|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | # What will be output of following  import numpy as np  import pandas as pd  obj1=pd.Series(np.linspace(1,10,4))  print(obj1) | | | | | | | | | | | | | | | | | | | | | | | |
|  | (a) | | 0 1.0  1 4.0  2 7.0  3 10.0  dtype: float64 | | | | | (b) | 0 1.0  1 5.5  2 10.0  dtype: float64 | | (c) | | | | | error | | | (d) | | | 0 2.0  1 5.0  2 9.0  dtype: float64 | | |
| Ans | **(a)** | | **0 1.0**  **1 4.0**  **2 7.0**  **3 10.0**  **dtype: float64** | | | | |  |  | |  | | | | |  | | |  | | |  | | |
| 2 | # What will be output of following  import numpy as np  import pandas as pd  lst1=[110.0,np.NaN,43.0,140]  lst2=[110.0,np.NaN,43.0,140]  ser1=pd.Series(lst1)  ser2=pd.Series(data=lst2,index=['A','B','C','D'])  print(ser1==ser2) | | | | | | | | | | | | | | | | | | | | | | | |
|  | (a) | | True | | | | | (b) | Error | | (c) | | | False | | | | (d) | | | None of the above | | | |
| Ans | **(b)** | | **Error** | | | | |  |  | |  | | |  | | | |  | | |  | | | |
| 3 | How can you change the index of the series , state with example and output ? | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | import numpy as np  import pandas as pd  Lst=[100,200,300,400,500]  ind=[2,3,4,5,6]  ser=pd.Series(data=Lst,index=ind)  print(ser)  ser.index=[20,30,40,50,60]  print(ser) | | | | | | | | | | | | | | OUTPUT  2 100  3 200  4 300  5 400  6 500  dtype: int64  20 100  30 200  40 300  50 400  60 500  dtype: int64 | | | | | | | | | |
| 4 | # Replace highlighted statement with correct statement so that it produce output  import numpy as np  import pandas as pd  lst1=[110.0,np.NaN,43.0,140]  lst2=[110.0,np.NaN,43.0,140]  ser1=pd.Series(lst1)  ser2=pd.Series(data=lst2,index=[‘A’,’B’,’C’,’D’])  **print(ser1==ser2) # Replace this statement** | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | ser1.equals(ser2) | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | How will you create series using dictionary ? | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | import numpy as np  import pandas as pd  data=dict({"sale":3,"purchase":2,"marketing":5})  print(data)  ser=pd.Series(data)  print(ser) | | | | | | | | | | | | | | OUTPUT  {'sale': 3, 'purchase': 2, 'marketing': 5}  sale 3  purchase 2  marketing 5  dtype: int64 | | | | | | | | | |
| 5 | Can you insert any value say 150 at index 0 by writing ser[0]=150  import numpy as np  import pandas as pd  Lst=[100,200,300,400,500]  ind=[2,3,4,5,6]  ser=pd.Series(data=Lst,index=ind)  print(ser) | | | | | | | | | | | | | | | | | | | | | | | |
|  | (a) | | | **Yes** | | (b) | | | No we can not insert at index 0 because it is starting from index 2 | | | | (c) | | | | It will produce size error | | | | | (d) | | No we can not insert because series are immutable . |
| 6 | # Write a statement that will find out number of total not null values in following series.  import numpy as np  import pandas as pd  ser=pd.Series([2,4,6,8,np.NaN])  print(ser) | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | ser.hasnans | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | How will you make series using arange function ? state with example ? | | | | | | | | | | | | | | | | | | | | | | | |
| 8. | # pandas series  import numpy as np  import pandas as pd  obj1=pd.Series(np.arange(10,4,-2))  print(obj1) | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | # **Write missing statement to produce** # **following output**  import numpy as np  import pandas as pd  Lst=[2,4,6,8,10]  Lst2=[20,40,60,80,100]  ser=pd.Series(data=Lst2,index=['a','b','c','d','e'])  print(ser)  **// Missing statement**  print(ser2) | | | | | | | | | | | | **OUTPUT**  a 20  b 40  c 60  d 80  e 100  dtype: int64  1 NaN  2 NaN  c 60.0  d 80.0  e 100.0  dtype: float64 | | | | | | | | | | | |
| Ans | **ser2=ser.reindex([1,2,'c','d','e'])** | | | | | | | | | | | | | | | | | | | | | | | |
|  | # Write missing statement to produce # following output  import numpy as np  import pandas as pd  Lst=[10,20,30,40]  data=Lst  ser=pd.Series(data)  print(ser)  **//Missing statement** | | | | | | | | | | | **OUTPUT**  **0 10**  **1 20**  **2 30**  **3 40**  **dtype: int64**  **0 20**  **1 40**  **2 60**  **3 80**  **dtype: int64** | | | | | | | | | | | | |
| Ans | print(ser\*2) | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| Q | Make a series with list and display the values in series those are greater than 50 . | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | # Filter operation  import numpy as np  import pandas as pd  Lst2=[20,40,60,80,100]  ser2=pd.Series(data=Lst2)  print(ser2[ser2>50]) | | | | | | | | | | | **OUTPUT**  2 60  3 80  4 100  dtype: int64 | | | | | | | | | | | | |
| Q | # What will be the output of following  import numpy as np  import pandas as pd  Lst=[100,200,300,400,500]  ind=[2,3,4,5,6]  ser=pd.Series(data=Lst,index=ind)  ser[0:5:2]=-5  print(ser) | | | | | | | | | | | **OUTPUT**  2 -5  3 200  4 -5  5 400  6 -5  dtype: int64 | | | | | | | | | | | | |
| Q | # What will be the outpt of following  import numpy as np  import pandas as pd  Lst=[2,4,6,8]  ser=pd.Series(data=Lst\*2)  print(ser.tail(3)) | | | | | | | | | | | **OUTPUT**  5 4  6 6  7 8  dtype: int64 | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| Q | # Write the missing statement to produce  following output  import numpy as np  import pandas as pd  lst1=[110.0,np.NaN,43.0,140]  ser1=pd.Series(lst1)  print("\nser1 = ")  **// Missing statement** | | | | | | | | | | | **OUTPUT**  ser1 =  0 110.0  2 43.0  3 140.0  dtype: float64 | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| Q | Create a following data frame using 2 D Dictionary ?  **2018 2019 2020**  **Sales 450 490 450**  **Purchase 600 610 100**  **Production 700 700 700**  **Marketing 400 500 550** | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | #Making DataFrame by 2 D Dictionary object  import numpy as np  import pandas as pd  Year\_2018={'Sales':450,'Purchase':600,'Production':700,'Marketing':400}  Year\_2019={'Sales':490,'Purchase':610,'Production':700,'Marketing':500}  Year\_2020={'Sales':450,'Purchase':100,'Production':700,'Marketing':550}  Turn\_over={2018:Year\_2018,2019:Year\_2019,2020:Year\_2020}  df1=pd.DataFrame(Turn\_over)  print(df1) | | | | | | | | | | | | | | | | | | | | | | | |
| Q | Create following data frame by using 2 D array .  0 1 2 3  0 **11 21 43 56**  1 **14 35 16 57** | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | # By using 2D nd Array  import numpy as np  import pandas as pd  arr=np.array([[11,21,43,56],[14,35,16,57]])  df=pd.DataFrame(arr)  print(df) | | | | | | | | | | | | | | | | | | | | | | | |
| Q | What is the usage of dataframe describe() function . | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | To know the count , mean , std , min , max , 25 , 50 75 percentile . | | | | | | | | | | | | | | | | | | | | | | | |
| Q | # Write the missing statement to generate following output  **import numpy as np**  **import pandas as pd**  **Total\_Sale=pd.Series([8609099,8360000,3390209,6789030],\**  **index=['Sales','Purchasing','Marketing','Production'])**  **Total\_Expense=pd.Series([7810927,6850878,2264631,\**  **5280634],index=['Sales','Purchasing','Marketing','Production'])**  **profit=Total\_Sale-Total\_Expense**  **dict2={'January':Total\_Sale,'February':Total\_Expense,'March':profit}**  **dft2=pd.DataFrame(dict2,columns=['January','February','March'])**  // **missing statement** | | | | | | | | | | | | | | | | | | | | | | | |
|  | **OUTPUT** | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | print(dft2[['February','January']]) # Notice two brackets here | | | | | | | | | | | | | | | | | | | | | | | |
| Q | ‘’’See the output , in second print the value of February is chaged to 100 for Sales , write the missing statement to produce that output’’’  import numpy as np  import pandas as pd  Total\_Sale=pd.Series([8609099,8360000,3390209,6789030],\  index=['Sales','Purchasing','Marketing','Production'])  Total\_Expense=pd.Series([7810927,6850878,2264631,\  5280634],index=['Sales','Purchasing','Marketing','Production'])  profit=Total\_Sale-Total\_Expense  dict2={'January':Total\_Sale,'February':Total\_Expense,'March':profit}  dft2=pd.DataFrame(dict2,columns=['January','February','March'])  print(dft2)  // Missing Statement  print(dft2) | | | | | | | | | | | | | | | | | | | | | | | |
|  | **OUTPUT** | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | dft2.February['Sales']=100 | | | | | | | | | | | | | | | | | | | | | | | |
|  | ‘’’ See the output and write the missing statement to delete the coloumn april and may .  ‘’’  import numpy as np  import pandas as pd  Total\_Sale=pd.Series([8609099,8360000,3390209,6789030],\  index=['Sales','Purchasing','Marketing','Production'])  Total\_Expense=pd.Series([7810927,6850878,2264631,\  5280634],index=['Sales','Purchasing','Marketing','Production'])  profit=Total\_Sale-Total\_Expense  dict2={'January':Total\_Sale,'February':Total\_Expense,'March':profit}  dft2=pd.DataFrame(dict2,columns=['January','February','March'])  print(dft2)  df3=dft2.assign(april=[10000,20000,30000,40000],may=[50000,60000,70000,80000])  print(df3) | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | df4=df3.drop(['april','may'] , axis=1)  print(df4) | | | | | | | | | | | | | | | | | | | | | | | |
| Q | # Write the missing statement to produce the following output  import numpy as np  import pandas as pd  Total\_Sale=pd.Series([8609099,8360000,3390209,6789030],\  index=['Sales','Purchasing','Marketing','Production'])  Total\_Expense=pd.Series([7810927,6850878,2264631,\  5280634],index=['Sales','Purchasing','Marketing','Production'])  profit=Total\_Sale-Total\_Expense  dict2={'January':Total\_Sale,'February':Total\_Expense,'March':profit}  dft2=pd.DataFrame(dict2,columns=['January','February','March'])  print(dft2)  **// Missing Statement** | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | print(dft2['January']) # Printing Individual column | | | | | | | | | | | | | | | | | | | | | | | |
|  | ‘’’Write the missing statement to produce following output  '''  import numpy as np  import pandas as pd  Total\_Sale=pd.Series([8609099,8360000,3390209,6789030,5678769,9876787],\  index=['Sales','Purchasing','Marketing','Production','Accounts','TechSupport'])  Total\_Expense=pd.Series([7810927,6850878,2264631,\  5280634,4678769,8797655],index=['Sales','Purchasing','Marketing','Production','Accounts','TechSupport'])  profit=Total\_Sale-Total\_Expense  commission=profit\*.1  donation=profit\*.05  tax=profit\*.4  dict2={'Sale':Total\_Sale,'Expense':Total\_Expense,'Profit':profit,'commission':commission,'donation':donation,'tax':tax}  dft2=pd.DataFrame(dict2,columns=['Sale','Expense','Profit','commission','donation','tax'])  **// Missing Statement**  print(dft2) | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | dft2.loc[:,'NP'] = [456778,678987,567884,568789,656444,565547] | | | | | | | | | | | | | | | | | | | | | | | |
|  | ''' Write the missing statements so that following output may be achieved  '''  import numpy as np  import pandas as pd  Total\_Sale=pd.Series([8609099,8360000,3390209,6789030,5678769,9876787],\  index=['Sales','Purchasing','Marketing','Production','Accounts','TechSupport'])  Total\_Expense=pd.Series([7810927,6850878,2264631,\  5280634,4678769,8797655],index=['Sales','Purchasing','Marketing','Production','Accounts','TechSupport'])  profit=Total\_Sale-Total\_Expense  commission=profit\*.1  donation=profit\*.05  tax=profit\*.4  dict2={'Sale':Total\_Sale,'Expense':Total\_Expense,'Profit':profit,'commission':commission,'donation':donation,'tax':tax}  dft2=pd.DataFrame(dict2,columns=['Sale','Expense','Profit','commission','donation','tax'])  dft2.loc[:,'NP'] = [456778,678987,567884,568789,656444,565547]  dft2.at[:,'QP'] = [4568,6787,5684,8789,6444,5547]  // Missing Statement  print(dft3)  // Missing Statement  print(dft3) | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | dft3=dft2.assign(HP = [456,678,564,889,644,547])  del dft3['HP'] | | | | | | | | | | | | | | | | | | | | | | | |
| Q | Make a program using dictionary to produce following output - | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | # Creating of dataframe having boolean indexes  import pandas as pd  # dictionary of lists  dict = {'StuName':["Ravi", "Suresh", "Mahesh", "Ramesh"],  'Subject': ["Science", "SocialSc", "Maths", "English"],  'score':[90, 940, 98, 98]}  df = pd.DataFrame(dict, index = [True, False, True, False])  print(df) | | | | | | | | | | | | | | | | | | | | | | | |
| Q | ''' Write missing statements to produce following output '''  import numpy as np  import pandas as pd  Total\_Sale=pd.Series([8609099,8360000,3390209,6789030,5678769,9876787],\  index=['Sales','Purchasing','Marketing','Production','Accounts','TechSupport'])  Total\_Expense=pd.Series([7810927,6850878,2264631,\  5280634,4678769,8797655],index=['Sales','Purchasing','Marketing','Production','Accounts','TechSupport'])  profit=Total\_Sale-Total\_Expense  commission=profit\*.1  donation=profit\*.05  tax=profit\*.4  dict2={'Sale':Total\_Sale,'Expense':Total\_Expense,'Profit':profit,'commission':commission,'donation':donation,'tax':tax}  dft2=pd.DataFrame(dict2,columns=['Sale','Expense','Profit','commission','donation','tax'])  print(dft2)  **// Missing Statement**  print(dft2)  **// Missing statement**  print(dft2) | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | dft2=dft2.drop("Sales",axis=0) # /dropping row using label , it will remove row with label Sales  dft2=dft2.iloc[2:,] # It will remove row 0 and 1 | | | | | | | | | | | | | | | | | | | | | | | |
| Q | A B C  0 10 20 30.0  1 40 60 NaN  A B  0 100.0 200  1 300.0 500  2 NaN 800  For above two dataframes df3 , df4 what will be the output of following -  print((df3+df4).equals(df3.radd(df4))) | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | True | | | | | | | | | | | | | | | | | | | | | | | |
| Q | A B C  0 NaN NaN NaN  1 NaN NaN NaN  2 NaN NaN NaN  For above data frame write the output of following command -  **df1.empty =** | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | (a) | True | | | (b) | | **False** | | | (c) | None | | | | | | | | | (d) | | | Data Frame Value error | |
| Q | **‘’’ Write the missing print statement that will delete all columns having any NaN value like**  **see in output**  **“””**  **import numpy as np**  **import pandas as pd**  **Total\_Sale=pd.Series([8609099,8360000,3390209,6789030,5678769,576876],\**  **index=['Sales','Purchasing','Marketing','Production','Accounts','TechSupport'])**  **Total\_Expense=pd.Series([7810927,6850878,2264631,\**  **5280634,334565,np.NaN],index=['Sales','Purchasing','Marketing','Production','Accounts','TechSupport'])**  **profit=Total\_Sale-Total\_Expense**  **commission=profit\*.1**  **donation=profit\*.05**  **tax=profit\*.10**  **dict2={'Sale':Total\_Sale,'Expense':Total\_Expense,'Profit':profit,'commission':commission,\**  **'donation':donation,'tax':tax}**  **dft2=pd.DataFrame(dict2,columns=['Sale','Expense','Profit','commission','donation','tax'])**  **print(dft2)**  **print('\n')**  **// Missing Print Statement**  **OUTPUT** | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | **print(dft2.dropna(axis=1))** | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| Q | ‘’’ Write missing statement so that desired output as shown below may be achieved ie. Delete column having all values NaN  ‘’’  import numpy as np  import pandas as pd  Total\_Sale=pd.Series([8609099,8360000,3390209,6789030,5678769,576876],\  index=['Sales','Purchasing','Marketing','Production','Accounts','TechSupport'])  Total\_Expense=pd.Series([7810927,6850878,2264631,\  5280634,334565,np.NaN],index=['Sales','Purchasing','Marketing','Production','Accounts','TechSupport'])  profit=Total\_Sale-Total\_Expense  commission=profit\*.1  donation=profit\*.05  tax=np.NaN  dict2={'Sale':Total\_Sale,'Expense':Total\_Expense,'Profit':profit,'commission':commission,\  'donation':donation,'tax':tax}  dft2=pd.DataFrame(dict2,columns=['Sale','Expense','Profit','commission','donation','tax'])  print(dft2)  print('\n')  **// Missing Statement** | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | print(dft2.dropna(how='all',axis=1)) | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| Q | Write a statement to fill the NaN with value 0 in following data frame dft2 | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | print(dft2.fillna(0)) | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| Q | See following output write command to deleted the TechSupport that is having all values NaN from the data frame dft2 | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | dft2.dropna(how='all',inplace=True) | | | | | | | | | | | | | | | | | | | | | | | |
| Q | What is the work of head() function in data frame ? what will be the output when we write for some data frame df the command df.head(20) where as that dataframe is having only 7 records. ? | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | It will show all records. | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| Q | Make a data frame using 2 D dictionary . | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | import pandas as pd  Total\_Sale=pd.Series([8609099,8360000,3390209,6789030],\  index=['Sales','Purchasing','Marketing','Production'])  Total\_Expense=pd.Series([7810927,6850878,2264631,\  5280634],index=['Sales','Purchasing','Marketing','Production'])  profit=Total\_Sale-Total\_Expense  dict2={'January':Total\_Sale,'February':Total\_Expense,'March':profit}  dft2=pd.DataFrame(dict2,columns=['March','January','February',])  print(dft2) | | | | | | | | | | | | | | | | | | | | | | | |
| Q | Make following data frame df using array | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | # By using 2D nd Array with column names  import numpy as np  import pandas as pd  arr=np.array([[11,21,43,56],[14,35,16,57]])  df=pd.DataFrame(columns=['one','two','three','four'],index=['A','B'],data=arr)  print(df) | | | | | | | | | | | | | | | | | | | | | | | |
| Q | For following data frame dft2 write output of following commands - | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| (i) | dft2.iloc[-10:40,-20:60] | | | | | | | | | | | | | | | | | | | | | | | |
| Ans |  | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| (ii) | **dft2.iloc[-1:-4:-1,-1:-5:-1]** | | | | | | | | | | | | | | | | | | | | | | | |
| Ans |  | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| (iii) | dft2.iloc[-4:6,-5:5] | | | | | | | | | | | | | | | | | | | | | | | |
| Ans |  | | | | | | | | | | | | | | | | | | | | | | | |
|  | Write output of following command if applied to data frame dft2 | | | | | | | | | | | | | | | | | | | | | | | |
| (iv) | dft2.iloc[0:4,0:6] | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| (v) | dft2.iloc[0:4:2,0:6:2] | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
| Q | ‘’’  Make a data frame dft2 as shown below write the code to produce output that shows the values of columns Sale and Expense  ‘’’  Sale Expense  Purchasing 8360000 6850878  Marketing 3390209 2264631  Production 6789030 5280634  Accounts 5678769 4678769  TechSupport 9876787 8797655  **OUTPUT**  Sale  8360000 3390209 6789030 5678769 9876787  Expense  6850878 2264631 5280634 4678769 8797655 | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | import numpy as np  import pandas as pd  Total\_Sale=pd.Series([8360000,3390209,6789030,5678769,9876787],\  index=['Purchasing','Marketing','Production','Accounts','TechSupport'])  Total\_Expense=pd.Series([6850878,2264631,\  5280634,4678769,8797655],index=['Purchasing','Marketing','Production','Accounts','TechSupport'])  dict2={'Sale':Total\_Sale,'Expense':Total\_Expense}  dft2=pd.DataFrame(dict2,columns=['Sale','Expense'])  print(dft2)  sum=0  for (col , colSeries) in dft2.iteritems():  print('\n',col)  for cl in colSeries :  print(cl,end= ' ') | | | | | | | | | | | | | | | | | | | | | | | |
| Q | Sale Expense  Purchasing 8360000 6850878  Marketing 3390209 2264631  Production 6789030 5280634  Accounts 5678769 4678769  TechSupport 9876787 8797655  Purchasing  8360000 6850878 profit= 1509122.0  Marketing  3390209 2264631 profit= 1125578.0  Production  6789030 5280634 profit= 1508396.0  Accounts  5678769 4678769 profit= 1000000.0  TechSupport  9876787 8797655 profit= 1079132.0  Profit is calculated for every row with calculation of sale – purchase , write complete program to produce above output . | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | import numpy as np  import pandas as pd  Total\_Sale=pd.Series([8360000,3390209,6789030,5678769,9876787],\  index=['Purchasing','Marketing','Production','Accounts','TechSupport'])  Total\_Expense=pd.Series([6850878,2264631,\  5280634,4678769,8797655],index=['Purchasing','Marketing','Production','Accounts','TechSupport'])  dict2={'Sale':Total\_Sale,'Expense':Total\_Expense}  dft2=pd.DataFrame(dict2,columns=['Sale','Expense'])  print(dft2)  sum=0  ar=np.empty(2)  for (row , rowSeries) in dft2.iterrows():  i=0  print('\n',row)  for ln in rowSeries :  ar[i]=ln  print(ln,end= ' ')  i+=1  print('profit= ' , ar[0]-ar[1]) | | | | | | | | | | | | | | | | | | | | | | | |
| Q | Sale Expense Profit commission donation tax  Sales 8609099 7810927 798172 79817.2 39908.6 319268.8  Purchasing 8360000 6850878 1509122 150912.2 75456.1 603648.8  Marketing 3390209 2264631 1125578 112557.8 56278.9 450231.2  Production 6789030 5280634 1508396 150839.6 75419.8 603358.4  Accounts 5678769 4678769 1000000 100000.0 50000.0 400000.0  TechSupport 9876787 8797655 1079132 107913.2 53956.6 431652.8 | | | | | | | | | | | | | | | | | | | | | | | |
|  | For above data frame dft2 the output of dft2.Profit['Sales'] will be 798172 write similar commands using at and iat to producing same output . | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | 1. dft2.at['Sales','Profit']   798172   1. dft2.iat[0,2]   798172 | | | | | | | | | | | | | | | | | | | | | | | |
| Q | For data frame dft2 write program to produce following output  Sale Expense  Purchasing 8360000 6850878  Marketing 3390209 2264631  Production 6789030 5280634  Accounts 5678769 4678769  TechSupport 9876787 8797655  Sale  8360000  3390209  6789030  5678769  9876787  The sum of column values is 34094795  Expense  6850878  2264631  5280634  4678769  8797655  The sum of column values is 61967362  Display the value of each column and display its sum. | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | import numpy as np  import pandas as pd  Total\_Sale=pd.Series([8360000,3390209,6789030,5678769,9876787],\  index=['Purchasing','Marketing','Production','Accounts','TechSupport'])  Total\_Expense=pd.Series([6850878,2264631,\  5280634,4678769,8797655],index=['Purchasing','Marketing','Production','Accounts','TechSupport'])  dict2={'Sale':Total\_Sale,'Expense':Total\_Expense}  dft2=pd.DataFrame(dict2,columns=['Sale','Expense'])  print(dft2)  sum=0  for (col , colSeries) in dft2.iteritems():  print('\n',col)  for cl in colSeries :  sum+=cl  print(cl)  print("The sum of column values is ",sum) | | | | | | | | | | | | | | | | | | | | | | | |
| Q | For following dataframe dft2  January February March  Sales 8609099 7810927 798172  Purchasing 8360000 6850878 1509122  Marketing 3390209 2264631 1125578  Production 6789030 5280634 1508396 | | | | | | | | | | | | | | | | | | | | | | | |
| Q | For data frame shown below we want to change the name of January to December and score to percentage write one statements for changing the names for following data frame df to get changed name as shown in out put -  StuName Subject score  January Ravi Science 90  February Suresh SocialSc 94  March Mahesh Maths 98  April Ramesh English 98    StuName Subject percentage  December Ravi Science 90  February Suresh SocialSc 94  March Mahesh Maths 98  April Ramesh English 98 | | | | | | | | | | | | | | | | | | | | | | | |
| Ans | df.rename(index={'January' : 'December'},columns = { 'score' : 'percentage'}, inplace = True) | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |