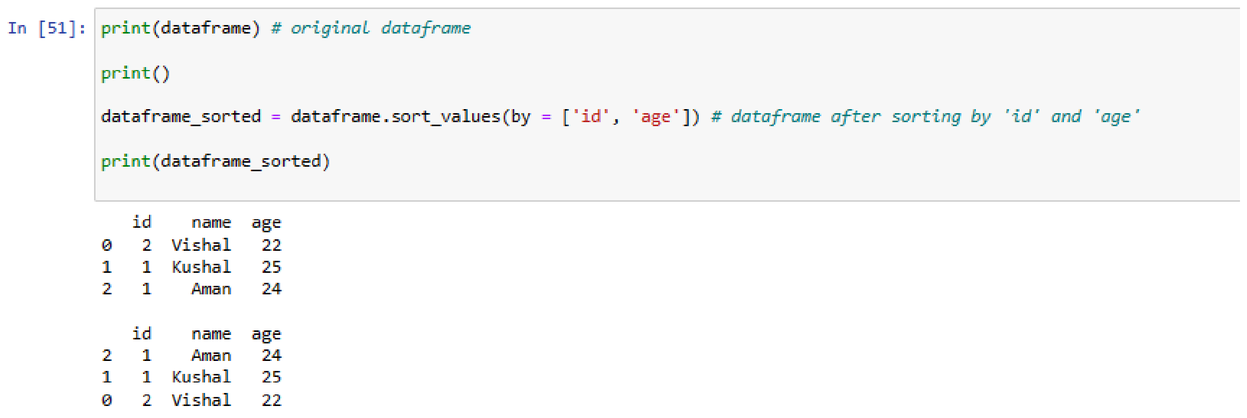
9. Given a dataframe (say, with custom indexing), sort it by it's index.



```
In [50]: print(dataframe) # original unsorted dataframe with custom indexing (id)

         print()

         dataframe_sorted = dataframe.sort_index()

         print(dataframe_sorted)
```

```
   id    name  age
0   2  Vishal   22
1   1  Kushal   25
2   1    Aman   24

   id    name  age
0   2  Vishal   22
1   1  Kushal   25
2   1    Aman   24
```

10. Given a dataframe, sort it by multiple columns.



```
In [51]: print(dataframe) # original dataframe

         print()

         dataframe_sorted = dataframe.sort_values(by = ['id', 'age']) # dataframe after sorting by 'id' and 'age'

         print(dataframe_sorted)
```

```
   id    name  age
0   2  Vishal   22
1   1  Kushal   25
2   1    Aman   24

   id    name  age
2   1    Aman   24
1   1  Kushal   25
0   2  Vishal   22
```

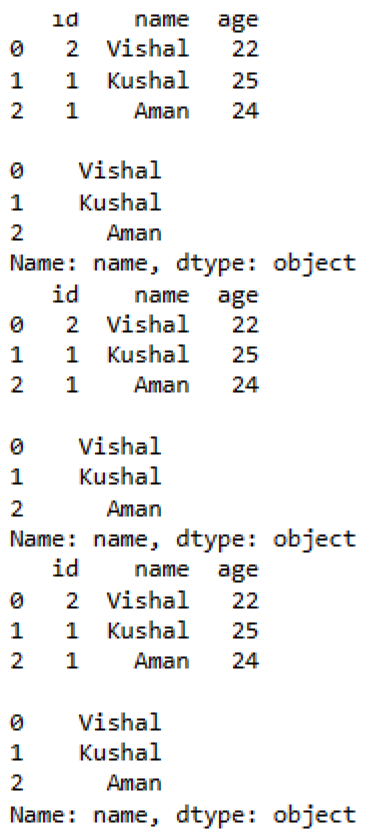
11. Given a dataframe with custom indexing, convert and it to default indexing and display it.

```
In [52]: print(dataframe_customindex) # printing the original dataframe with custom indexing

         print()

         dataframe = dataframe_customindex.reset_index()

         print(dataframe) # printing the dataframe with default indexes
```

```
      name  age
id
2   Vishal   22
1   Kushal   25
1     Aman   24

   id    name  age
0   2  Vishal   22
1   1  Kushal   25
2   1    Aman   24
```

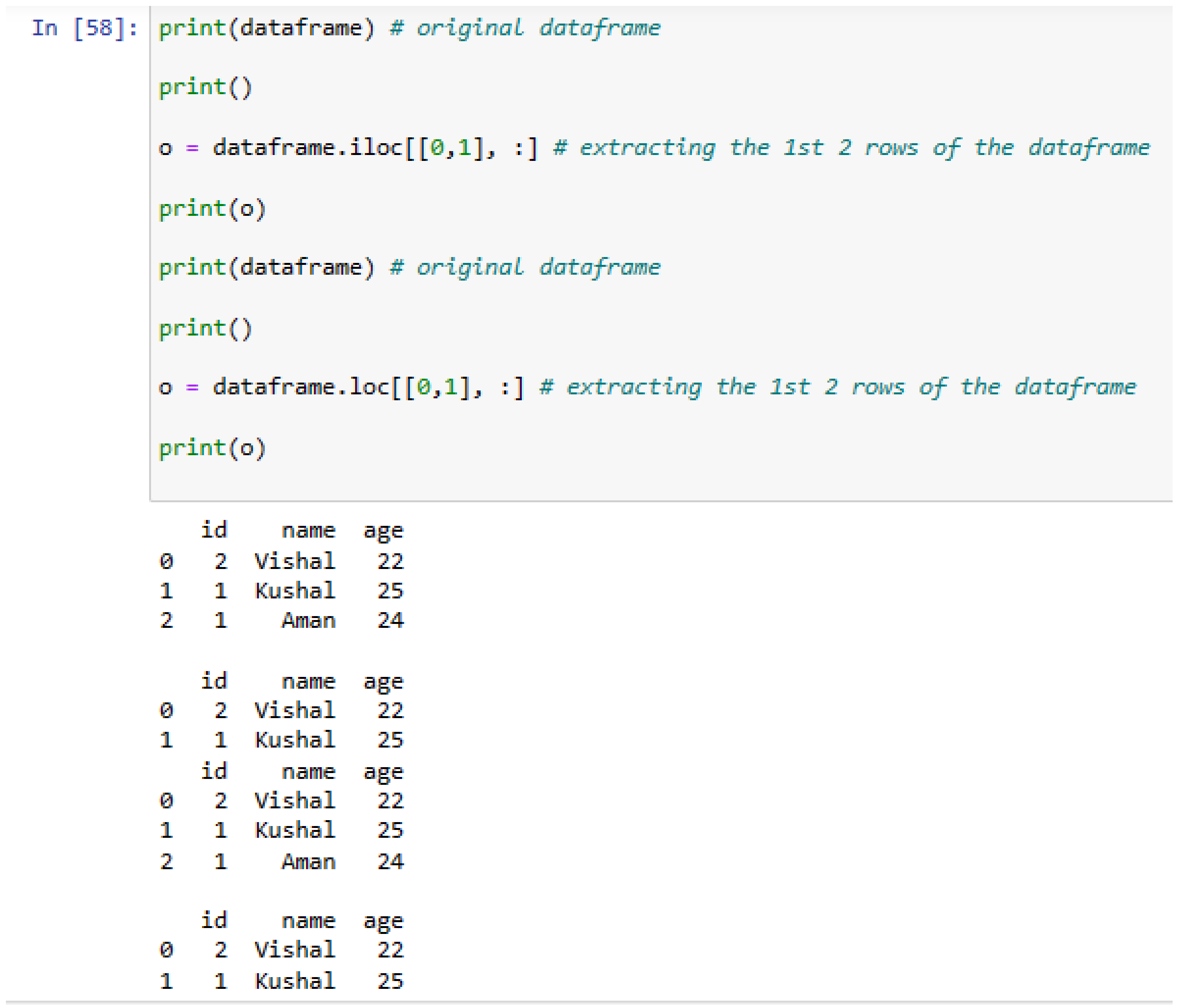
12. Given a dataframe, select a particular column and display it

```
In [56]: print(dataframe) # original dataframe

         print()

         o = dataframe['name'] # extracting the column 'name'

         print(o)

         print(dataframe) # original dataframe

         print()

         o = dataframe.iloc[:,1] # extracting the column 'name'

         print(o)

         print(dataframe) # original dataframe

         print()

         o = dataframe.loc[:,'name'] # extracting the column 'name'

         print(o)
```



```
   id    name  age
0   2  Vishal   22
1   1  Kushal   25
2   1    Aman   24

0    Vishal
1    Kushal
2      Aman
Name: name, dtype: object
   id    name  age
0   2  Vishal   22
1   1  Kushal   25
2   1    Aman   24

0    Vishal
1    Kushal
2      Aman
Name: name, dtype: object
   id    name  age
0   2  Vishal   22
1   1  Kushal   25
2   1    Aman   24

0    Vishal
1    Kushal
2      Aman
Name: name, dtype: object
```

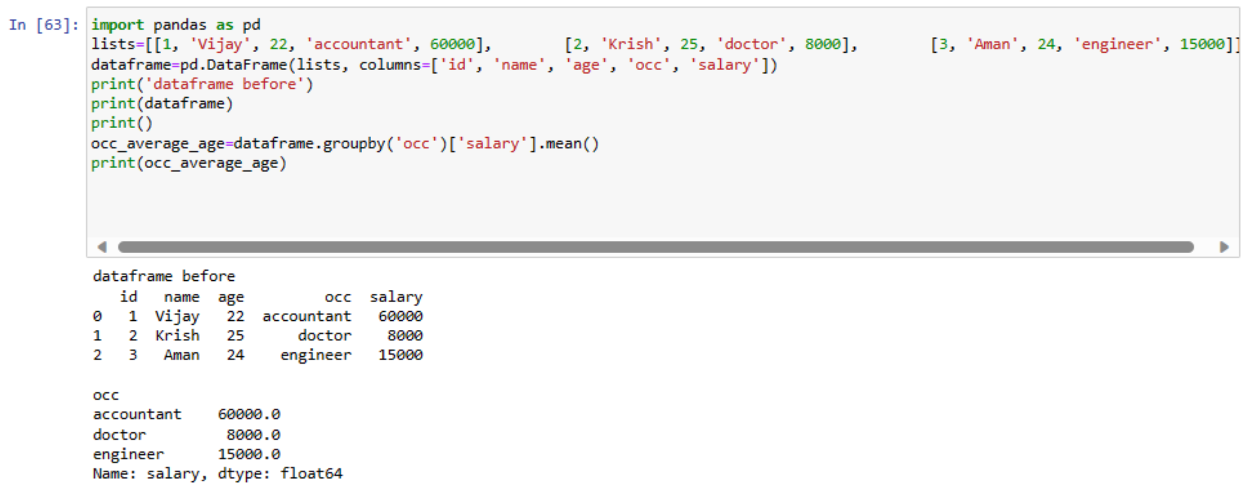
13. Given a dataframe, select first 2 rows and output them.



```
In [58]: print(dataframe) # original dataframe

         print()

         o = dataframe.iloc[[0,1], :] # extracting the 1st 2 rows of the dataframe

         print(o)

         print(dataframe) # original dataframe

         print()

         o = dataframe.loc[[0,1], :] # extracting the 1st 2 rows of the dataframe

         print(o)
```

```
   id    name  age
0   2  Vishal   22
1   1  Kushal   25
2   1    Aman   24

   id    name  age
0   2  Vishal   22
1   1  Kushal   25
   id    name  age
0   2  Vishal   22
1   1  Kushal   25
2   1    Aman   24

   id    name  age
0   2  Vishal   22
1   1  Kushal   25
```

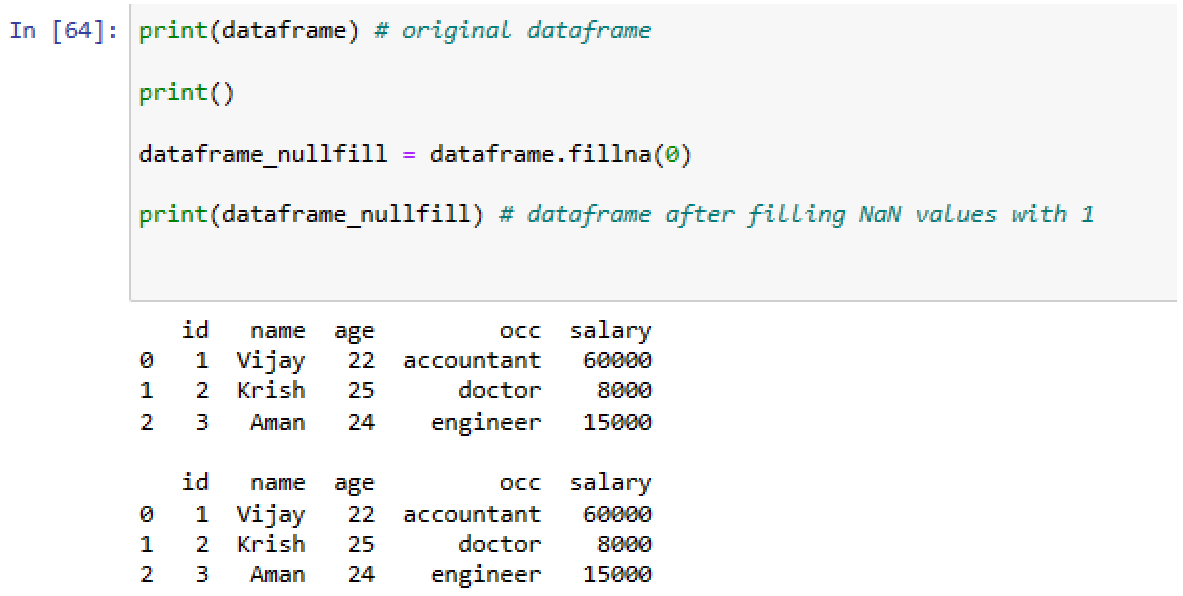
15. Given is a dataframe showing name, occupation, salary of people. Find the average salary per occupation.



```
In [63]: import pandas as pd
         lists=[[1, 'Vijay', 22, 'accountant', 60000],        [2, 'Krish', 25, 'doctor', 8000],        [3, 'Aman', 24, 'engineer', 15000]]
         dataframe=pd.DataFrame(lists, columns=['id', 'name', 'age', 'occ', 'salary'])
         print('dataframe before')
         print(dataframe)
         print()
         occ_average_age=dataframe.groupby('occ')['salary'].mean()
         print(occ_average_age)
```

```
dataframe before
   id   name  age         occ  salary
0   1  Vijay   22  accountant   60000
1   2  Krish   25      doctor    8000
2   3   Aman   24    engineer   15000

occ
accountant    60000.0
doctor         8000.0
engineer      15000.0
Name: salary, dtype: float64
```

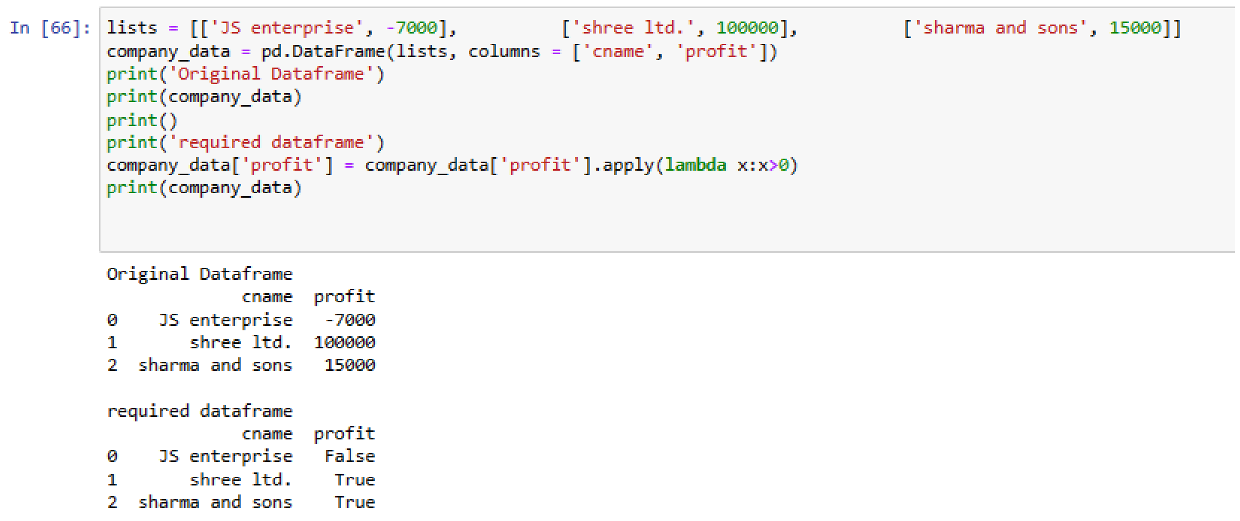
16. Given a dataframe with NaN Values, fill the NaN values with 0



```
In [64]: print(dataframe) # original dataframe

         print()

         dataframe_nullfill = dataframe.fillna(0)

         print(dataframe_nullfill) # dataframe after filling NaN values with 1
```

```
   id   name  age         occ  salary
0   1  Vijay   22  accountant   60000
1   2  Krish   25      doctor    8000
2   3   Aman   24    engineer   15000

   id   name  age         occ  salary
0   1  Vijay   22  accountant   60000
1   2  Krish   25      doctor    8000
2   3   Aman   24    engineer   15000
```

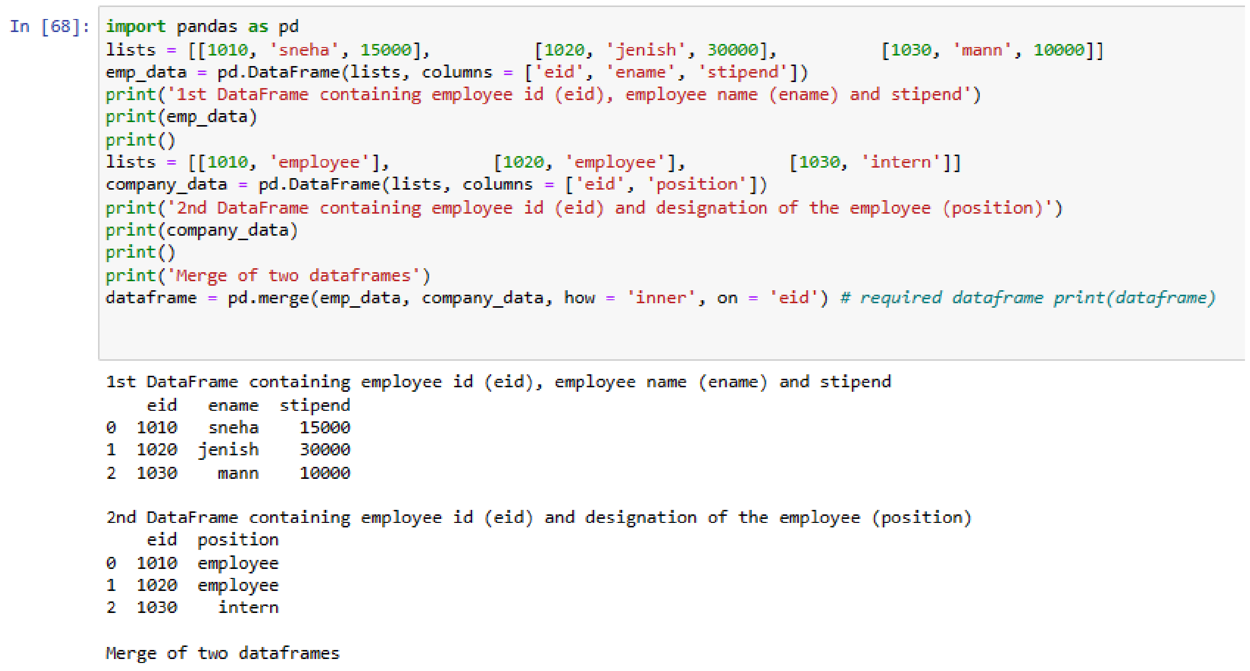
17. Given is a dataframe showing Company Names (cname) and corresponding Profits (profit). Convert the values of Profit column such that values in it greater than 0 are set to True and the rest are set to False.



```
In [66]: lists = [['JS enterprise', -7000],          ['shree ltd.', 100000],          ['sharma and sons', 15000]]
         company_data = pd.DataFrame(lists, columns = ['cname', 'profit'])
         print('Original Dataframe')
         print(company_data)
         print()
         print('required dataframe')
         company_data['profit'] = company_data['profit'].apply(lambda x:x>0)
         print(company_data)
```

```
Original Dataframe
             cname  profit
0    JS enterprise   -7000
1       shree ltd.  100000
2  sharma and sons   15000

required dataframe
             cname  profit
0    JS enterprise   False
1       shree ltd.    True
2  sharma and sons    True
```

18. Given are 2 dataframes, with one dataframe containing Employee ID (eid), Employee Name (ename) and Stipend (stipend) and the other dataframe containing Employee ID (eid) and designation of the employee (designation). Output the Dataframe containing Employee ID (eid), Employee Name (ename), Stipend (stipend) and Position (position).



```
In [68]: import pandas as pd
         lists = [[1010, 'sneha', 15000],          [1020, 'jenish', 30000],          [1030, 'mann', 10000]]
         emp_data = pd.DataFrame(lists, columns = ['eid', 'ename', 'stipend'])
         print('1st DataFrame containing employee id (eid), employee name (ename) and stipend')
         print(emp_data)
         print()
         lists = [[1010, 'employee'],          [1020, 'employee'],          [1030, 'intern']]
         company_data = pd.DataFrame(lists, columns = ['eid', 'position'])
         print('2nd DataFrame containing employee id (eid) and designation of the employee (position)')
         print(company_data)
         print()
         print('Merge of two dataframes')
         dataframe = pd.merge(emp_data, company_data, how = 'inner', on = 'eid') # required dataframe print(dataframe)
```

```
1st DataFrame containing employee id (eid), employee name (ename) and stipend
    eid   ename  stipend
0  1010   sneha    15000
1  1020  jenish    30000
2  1030    mann    10000

2nd DataFrame containing employee id (eid) and designation of the employee (position)
    eid  position
0  1010  employee
1  1020  employee
2  1030    intern

Merge of two dataframes
```

19. Given a dataframe, output the non-null count and data-type for every column

```
In [69]: print(dataframe)
         print()
         print(dataframe.info())
```

```
    eid   ename  stipend  position
0  1010   sneha    15000  employee
1  1020  jenish    30000  employee
2  1030    mann    10000    intern

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   eid       3 non-null      int64 
 1   ename     3 non-null      object
 2   stipend   3 non-null      int64 
 3   position  3 non-null      object
dtypes: int64(2), object(2)
memory usage: 228.0+ bytes
None
```

20. Given a dataframe, generate the statistical summary of all the numerical features present in it

```
In [70]: print(dataframe) # the dataframe

         print()

         print(dataframe.describe())
```

```
    eid   ename  stipend  position
0  1010   sneha    15000  employee
1  1020  jenish    30000  employee
2  1030    mann    10000    intern

          eid       stipend
count     3.0      3.000000
mean   1020.0  18333.333333
std      10.0  10408.329997
min    1010.0  10000.000000
25%    1015.0  12500.000000
50%    1020.0  15000.000000
75%    1025.0  22500.000000
max    1030.0  30000.000000
```