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

August 21, 2025

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

[9]:
```
# NumPy is a Linear Algebra Library used for multidimensional arrays
# NumPy brings the best of two worlds: (1) C/Fortran computational efficiency,␣
  ↪(2)Pythonlanguageeasysyntax
importnumpyasnp
# Let's define a one-dimensional array
my_list = [50, 60,          80, 100, 200, 300, 500, 600]
```

[10]:
```
# Let's create a numpy array from the list "my_list"
array1 = np.array(my_list)
array1
```

[10]:
```
array([ 50, 60, 80, 100, 200, 300, 500, 600])
```

[11]:
```
type(array1)
```

[11]:
```
numpy.ndarray
```

[12]:
```
# Multi-dimensional (Matrix definition)
array1 = np.array([[3,4,5,6],[7,8,2,6]])
array1
```

[12]:
```
array([[3, 4, 5, 6],
       [7, 8, 2, 6]])
```

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

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

[]:
```

```

2 TASK#2:LEVERAGENUMPYBUILT-INMETHODSAND FUNCTIONS

[13]:
```
# "rand()" uniform distribution between 0 and 1
x = np.random.rand(20)
x
```

[13]: array([0.27971061, 0.46261807, 0.53723437, 0.58502856, 0.89282425,
0.76473258, 0.29686361, 0.37553664, 0.72816157, 0.50616463,
0.87748365, 0.00268287, 0.40839958, 0.28707817, 0.70594521,
0.16657046, 0.83507506, 0.47785664, 0.58420662, 0.10975746])

[14]:
```
# you can create a matrix of random number as well
x = np.random.rand(3,3)
x
```

[14]: array([[0.19761322, 0.64335951, 0.86683971],
[0.32251555, 0.19640324, 0.20048193],
[0.63301302, 0.02266398, 0.77982019]])

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

[27]: 2 [28]: # "randint" can be used to generate a certain number of random itegers

```
as␣
  follows
x=np.random.randint(1,100,15)
x
```

[28]: array([54, 16, 63, 50, 72, 86, 59, 81, 22, 25, 98, 91, 63, 44, 96]) [30]: # np.arange

```
creates an evenly spaced values within a given interval
x = np.arange(1,50)
x
```

[30]: array([1, 9,10,11,2,2,13,3,14,4,15,5,16,17, 6, 7, 8,
18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49])

[32]:
```
# create a diagonal of ones and zeros everywhere else
x = np.eye(5) # this method set the matrix og 5*5 row and column and diagnoly␣
  setthe1digitandtherestofitzeros
x
```

[32]: array([[1., 0., 0., 0., 0.], [0., 1., 0., 0., 0.],
[0., 0., 1., 0., 0.], [0., 0., 0., 1., 0.], [0., 0., 0.,
0., 1.]])

[37]:
```
# Matrix of ones
x = np.ones((7,7))
x
```

[37]: 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.]])

[40]:
```
# Array of zeros
x = np.zeros((1,10))
x
```

[40]: array([[0., 0., 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"

[]:
```

```

3 TASK#3: PERFORMMATHEMATICALOPERATIONSIN NUMPY

[41]:
```
# np.arange() returns an evenly spaced values within a given interval
x = np.arange(1,10)
x
```

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

[42]:
```
y = np.arange(1,10)
y
```

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

[47]:
```
# Add 2 numpy arrays together
sum = x+y
sum
```

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

[45]:
```
squared = x**2
squared
```

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

[]:

[48]:
```
sqrt = np.sqrt(squared)
sqrt
```

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

[51]:
```
X = np.array([5,7,20])
Y = np.array([9,15,4])
Z = np.sqrt(X**2 + Y**2)
Z
```

[51]: array([10.29563014, 16.55294536, 20.39607805])

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

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

[]:

4 TASK#4: PERFORMARRAYSSLICINGANDINDEXING

[53]:
```
array = np.array([3,5,6,2,8,10,20,50])
array
```

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

[54]:
```
# Access specific index from the numpy array
array[1]
```

[54]: 5

[59]:
```
# Starting from the first index 0 up until and NOT including the last element
array[0:3]
```

[59]: array([3, 5, 6])

[62]:
```
# Broadcasting, altering several values in a numpy array at once
array[0:4] = 7
array
```

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

[65]:
```
# Let's define a two dimensional numpy array
matrix = np.random.randint(1,10,(4,4)) # here 4*4 matrix
matrix
```

[65]: array([[9, 6, 9, 2],
[6, 6, 5, 1],
[9, 3, 6, 6],
[8, 6, 5, 2]])

[70]:
```
# Get a row from a mtrix
matrix[0]
```

[70]: array([9, 6, 9, 2])

[71]:
```
# Get one element
matrix[0][0]
```

[71]: 9

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]

[78]:
```
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
```

[78]: 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)

[80]:
```
matrix = np.random.randint(1,10,(5,5))
matrix
```

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

[87]:
```
new_matrix = matrix[matrix > 7]
new_matrix
```

[87]: array([9, 9, 8, 9, 9, 9, 8])

[89]:
```
# Obtain odd elements only
new_matrix = matrix[matrix % 2 ==1]
new_matrix
```

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

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]

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

[90]: 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: UNDERSTANDPANDASFUNDAMENTALS

[ ]:
```
# 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 excelinPython).
#DataFramesempowerprogrammerstostoreandmanipulatedatainatabular fashion(rowsandcolumns).
#SeriesVs.DataFrame?Seriesisconsideredasinglecolumnofa DataFrame.
```

[92]:
```
import pandas as pd
```

[93]:
```
# Let's define a two-dimensional Pandas DataFrame
# Note that you can create a pandas dataframe from a python dictionary
bank_client_df = pd.DataFrame({"Bank Client ID":[111, 222, 333, 444],
                               "Bank Client Name":['Chanel','Steve','Mitch','Ryan'],
                               "NetWorth[$]":[3500,29000,10000,2000],
                               "Years with bank":[3,4,9,5]

                              })
bank_client_df
```

[93]:

	BankClient ID	BankClient Name	Net Worth[$]	Yearswithbank
0	111	Chanel	3500	3
1	222	Steve	29000	4
2	333	Mitch	10000	9
3	444	Ryan	2000	5

[94]:
```
# Let's obtain the data type
type(bank_client_df)
```

[94]: pandas.core.frame.DataFrame

[96]:
```
# you can only view the first couple of rows using .head()
bank_client_df.head(2)
```

[96]:

	BankClient ID	BankClient Name	Net Worth[$]	Yearswithbank
0	111	Chanel	3500	3
1	222	Steve	29000	4

[99]:
```
# you can only view the last couple of rows using .tail()
bank_client_df.tail(2)
```

[99]:

	BankClient ID	BankClient Name	Net Worth[$]	Yearswithbank
2	333	Mitch	10000	9
3	444	Ryan	2000	5

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

[104]:
```
portfolio_df = pd.DataFrame({"stock ticker symbol":['AAPL','AMZN','TSLA'],
                             "price per share [$]":[3500, 200, 40],
                             "Number of stocks":[3, 4, 9]

                            })
portfolio_df
```

[104]:

	stockticker symbol	price per share [$]	Numberofstocks
0	AAPL	3500	3
1	AMZN	200	4
2	TSLA	40	9

[107]:
```
stocks_dollar_value = portfolio_df['price per share [$]'] *␣
↪portfolio_df['Numberofstocks']
stocks_dollar_value
```

[107]:
```
0    10500
1      800
2      360
dtype: int64
```

7 TASK #7: PANDAS WITH CSV AND HTML DATA

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

[]:
```
```

[9]:
```
import pandas as pd
# Read tabular data using read_html
house_price_df = pd.read_html("https://www.livingin-canada.com/
↪house-prices-canada.html")
house_price_df[0]
```

[9]:

	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(...

[10]:
```
house_price_df[1]
```

[10]:

	Province
0	British Columbia
1	Ontario
2	Alberta
3	Quebec
4	Manitoba
5	Saskatchewan
6	NovaScotia
7	PrinceEdwardIsland
8	Newfoundland/Labrador
9	NewBrunswick
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		$266,00
7		0
8		$243,00
9		0
10		$236,00
11	(adsbygoogle = window.adsbygoogle || []).push(...	0
		$183,00
		0

	12 Month Change
0	$488,00
1	0 – 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

[]:

8 TASK#8: PANDASOPERATIONS

[20]:
```
# Let's define a dataframe as follows:
bank_client_df = pd.DataFrame({"Bank Client ID":[111, 222, 333, 444],
                               "Bank Client Name":['Chanel','Steve','Mitch',
 'Ryan'],
                               "NetWorth[$]":[3500,29000,10000,2000],
                               "Years with bank":[3,4,9,5]
                              })
bank_client_df
```

[20]:

	Bank Client ID	Bank Client Name	Net Worth[$]	Years with bank
0	111	Chanel	3500	3
1	222	Steve	29000	4
2	333	Mitch	10000	9
3	444	Ryan	2000	5

[16]:
```
# Pick certain rows that satisfy a certain criteria
loyal_customer = bank_client_df[ bank_client_df["Years with bank"]>=5 ]
loyal_customer
```

[16]:

	BankClient ID	BankClient Name	Net Worth[$]	Yearswithbank
2	333	Mitch	10000	9
3	444	Ryan	2000	5

[21]:
```
# Delete a column from a DataFrame
del bank_client_df['Bank Client ID']
bank_client_df
```

[21]:

	BankClient Name	NetWorth [$]	Yearswithbank
0	Chanel	3500	3
1	Steve	29000	4
2	Mitch	10000	9
3	Ryan	2000	5

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?

[24]:
```
net_worth = bank_client_df[bank_client_df["Net Worth [$]"]>5000]
net_worth
```

[24]:

	BankClient Name	NetWorth [$]	Yearswithbank
1	Steve	29000	4
2	Mitch	10000	9

9 TASK#9:PANDASWITHFUNCTIONS

[38]:
```
# Let's define a dataframe as follows:
bank_client_df = pd.DataFrame({'Bank client ID':[111, 222, 333, 444],
                               'Bank Client Name':['Chanel', 'Steve', 'Mitch',␣
 'Ryan'],
                               'Networth[$]':[3500,29000,10000,2000],
                               'Years with bank':[3, 4, 9, 5]})
bank_client_df
```

[38]:

	Bankclient ID	BankClient Name	Net worth[$]	Yearswithbank
0	111	Chanel	3500	3
1	222	Steve	29000	4
2	333	Mitch	10000	9
3	444	Ryan	2000	5

[41]:
```
# 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
```

[42]:
```
# You can apply a function to the DataFrame
bank_client_df['Net worth [$]'].apply(networth_update)
```

[42]:
```
0
1     4200.0
2    34800.0
3    12000.0
4     2400.0
Name: Net worth [$], dtype: float64
```

[43]:
```
bank_client_df['Bank Client Name'].apply(len)
```

[43]:
```
0    6
1    5
2    5
3    4
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

[47]:
```
def stock_price(update):
    return ((update*3)+200)

result = bank_client_df['Net worth [$]'].apply(stock_price)
result
```

[47]:
```
0    10700
1    87200
2    30200
3     6200
Name: Net worth [$], dtype: int64
```

[48]:
```
result.sum()
```

[48]:
```
134300
```

10 TASK #10: PERFORM SORTING AND ORDERING IN PANDAS

[53]:
```
# Let's define a dataframe as follows:
bank_client_df = pd.DataFrame({'Bank client ID':[111, 222, 333, 444],
                               'Bank Client Name':['Chanel', 'Steve', 'Mitch',␣
                               'Ryan'],
                               'Networth[$]':[3500,29000,10000,2000],
                               'Years with bank':[3, 4, 9, 5]})
```

[55]:
```
# You can sort the values in the dataframe according to number of years with␣
bank
bank_client_df.sort_values(by="Yearswithbank")
```

[55]:

	Bankclient ID	BankClient Name	Net worth[$]	Yearswithbank
0	111	Chanel	3500	3
1	222	Steve	29000	4
3	444	Ryan	2000	5
2	333	Mitch	10000	9

[61]:
```
# Note that nothing changed in memory! you have to make sure that inplace is␣
settoTrue
bank_client_df.sort_values(by="Yearswithbank",inplace=True)
 bank_client_df
```

[61]:

	Bankclient ID	BankClient Name	Net worth[$]	Yearswithbank
0	111	Chanel	3500	3
1	222	Steve	29000	4
3	444	Ryan	2000	5
2	333	Mitch	10000	9

[62]:
```
# Set inplace = True to ensure that change has taken place in memory
 bank_client_df
```

[62]:

	Bankclient ID	BankClient Name	Net worth[$]	Yearswithbank
0	111	Chanel	3500	3
1	222	Steve	29000	4
3	444	Ryan	2000	5
2	333	Mitch	10000	9

[63]:
```
# Note that now the change (ordering) took place
 bank_client_df
```

[63]:

	Bankclient ID	BankClient Name	Net worth[$]	Yearswithbank
0	111	Chanel	3500	3
1	222	Steve	29000	4
3	444	Ryan	2000	5

2	333	Mitch	10000	9

11 TASK #11: PERFORM CONCATENATING AND MERGING WITH PANDAS

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

[76]:
```
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],
)
```

[77]:
```
df1
```

[77]:

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

[78]:
```
df2 = pd.DataFrame(
    {
        "A": ["A4", "A5", "A6", "A7"],
        "B": ["B4", "B5", "B6", "B7"],
        "C": ["C4", "C5", "C6", "C7"],
        "D": ["D4", "D5", "D6", "D7"],
    },
    index=[4, 5, 6, 7],
)
```

[79]:
```
df2
```

[79]:

	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

[80]:
```
df3 = pd.DataFrame(
    {
```

```
            "A": ["A8", "A9", "A10", "A11"], "B":
            ["B8", "B9", "B10", "B11"], "C": ["C8",
            "C9", "C10", "C11"], "D": ["D8", "D9",
            "D10", "D11"],
        },
        index=[8, 9, 10, 11],
)
```

[81]:
```
df3
```

[81]:

	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

[83]:
```
frames = [df1, df2, df3]
result = pd.concat(frames)
result
```

[83]:

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

12 TASK#12: PROJECTANDCONCLUDINGREMARKS

- 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'.

[4]:
```
import pandas as pd
import numpy as np

raw_data = {
                        'Bank Client ID':['1', '2', '3', '4', '5'], 'First Name':['Nancy',
                        'Alex','Shep', 'Max', 'Allen'], 'Last Name':['Rob', 'Ali', 'George',
                        'Mitch', 'Steve']

}

Bank_df_1 = pd.DataFrame(raw_data, columns = ['Bank Client ID', 'First Name',␣
  ↪'LastName'])
Bank_df_1
```

[4]:

	BankClient ID	First Name	Last Name
0	1	Nancy	Rob
1	2	Alex	Ali
2	3	Shep	George
3	4	Max	Mitch
4	5	Allen	Steve

[6]:
```
raw_data = {
            'Bank Client ID': ['6','7','8','9','10'],
            'First Name':['Nancy','Alex','Shep','Max','Allen'],
            'Last Name':['Rob','Ali','George','Mitch','Steve']
}
Bank_df_2 = pd.DataFrame(raw_data, columns = ['Bank Client ID','First␣
  ↪Name','LastName'])
Bank_df_2
```

[6]:

	BankClient ID	First Name	Last Name
0	6	Nancy	Rob
1	7	Alex	Ali
2	8	Shep	George
3	9	Max	Mitch
4	10	Allen	Steve

[7]:
```
raw_data = {
            'Bank Client ID': ['1', '2', '3', '4', '5', '6', '7', '8', '9',␣
  ↪'10'],'Annual Salary [$/year]': [25000, 35000, 45000, 48000, 49000, 32000,␣

  ↪33000,34000,23000,22000]
            }
```

```
bank_df_salary = pd.DataFrame(raw_data, columns = ['Bank Client ID','Annual␣
↪Salary[$/year]'])
bank_df_salary
```

[7]:

	BankClient ID	Annual Salary [$/year]
0	1	25000
1	2	35000
2	3	45000
3	4	48000
4	5	49000
5	6	32000
6	7	33000
7	8	34000
8	9	23000
9	10	22000

[8]:
```
bank_df_all = pd.concat([Bank_df_1 , Bank_df_2])
bank_df_all
```

[8]:

	BankClient	ID	First Name	Last Name
0		1	Nancy	Rob
1		2	Alex	Ali
2		3	Shep	George
3		4	Max	Mitch
4		5	Allen	Steve
0		6	Nancy	Rob
1		7	Alex	Ali
2		8	Shep	George
3		9	Max	Mitch
4		10	Allen	Steve

[9]:
```
bank_df_all = pd.merge(bank_df_all, bank_df_salary, on = 'Bank Client ID')
bank_df_all
```

[9]:

	BankClient	ID	First Name	Last Name	Annual Salary [$/year]
0		1	Nancy	Rob	25000
1		2	Alex	Ali	35000
2		3	Shep	George	45000
3		4	Max	Mitch	48000
4		5	Allen	Steve	49000
5		6	Nancy	Rob	32000
6		7	Alex	Ali	33000
7		8	Shep	George	34000
8		9	Max	Mitch	23000
9		10	Allen	Steve	22000

[10]:
```
new_client = {
            'Bank Client ID': ['11'],
            'First Name': ['Ry'],
            'Last Name': ['Aly'],
            'Annual Salary [$/year]' : [1000]}
new_client_df = pd.DataFrame(new_client, columns = ['Bank Client ID','First␣
 ↪Name','LastName','AnnualSalary[$/year]'])
new_client_df
```

[10]:

	BankClient ID	First Name	LastName	AnnualSalary[$/year]
0	11	Ry	Aly	1000

[12]:
```
new_df = pd.concat([bank_df_all,new_client_df])
new_df
```

[12]:

	BankClient	ID	First Name	LastName	AnnualSalary[$/year]
0		1	Nancy	Rob	25000
1		2	Alex	Ali	35000
2		3	Shep	George	45000
3		4	Max	Mitch	48000
4		5	Allen	Steve	49000
5		6	Nancy	Rob	32000
6		7	Alex	Ali	33000
7		8	Shep	George	34000
8		9	Max	Mitch	23000
9		10	Allen	Steve	22000
0		11	Ry	Aly	1000

x