```python
import numpy as np
import pandas as pd
# Basic Pandas Sheet:
print("Version of Pandas: ", pd.__version__)
#help(pd.Series) # Long help shows all about series in pandas
index = ["us","ca","ir","gp"]
data = [1548,1354,6541,9874]
sery = pd.Series(data=data)
print("Convert an array to a Pandas Sery with numeric index: ")
print(sery)
print("conver sery value to string and uppercase, for string data: ")
print("sery.str.upper()")
print("check if value is digit: ")
print("5".isdigit())
sery2 = pd.Series(data=data,index=index)
print("Convert an array to a Pandas Sery with label index: ")
print(sery2)
print("Search value in a sery by numeric index and label index: ")
#print(sery[0]) # old depricated
print(sery2["us"] , " > 1")
print(sery2.iloc[0] , " > 2") # New
print(sery2.loc["us"] , " > 3") # New
print(" > 4") # New
print(sery2.iloc[0:2]) # New
dictunary1 = {"babak":10,"faranak":9,"sahar":7}
sery3 = pd.Series(dictunary1)
print("Convert a Dictunary to a Pandas Sery: ")
print(sery3)
print("Show Keys in a Pandas Sery: ")
print(sery3.keys())
print("Show Values in a Pandas Sery: ")
print(sery3.values)
print("Double an array: ")
print([1,2] * 2)
print("Double a Numpy array Values: ")
print(np.array([1,2]) * 2)
print("Double a Pandas Sery Values: ")
print(sery3 * 2)
dictunary2 = {"sara":1,"faranak":6,"sahar":4}
sery4 = pd.Series(dictunary2)
print("Adding 2 Dictunary Pandas Series with different keys: ")
print(sery4 + sery3)
print("Normalizing 2 Series with different keys for addition: ")
normalizedPandasSeriesAddition =  sery3.add(sery4,fill_value=0)
print(normalizedPandasSeriesAddition)
# DataFrames in Pandas are A Collection of several Series sharing the same Index
print("DataFrames in Pandas are A Collection of several Series sharing the same Index.")
print("Create DataFrame from Data: ")
data = np.random.randint(1000,100001,(4,3))
columns = ["Jan","Feb","Mar"]
```

```
print("DataFrame without Index: ")
dataframe = pd.DataFrame(data)
print(dataframe)
print("DataFrame with Index: ")
dataframe2 = pd.DataFrame(data,index)
print(dataframe2)
print("DataFrame with Index and Columns: ")
dataframe3 = pd.DataFrame(data,index,columns)
print(dataframe3)
print("***Some actions and commands needing consist values, as we used random data those
actions are commented here, use a constant data instead random.***")
print("Information about this DataFrame: ")
print(dataframe3.info())
print("Main Calculation for numeric DataFrame: ")
print(dataframe3.describe())
print("Changing Columns with Index in a DataFrame: ")
print(dataframe3.describe().transpose())
print("Show Columns of the DataFrame: ")
print(dataframe3.columns)
print("Show Indexes of the DataFrame: ")
print(dataframe3.index)
print("Show first rows (head) of the DataFrame: ")
print(dataframe3.head(2))
print("Show last rows (tail) of the DataFrame: ")
print(dataframe3.tail(2))
print("Show a column by its name in the DataFrame: ")
print(dataframe3["Jan"])
print("Show type of a column by its name in the DataFrame: ")
print(type(dataframe3["Jan"]))
print("Show only some of columns in a DataFrame: ")
newColumns = ["Feb","Mar"]
print(dataframe3[newColumns])
print(dataframe3[["Feb","Mar"]])
print("Adding a column to a DataFrame: ")
dataframe3["Jul"] = np.round(100 * dataframe3["Jan"] / dataframe3["Feb"])
print(dataframe3)
print("Drop/Delete a column from a DataFrame: ")
dataframe3 = dataframe3.drop("Jul",axis=1)
print(dataframe3)
print("Show shape of a DataFrame: ")
print(dataframe3.shape)
print("Show shape of a DataFrame on axis 0: ")
print(dataframe3.shape[0])
print("Show shape of a DataFrame on axis 1: ")
print(dataframe3.shape[1])
print("Change index of a DataFrame: ")
print(dataframe3.set_index("Feb"))
print("Reset index of a DataFrame: ")
print(dataframe3.reset_index())
# print("Adding a row to a DataFrame: ")
# one_row = dataframe3.iloc[0]
# print(dataframe3.append(one_row)) # deprecated
```

```
print("Put Condition on a DataFrame: ")
print(dataframe3["Jan"] < 50)
print(dataframe3[dataframe3["Jan"] < 50])
print(dataframe3[dataframe3["Jan"] == 15] | dataframe3[dataframe3["Jan"] == 10])
options = [10,15,20,25,30]
print(dataframe3[dataframe3["Jan"].isin(options)])

def last_two(num):
    return str(num)[-2:]

print("Show last 2 digits of a number (123456789): ")
print(last_two(123456789))
print("Apply function last_two to the DataFrame: ")
print(dataframe3["Jan"].apply(last_two))
print(dataframe3["Jan"].apply(lambda num : str(num)[-2:]))
def add_two(num1,num2):
    return num1 + num2
print(dataframe3[["Jan","Feb"]].apply(lambda dataframe3 :
add_two(dataframe3["Jan"],dataframe3["Feb"]),axis=1))
dataframe3["addition"] = np.vectorize(add_two)(dataframe3["Jan"],dataframe3["Feb"])
print(dataframe3)
print("Sorting a column in the DataFrame Assending: ")
print(dataframe3.sort_values(["Jan","addition"],ascending=True))
print("Max value in a column in dataframe: ",dataframe3["Jan"].max())
print("Index of Max value in a column in dataframe: ",dataframe3["Jan"].idxmax())
print("Get Max value in a column in dataframe by index: ")
print(dataframe3.loc["ca"])
print("Correlation of Values(attention to orib 1 value in rows and index of their
columns): ")
print(dataframe3.corr())
print("Count categorical values: ")
print(dataframe3["addition"].value_counts())
print("get unique values: ")
print(dataframe3["addition"].unique())
print("get number of unique values: ")
print(dataframe3["addition"].nunique())
print("count number of all values: ")
print(dataframe3["addition"].value_counts())
print("Change a value in dataframe(IF EXIST): ")
print(dataframe3["addition"].replace(["Male","Female"],["M","F"]))
print(dataframe3["addition"].map({"Male":"M","Female":"F"}))
print("Show duplicated rows in a DataFrame: ")
print(dataframe3.duplicated())
print("Drop duplicated rows in a DataFrame: ")
print(dataframe3.drop_duplicates())
print("Show values between 2 params in a column in a DataFrame: ")
print(dataframe3["addition"].between(1000,6000,inclusive="both"))
print("2 ways to Get first N rows ordered by largest values in Column in decsending order
in a DataFrame: ")
print(dataframe3.nlargest(2,"Jan"))
print(dataframe3.sort_values("Jan",ascending=False).iloc[0:2])
```

```
print("2 ways to Get first N rows ordered by smallest values in Column in decsending order
in a DataFrame: ")
print(dataframe3.nsmallest(2,"Jan"))
print(dataframe3.sort_values("Jan",ascending=True).iloc[0:2])
print("Show n random sample rows in a DataFrame: ")
print(dataframe3.sample(frac=0.5))
print("Show numpy null values: ", np.nan)
print("Show pandas null values: ", pd.NA)
print("Show pandas null values: ", pd.NaT)
print("2 null or missing values are not equal because of their type and value: ")
print("np.nan == np.nan ",np.nan == np.nan)
print("np.nan is np.nan ",np.nan is np.nan)
print("Check null values in a DataFrame: ")
print(dataframe3.isnull())
print("Check not null values in a column in a DataFrame: ")
print(dataframe3["addition"].notnull())
print("for missing data there are 3 options: ")
print("1) keep data")
print("2) drop data")
print("3) fill data")
print("drop rows with missing values(default axis =0): ")
print(dataframe3.dropna())
print("drop rows having a column with missing values: ")
print(dataframe3.dropna(axis=1))
print("drop rows with missing values if they do not have atleast n(here 1) non value: ")
print(dataframe3.dropna(thresh=1))
print("drop rows with missing values if they do not have atleast 1 non value in subset
column: ")
print(dataframe3.dropna(subset="addition"))
print("Help fo Fill Missing Values(nul values): ")
print("help(dataframe3.fillna)")
print("Fill missing values: ")
print(dataframe3.fillna("New Value!"))
print("Fill missing values in column to fill with same type: ")
print(dataframe3["addition"].fillna(0))
print("Practical Fill missing values: ")
print(dataframe3.fillna(dataframe3.mean()))
print("Filling missing values in a pandas sery by linear interpolation between point above
and point below: ")
print(sery4.interpolate())
print("Grouping in Pandas DataFrame: ")
print("1 column grouping: ")
print(dataframe3.groupby("Jan").mean())
print("show describe for groupby: ")
print(dataframe3.groupby("Jan").describe())
print("2 columns grouping: ")
example_mean = dataframe3.groupby(["Jan","Feb"]).mean()
print(example_mean)
print("show index: ")
print(example_mean.index)
print("base on mean of a column: ")
print(example_mean["addition"])
```

```
print("show levels of indexes: ")
print(example_mean.index.levels)
print("get in a range by index: ")
print("example_mean.loc[(value1_in_grouping,value2_in_grouping)]")
print("Get by Cross Section(xs) in desired Level: ")
print("example_mean.xs(key=value_in_grouped,level=column_name_grouped)")
print("filter by isin (here is empty because of random data not contains these two
values): ")
print(dataframe3[dataframe3["Jan"].isin(["1000","5000"])].groupby(["Feb","addition"]).mean
())
print("swap levels: ")
print(example_mean.swaplevel())
print("sort base on levels: ")
print(example_mean.sort_index(level="Feb",ascending=False))
print("Aggregate function into a dataframe:")
print(dataframe3.agg(["std","mean"]))
print("Aggregate function into a column in a dataframe:")
print(dataframe3.agg(["std","mean"])["Feb"])
print(dataframe3.agg({ "Feb":["max","min","mean"], "Jan":["mean","std"] }))
print("sync column names: ")
dataframe2.columns = dataframe.columns
print("concatenate 2 dataframes(axis=0 > rows, axis=1 > columns): ")
print(pd.concat([dataframe,dataframe2],axis=0))
print("help(pd.merge)")
print("merge 2 dataframes(on must be a shared column_name): ")
print(pd.merge(dataframe,dataframe2,how="inner",on=1))
print(pd.merge(dataframe,dataframe2,how="outer",on=1))
print(pd.merge(left=dataframe,right=dataframe2,how="left",on=1))
dataframe = dataframe.set_index(1)
print(pd.merge(dataframe,dataframe2,left_index=True,right_on=1,how="inner",suffixes=("_reg
","_log")) )

print("help(pd.pivot)")
print("Pivot on DataFrame in pandas")
print(dataframe3.pivot(index="Feb",columns="Jan",values="addition"))
print(pd.pivot_table(dataframe3,index="Jan",aggfunc="sum",values=["Feb","addition"]))

# check time of execution for comparison of two method:
print("check time of execution for comparison of two method:")
```

```
import timeit
setup = '''
import numpy as np
import pandas as pd
data = np.random.randint(1000,100001,(4,3))
columns = ["Jan","Feb","Mar"]
index = ["us","ca","ir","gp"]
dataframe3 = pd.DataFrame(data,index,columns)
def add_two(num1,num2):
    return num1 + num2
'''
statement1 = '''
dataframe3["addition"] = np.vectorize(add_two)(dataframe3["Jan"],dataframe3["Feb"])
'''
statement2 = '''
dataframe3[["Jan","Feb"]].apply(lambda dataframe3 :
add_two(dataframe3["Jan"],dataframe3["Feb"]),axis=1)
'''
print(timeit.timeit(setup=setup,stmt=statement1,number=1000))
print(timeit.timeit(setup=setup,stmt=statement2,number=1000))

# Pandas date conversion:
print("Pandas date conversion: ")
from datetime import datetime
example_date = "01-01-2001"
example_date2 = "12--sep--2000"
print(pd.to_datetime(example_date,dayfirst=True))
print(pd.to_datetime(example_date2,format="%d--%b--%Y"))
print("resample example only for datetime values: ")
print("dataframe3.resample(rule='A').mean()")

print("Read from csv(comma seperated value) file: ")
#dataframe4 = pd.read_csv("path to example.csv")
print('dataframe4 = pd.read_csv("path to example.csv")')

print("write to csv file: ")
print("dataframe3.to_csv('newfile.csv',index=True)")

print("get current directory path: ")
import os
print(os.getcwd)

print("convert/read html to pandas dataframe and vice versa: ")
print("pip install lxml")
print("read html from url or path: ")
print("htmlpage = pd.read_html('url or path')")
print("above htmlpage is an object, you can access and change html tags inside it.")
print("after changing above htmlpage or par of it you can save it as a new html page: ")
print("changedhtmlpage.to_html('new_html.html',index=False)")

print("convert/read excel (only data without macros) to pandas dataframe and vice versa:
")
```

```
print("pip install xlrd ")# for old excel files
print("pip install openpyxl ")# for new excel files
print("read excel from url or path: ")
print("excelfile = pd.read_excel('excelfile.xlsx',sheet_name='First_Sheet')")
print("excelfile2 = pd.ExcelFile('excelfile.xlsx')")
print("excelfile2.sheet_names")
print("excelfile2.keys()")
print("excelfile2.First_Sheet")
print("changedexcelpage.to_excel('new_excel.xlsx',sheet_name='First_Sheet',index= False)")

print("convert/read from database to pandas dataframe and vice versa: ")
print("pip install SQLAlchemy ", )
print("from SQLAlchemy import create_engine")#memory
print("temp_db = create_engine('sqlite:///:memory:')")#memory
import sqlite3#file
temp_db = sqlite3.connect('db.db')#file
df =
pd.DataFrame(data=np.random.randint(low=0,high=100,size=(4,4)),columns=['a','b','c','d'])
df.to_sql(name='new_table',con=temp_db)
new_df = pd.read_sql(sql='new_table',con=temp_db)
result = pd.read_sql_query(sql="select a,c from new_table")
```