Python 101 - Python Libraries for Data Analysis - Numpy and Pandas

March 12, 2025

1 TASK #1: DEFINE SINGLE AND MULTI-DIMENSIONAL NUMPY ARRAYS

[2]:
```
# NumPy is a Linear Algebra Library used for multidimensional arrays
# NumPy brings the best of two worlds: (1) C/Fortran computational efficiency,␣
 ↪(2) Python language easy syntax
import numpy as np
# Let's define a one-dimensional array
list_1=[50,60,80,200,400,600]
list_1
```

[2]:
```
[50, 60, 80, 200, 400, 600]
```

[8]:
```
# Let's create a numpy array from the list "my_list"
my_numpy_array = np.array(list_1)
my_numpy_array
```

[8]:
```
array([ 50,  60,  80, 200, 400, 600])
```

[9]:
```
type(my_numpy_array)    #TYPE OF ARRAY
```

[9]:
```
numpy.ndarray
```

[10]:
```
# Multi-dimensional (Matrix definition)
my_matrix = np.array([[2,5,8],[7,3,6]   ])
my_matrix                    #md array formation
```

[10]:
```
array([[2, 5, 8],
       [7, 3, 6]])
```

MINI CHALLENGE #1: - Write a code that creates the following 2x4 numpy array

```
[[3 7 9 3]
[4 3 2 2]]
```

[16]:
```
matrix= np.array([[3, 7, 9, 3],[4, 3 ,2,2]   ])
```

2 TASK #2: LEVERAGE NUMPY BUILT-IN METHODS AND FUNCTIONS

[19]:
```
# "rand()" uniform distribution between 0 and 1
x=np.random.rand(20)
x
```

[19]:
```
array([0.27232682, 0.06299236, 0.28990714, 0.68866086, 0.98940755,
       0.21838975, 0.09481182, 0.68570891, 0.34887106, 0.67898485,
       0.57318156, 0.75260968, 0.15996081, 0.13769597, 0.16829168,
       0.47828429, 0.77416428, 0.22338817, 0.13448085, 0.17216128])
```

[20]:
```
# you can create a matrix of random number as well
x=np.random.rand(3,3)
x
```

[20]:
```
array([[0.07606883, 0.67211099, 0.54853555],
       [0.65006596, 0.53541714, 0.95996861],
       [0.87190149, 0.51416981, 0.25830688]])
```

[27]:
```
# "randint" is used to generate random integers between upper and lower bounds
x=np.random.randint(1,30,5)
x      # random number
```

[27]:
```
array([10,  7,  2,  2, 29])
```

[32]:
```
# "randint" can be used to generate a certain number of random itegers as␣
↪follows
```

[34]:
```
# np.arange creates an evenly spaced values within a given interval
x=np.arange(1,30)
x
```

[34]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29])
```

[35]:
```
# create a diagonal of ones and zeros everywhere else
x=np.eye(7)
x
```

[35]:
```
array([[1., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 0., 1.]])
```

[40]:
```
# Matrix of ones
x=np.ones((7,7))
x
```

[40]:
```
array([[1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.]])
```

[44]:
```
# Array of zeros
x=np.zeros(8)
x
```

[44]:
```
array([0., 0., 0., 0., 0., 0., 0., 0.])
```

MINI CHALLENGE #2: - Write a code that takes in a positive integer "x" from the user and creates a 1x10 array with random numbers ranging from 0 to "x"

[46]:
```
x=int(input('enter a number'))
x=np.random.randint(1,x,10)
x
```

```
enter a number10
```

[46]:
```
array([1, 7, 3, 9, 1, 9, 6, 7, 2, 7])
```

3 TASK #3: PERFORM MATHEMATICAL OPERATIONS IN NUMPY

[48]:
```
# np.arange() returns an evenly spaced values within a given interval
x=np.arange(1,10)
x
```

[48]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[49]:
```
y=np.arange(1,10)
y
```

[49]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[50]:
```
# Add 2 numpy arrays together
sum = x+y
sum
```

[50]:
```
array([ 2,  4,  6,  8, 10, 12, 14, 16, 18])
```

[51]:
```
square=x**2
square
```

[51]:
```
array([ 1,  4,  9, 16, 25, 36, 49, 64, 81])
```

[52]:
```
root=np.sqrt(square)
root
```

[52]:
```
array([1., 2., 3., 4., 5., 6., 7., 8., 9.])
```

[53]:
```
z= np.exp(y)
z
```

[53]:
```
array([2.71828183e+00, 7.38905610e+00, 2.00855369e+01, 5.45981500e+01,
       1.48413159e+02, 4.03428793e+02, 1.09663316e+03, 2.98095799e+03,
       8.10308393e+03])
```

MINI CHALLENGE #3: - Given the X and Y values below, obtain the distance between them

```
X = [5, 7, 20]
Y = [9, 15, 4]
```

[66]:
```
X = np.array([5, 7, 20])
Y = np.array([9, 15, 4])
z=np.sqrt(X**2+Y**2)
z
```

[66]:
```
array([10.29563014, 16.55294536, 20.39607805])
```

4 TASK #4: PERFORM ARRAYS SLICING AND INDEXING

[3]:
```
my_numpy_array=np.array([3,5,6,2,8,10,20,50])
my_numpy_array
```

[3]:
```
array([ 3,  5,  6,  2,  8, 10, 20, 50])
```

[5]:
```
# Access specific index from the numpy array
my_numpy_array[0]
my_numpy_array[-1]
```

[5]:
```
50
```

[8]:
```
# Starting from the first index 0 up until and NOT including the last element
my_numpy_array[0:3]
```

[8]:
```
array([3, 5, 6])
```

[9]:
```
# Broadcasting, altering several values in a numpy array at once
my_numpy_array[0:4]=7
my_numpy_array
```

[9]:
```
array([ 7,  7,  7,  7,  8, 10, 20, 50])
```

[13]:
```
# Let's define a two dimensional numpy array
matrix=np.random.randint(1,10,(4,4))
matrix
```

[13]:
```
array([[8, 2, 3, 6],
       [4, 2, 4, 4],
       [7, 4, 9, 2],
       [8, 2, 8, 5]])
```

[16]:
```
# Get a row from a mtrix
matrix[0]
matrix[1]
matrix[-1]
```

[16]:
```
array([8, 2, 8, 5])
```

[26]:
```
# Get one element
matrix=[0][0]
```

MINI CHALLENGE #4: - In the following matrix, replace the last row with 0

```
X = [2 30 20 -2 -4]
    [3 4  40 -3 -2]
    [-3 4 -6 90 10]
    [25 45 34 22 12]
    [13 24 22 32 37]
```

[32]:
```
X = np.array([[2, 30, 20, -2 ,-4],
              [3 ,4,  40, -3, -2],
              [-3 ,4, -6 ,90 ,10],
              [25, 45 ,34, 22, 12],
              [13, 24 ,22 ,32, 37]])
X[4]=0
X
```

[32]:
```
array([[ 2, 30, 20, -2, -4],
       [ 3,  4, 40, -3, -2],
       [-3,  4, -6, 90, 10],
       [25, 45, 34, 22, 12],
       [ 0,  0,  0,  0,  0]])
```

5 TASK #5: PERFORM ELEMENTS SELECTION (CONDITIONAL)

[33]:
```
matrix=np.random.randint(1,10,(5,5))
matrix
```

[33]:
```
array([[7, 5, 4, 5, 5],
       [2, 6, 4, 8, 4],
       [1, 4, 8, 5, 1],
       [9, 1, 4, 1, 5],
       [9, 4, 3, 1, 5]])
```

[38]:
```
x=matrix[matrix > 7]
x
```

[38]:
```
array([8, 8, 9, 9])
```

[42]:
```
# Obtain odd elements only
x=matrix[matrix %2==1]
x
```

[42]:
```
array([7, 5, 5, 5, 1, 5, 1, 9, 1, 1, 5, 9, 3, 1, 5])
```

MINI CHALLENGE #5: - In the following matrix, replace negative elements by 0 and replace odd elements with -2

```
X = [2 30 20 -2 -4]
    [3 4  40 -3 -2]
    [-3 4 -6 90 10]
    [25 45 34 22 12]
    [13 24 22 32 37]
```

[51]:
```
X = np.array([[2, 30 ,20, -2, -4],
    [3, 4  ,40, -3 ,-2],
    [-3, 4, -6 ,90, 10],
    [25 ,45 ,34 ,22 ,12],
    [13, 24 ,22 ,32 ,37]])
X[X<0]=0
X[X % 2==1]=-2
X
```

[51]:
```
array([[ 2, 30, 20,  0,  0],
       [-2,  4, 40,  0,  0],
       [ 0,  4,  0, 90, 10],
       [-2, -2, 34, 22, 12],
       [-2, 24, 22, 32, -2]])
```

6 TASK #6: UNDERSTAND PANDAS FUNDAMENTALS

[ ]:
```
# Pandas is a data manipulation and analysis tool that is built on Numpy.
# Pandas uses a data structure known as DataFrame (think of it as Microsoft␣
 ↪excel in Python).
# DataFrames empower programmers to store and manipulate data in a tabular␣
 ↪fashion (rows and columns).
# Series Vs. DataFrame? Series is considered a single column of a DataFrame.
```

[53]:
```
import pandas as pd
```

[64]:
```
# Let's define a two-dimensional Pandas DataFrame
# Note that you can create a pandas dataframe from a python dictionary
bank_clint_df = pd.DataFrame({
    'Bank Client Id': [111, 112, 113, 114],
    'Bank Client Name': ['rohit', 'kohli', 'rahul', 'dhawan'],
    'Net Worth [$]': [555000, 34000, 56000, 34000]
})
bank_clint_df
```

[64]:

	Bank Client Id	Bank Client Name	Net Worth [$]
0	111	rohit	555000
1	112	kohli	34000
2	113	rahul	56000
3	114	dhawan	34000

[65]:
```
# Let's obtain the data type
type(bank_clint_df)
```

[65]:
```
pandas.core.frame.DataFrame
```

[68]:
```
# you can only view the first couple of rows using .head()
bank_clint_df.head
```

[68]:
```
<bound method NDFrame.head of    Bank Client Id Bank Client Name  Net Worth [$]
0             111            rohit         555000
1             112            kohli          34000
2             113            rahul          56000
3             114           dhawan          34000>
```

[70]:
```
# you can only view the last couple of rows using .tail()
bank_clint_df.tail(2)
```

[70]:

	Bank Client Id	Bank Client Name	Net Worth [$]
2	113	rahul	56000
3	114	dhawan	34000

MINI CHALLENGE #6: - A porfolio contains a collection of securities such as stocks, bonds and ETFs. Define a dataframe named 'portfolio_df' that holds 3 different stock ticker symbols, number of shares, and price per share (feel free to choose any stocks) - Calculate the total value of the porfolio including all stocks

[79]:
```
portfolio_df=pd.DataFrame({'Stock ticker symbol':['Apple','Amz','T'],'price per␣
 ↪share[$]':[
                          3500,200,250],'Number of stocks':[3,4,5] })
portfolio_df
stock_dollar_value=portfolio_df['price per share[$]']*portfolio_df['Number of␣
 ↪stocks']
stock_dollar_value
```

[79]:
```
0    10500
1      800
2     1250
dtype: int64
```

7 TASK #7: PANDAS WITH CSV AND HTML DATA

[75]:
```
# Pandas is used to read a csv file and store data in a DataFrame
```

[ ]:
```

```

[85]:
```
# Read tabular data using read_html
house_price_df=pd.read_html('https://www.livingin-canada.com/
 ↪house-prices-canada.html#google_vignette')
house_price_df[0]
```

[85]:
```
                                                City  \
0                                        Vancouver, BC
1                                         Toronto, Ont
2                                          Ottawa, Ont
3                                         Calgary, Alb
4                                        Montreal, Que
5                                          Halifax, NS
6                                         Regina, Sask
7                                       Fredericton, NB
8  (adsbygoogle = window.adsbygoogle || []).push(…

                                    Average House Price  \
0                                             $1,036,000
1                                               $870,000
2                                               $479,000
3                                               $410,000
4                                               $435,000
```

```
5                                                $331,000
6                                                $254,000
7                                                $198,000
8  (adsbygoogle = window.adsbygoogle || []).push(…

                                      12 Month Change
0                                             + 2.63 %
1                                              +10.2 %
2                                             + 15.4 %
3                                              - 1.5 %
4                                              + 9.3 %
5                                              + 3.6 %
6                                              - 3.9 %
7                                              - 4.3 %
8  (adsbygoogle = window.adsbygoogle || []).push(…
```

[88]:
```
house_price_df[1]
```

[88]:
```
                                        Province  \
0                                  British Columbia
1                                           Ontario
2                                           Alberta
3                                            Quebec
4                                          Manitoba
5                                      Saskatchewan
6                                       Nova Scotia
7                                Prince Edward Island
8                              Newfoundland / Labrador
9                                      New Brunswick
10                                  Canadian Average
11  (adsbygoogle = window.adsbygoogle || []).push(…

                                    Average House Price  \
0                                              $736,000
1                                              $594,000
2                                              $353,000
3                                              $340,000
4                                              $295,000
5                                              $271,000
6                                              $266,000
7                                              $243,000
8                                              $236,000
9                                              $183,000
10                                             $488,000
11  (adsbygoogle = window.adsbygoogle || []).push(…

                                      12 Month Change
```

```
0                                                + 7.6 %
1                                                - 3.2 %
2                                                - 7.5 %
3                                                + 7.6 %
4                                                - 1.4 %
5                                                - 3.8 %
6                                                + 3.5 %
7                                                + 3.0 %
8                                                - 1.6 %
9                                                - 2.2 %
10                                               - 1.3 %
11  (adsbygoogle = window.adsbygoogle || []).push(…
```

[ ]:

MINI CHALLENGE #7: - Write a code that uses Pandas to read tabular US retirement data - You can use data from here: https://www.ssa.gov/oact/progdata/nra.html

[90]:
```
house_price_df=pd.read_html('https://www.ssa.gov/oact/progdata/nra.html')
house_price_df[0]
```

[90]:
```
                                    Year of birth  \
0                                  1937 and prior
1                                            1938
2                                            1939
3                                            1940
4                                            1941
5                                            1942
6                                         1943-54
7                                            1955
8                                            1956
9                                            1957
10                                           1958
11                                           1959
12                                1960 and later
13  Notes: 1. Persons born on January 1 of any yea…

                                             Age
0                                             65
1                                 65 and 2 months
2                                 65 and 4 months
3                                 65 and 6 months
4                                 65 and 8 months
5                                65 and 10 months
6                                             66
7                                 66 and 2 months
8                                 66 and 4 months
```

```
9                                    66 and 6 months
10                                   66 and 8 months
11                                  66 and 10 months
12                                                67
13  Notes: 1. Persons born on January 1 of any yea…
```

8 TASK #8: PANDAS OPERATIONS

[93]:
```
# Let's define a dataframe as follows:
bank_clint_df = pd.DataFrame({
    'Bank Client Id': [111, 112, 113, 114],
    'Bank Client Name': ['rohit', 'kohli', 'rahul', 'dhawan'],
    'Net Worth [$]': [555000, 34000, 56000, 34000],
    'years with bank':[9,3,5,4]
})
bank_clint_df
```

[93]:

	Bank Client Id	Bank Client Name	Net Worth [$]	years with bank
0	111	rohit	555000	9
1	112	kohli	34000	3
2	113	rahul	56000	5
3	114	dhawan	34000	4

[94]:
```
# Pick certain rows that satisfy a certain criteria
df_loyal=bank_clint_df[bank_clint_df['years with bank']>=5]
df_loyal
```

[94]:

	Bank Client Id	Bank Client Name	Net Worth [$]	years with bank
0	111	rohit	555000	9
2	113	rahul	56000	5

[95]:
```
# Delete a column from a DataFrame
del bank_clint_df['years with bank']
bank_clint_df
```

[95]:

	Bank Client Id	Bank Client Name	Net Worth [$]
0	111	rohit	555000
1	112	kohli	34000
2	113	rahul	56000
3	114	dhawan	34000

MINI CHALLENGE #8: - Using “bank_client_df” DataFrame, leverage pandas operations to only select high networth individuals with minimum $5000 - What is the combined networth for all customers with 5000+ networth?

[104]:
```
rich_clint_df=bank_clint_df[bank_clint_df['Net Worth [$]'] >=5000]
rich_clint_df
bank_clint_df['Net Worth [$]'].sum()
```

[104]:
```
679000
```

9 TASK #9: PANDAS WITH FUNCTIONS

[ ]:
```
# Let's define a dataframe as follows:
bank_clint_df = pd.DataFrame({
    'Bank Client Id': [111, 112, 113, 114],
    'Bank Client Name': ['rohit', 'kohli', 'rahul', 'dhawan'],
    'Net Worth [$]': [555000, 34000, 56000, 34000],
    'years with bank':[9,3,5,4]
})
bank_clint_df
```

[105]:
```
# Define a function that increases all clients networth (stocks) by a fixed␣
 ↪value of 20% (for simplicity sake)
def networth_update(balance):
    return balance*1.2
```

[107]:
```
# You can apply a function to the DataFrame
bank_clint_df['Net Worth [$]'].apply(networth_update)
```

[107]:
```
0    666000.0
1     40800.0
2     67200.0
3     40800.0
Name: Net Worth [$], dtype: float64
```

[108]:
```
bank_clint_df['Bank Client Name'].apply(len)
```

[108]:
```
0    5
1    5
2    5
3    6
Name: Bank Client Name, dtype: int64
```

MINI CHALLENGE #9: - Define a function that triples the stock prices and adds $200 - Apply the function to the DataFrame - Calculate the updated total networth of all clients combined

[118]:
```
def networth_update(balance):
    return balance*3 + 200
```

[119]:
```
bank_clint_df['Net Worth [$]'].apply(networth_update)
```

[119]:
```
0    1665200
1     102200
2     168200
3     102200
Name: Net Worth [$], dtype: int64
```

10 TASK #10: PERFORM SORTING AND ORDERING IN PANDAS

[120]:
```
# Let's define a dataframe as follows:
bank_clint_df = pd.DataFrame({
    'Bank Client Id': [111, 112, 113, 114],
    'Bank Client Name': ['rohit', 'kohli', 'rahul', 'dhawan'],
    'Net Worth [$]': [555000, 34000, 56000, 34000],
    'years with bank':[9,3,5,4]
})
bank_clint_df
```

[120]:

	Bank Client Id	Bank Client Name	Net Worth [$]	years with bank
0	111	rohit	555000	9
1	112	kohli	34000	3
2	113	rahul	56000	5
3	114	dhawan	34000	4

[124]:
```
# You can sort the values in the dataframe according to number of years with
bank_clint_df.sort_values(by = 'years with bank')
```

[124]:

	Bank Client Id	Bank Client Name	Net Worth [$]	years with bank
1	112	kohli	34000	3
3	114	dhawan	34000	4
2	113	rahul	56000	5
0	111	rohit	555000	9

[125]:
```
# Note that nothing changed in memory! you have to make sure that inplace is␣
 ↪set
bank_clint_df
```

[125]:

	Bank Client Id	Bank Client Name	Net Worth [$]	years with bank
0	111	rohit	555000	9
1	112	kohli	34000	3
2	113	rahul	56000	5
3	114	dhawan	34000	4

[127]:
```
# Set inplace = True to ensure that change has taken place in memory
bank_clint_df.sort_values(by = 'years with bank',inplace = True)
```

[128]:
```
# Note that now the change (ordering) took place
bank_clint_df
```

[128]:

	Bank Client Id	Bank Client Name	Net Worth [$]	years with bank
1	112	kohli	34000	3
3	114	dhawan	34000	4
2	113	rahul	56000	5
0	111	rohit	555000	9

11 TASK #11: PERFORM CONCATENATING AND MERGING WITH PANDAS

[ ]:
```
# Check this out: https://pandas.pydata.org/pandas-docs/stable/user_guide/
↪merging.html
```

[135]:
```
df1 = pd.DataFrame({'A':['A0','A1','A2','A3'],
                    'B':['B0','B1','B2','B3']},
                   index=[0,1,2,3])
df1
```

[135]:

	A	B
0	A0	B0
1	A1	B1
2	A2	B2
3	A3	B3

[136]:
```
df2 = pd.DataFrame({'A':['A4','A5','A6','A7'],
                    'B':['B4','B5','B6','B7']},
                   index=[0,1,2,3])
df2
```

[136]:

	A	B
0	A4	B4
1	A5	B5
2	A6	B6
3	A7	B7

[139]:
```
df2
```

[139]:

	A	B
0	A4	B4
1	A5	B5
2	A6	B6
3	A7	B7

[144]:
```
pd.concat([df1,df2])
```

[144]:

	A	B
0	A0	B0
1	A1	B1
2	A2	B2
3	A3	B3
0	A4	B4
1	A5	B5
2	A6	B6
3	A7	B7

[ ]:

[ ]:

[ ]:

[ ]:

12 TASK #12: PROJECT AND CONCLUDING REMARKS

- Define a dataframe named 'Bank_df_1' that contains the first and last names for 5 bank clients with IDs = 1, 2, 3, 4, 5
- Assume that the bank got 5 new clients, define another dataframe named 'Bank_df_2' that contains a new clients with IDs = 6, 7, 8, 9, 10
- Let's assume we obtained additional information (Annual Salary) about all our bank customers (10 customers)
- Concatenate both 'bank_df_1' and 'bank_df_2' dataframes
- Merge client names and their newly added salary information using the 'Bank Client ID'
- Let's assume that you became a new client to the bank
- Define a new DataFrame that contains your information such as client ID (choose 11), first name, last name, and annual salary.
- Add this new dataframe to the original dataframe 'bank_df_all'.

[154]:
```
raw_data={ 'Bank Client Id': [1, 2, 3, 4,5],
    'First Name': ['rohit', 'kohli', 'rahul', 'dhawan','suresh'],
    'Last name':['sharma','virat','koonor','shikhar','raina']}
Bank_df_1 = pd.DataFrame(raw_data,columns=['Bank Client Id', 'First Name',␣
 ↪'Last name'])

Bank_df_1
```

[154]:

	Bank Client Id	First Name	Last name
0	1	rohit	sharma
1	2	kohli	virat
2	3	rahul	koonor
3	4	dhawan	shikhar

```
4               5     suresh     raina
```

[168]:
```
raw_data={ 'Bank Client Id':[6, 7, 8,9,10],
    'First Name': ['shubhman', 'shreyas', 'varun', 'jassi','siraj'],
    'Last name':['gill','iyer','cv','bumrah','miya']}
Bank_df_2 = pd.DataFrame(raw_data,columns=['Bank Client Id', 'First Name',␣
 ↪'Last name'])

Bank_df_2
```

[168]:
```
   Bank Client Id First Name Last name
0               6   shubhman      gill
1               7    shreyas      iyer
2               8      varun        cv
3               9      jassi    bumrah
4              10      siraj      miya
```

[175]:
```
raw_data={'Bank Client Id': [1, 2, 3, 4,5,6,7,8,9,10],
         'Annual salary[$/year]':
 ↪[100000,90000,80000,70000,60000,50000,40000,30000,20000,1000]}
bank_df_salary=pd.DataFrame(raw_data,columns=['Bank Client Id','Annual salary[$/
 ↪year]'])
bank_df_salary
```

[175]:
```
   Bank Client Id  Annual salary[$/year]
0               1                 100000
1               2                  90000
2               3                  80000
3               4                  70000
4               5                  60000
5               6                  50000
6               7                  40000
7               8                  30000
8               9                  20000
9              10                   1000
```

[180]:
```
bank_df_all=pd.concat([Bank_df_1,Bank_df_2])
bank_df_all
```

[180]:
```
   Bank Client Id First Name Last name
0               1      rohit    sharma
1               2      kohli     virat
2               3      rahul    koonor
3               4     dhawan   shikhar
4               5     suresh     raina
0               6   shubhman      gill
1               7    shreyas      iyer
```

```
2                8      varun        cv
3                9      jassi    bumrah
4               10      siraj      miya
```

[183]:
```
bank_df_all = pd.merge(bank_all, bank_df_salary, on='Bank Client Id')
bank_df_all
```

[183]:
```
   Bank Client Id First Name Last name  Annual salary[$/year]
0               1      rohit    sharma                 100000
1               2      kohli     virat                  90000
2               3      rahul    koonor                  80000
3               4     dhawan   shikhar                  70000
4               5     suresh     raina                  60000
5               6   shubhman      gill                  50000
6               7    shreyas      iyer                  40000
7               8      varun        cv                  30000
8               9      jassi    bumrah                  20000
9              10      siraj      miya                   1000
```

[195]:
```
new_clint={'Bank Client Id':['11'],
           'First Name':['Ajinkya'],
           'Last name':['rahane'],
           'Annual salary[$/year]':['40000']}
new_clint=pd.DataFrame(new_clint,columns=['Bank Client Id', 'First Name', 'Last␣
 ↪name','Annual salary[$/year]'])
new_clint
```

[195]:
```
  Bank Client Id First Name Last name Annual salary[$/year]
0             11    Ajinkya    rahane                 40000
```

[196]:
```
new_df=pd.concat([bank_df_all,new_clint],axis=0)
new_df
```

[196]:
```
  Bank Client Id First Name Last name Annual salary[$/year]
0              1      rohit    sharma                100000
1              2      kohli     virat                 90000
2              3      rahul    koonor                 80000
3              4     dhawan   shikhar                 70000
4              5     suresh     raina                 60000
5              6   shubhman      gill                 50000
6              7    shreyas      iyer                 40000
7              8      varun        cv                 30000
8              9      jassi    bumrah                 20000
9             10      siraj      miya                  1000
0             11    Ajinkya    rahane                 40000
```