LAB ASSIGNMENT - 4

PANDAS LIBRARY

NAME : PRATHAPANI SATWIKA

REG.NO. : 20BCD7160

Q1. Import pandas under the alias pd

CODE :

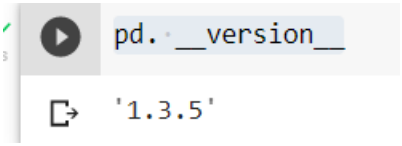
```
import pandas as pd
```

Q2. Print the version of pandas

CODE :

```
pd. __version__
```

OUTPUT :

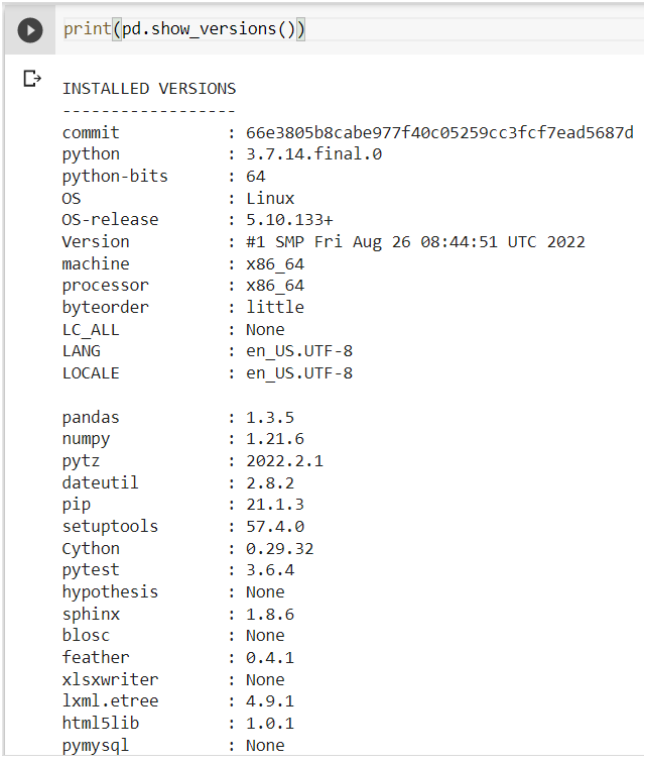


Q3. Print out all the version information of the libraries that are required by the panda's library

CODE :

```
print(pd.show_versions())
```

OUTPUT :



```
psycopg2         : 2.9.3 (dt dec pq3 ext lo64)
jinja2           : 2.11.3
IPython          : 7.9.0
pandas_datareader: 0.9.0
bs4              : 4.6.3
bottleneck       : None
fsspec           : 2022.8.2
fastparquet      : None
gcsfs            : None
matplotlib       : 3.2.2
numexpr          : 2.8.3
odfpy            : None
openpyxl         : 3.0.10
pandas_gbq       : 0.13.3
pyarrow          : 6.0.1
pyxlsb           : None
s3fs             : None
scipy            : 1.7.3
sqlalchemy       : 1.4.41
tables           : 3.7.0
tabulate         : 0.8.10
xarray           : 0.20.2
xlrd             : 1.1.0
xlwt             : 1.3.0
numba            : 0.56.2
None
```

Q4. Create a DataFrame df from this dictionary data which has the index labels

CODE :

```
import numpy as np
data = {'animal': ['cat', 'cat', 'snake', 'dog', 'dog', 'cat', 'snake',
 'cat', 'dog', 'dog'],
'age': [2.5, 3, 0.5, np.nan, 5, 2, 4.5, np.nan, 7, 3],
'visits': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'priority': ['yes', 'yes', 'no', 'yes', 'no', 'no', 'no', 'yes', 'no',
'no']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(data, index = labels)
```

OUTPUT :

```
import numpy as np
data = {'animal': ['cat', 'cat', 'snake', 'dog', 'dog', 'cat', 'snake', 'cat', 'dog', 'dog'],
'age': [2.5, 3, 0.5, np.nan, 5, 2, 4.5, np.nan, 7, 3],
'visits': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'priority': ['yes', 'yes', 'no', 'yes', 'no', 'no', 'no', 'yes', 'no', 'no']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(data, index = labels)
```

```
df
```

	animal	age	visits	priority
a	cat	2.5	1	yes
b	cat	3.0	3	yes
c	snake	0.5	2	no
d	dog	NaN	3	yes
e	dog	5.0	2	no
f	cat	2.0	3	no
g	snake	4.5	1	no
h	cat	NaN	1	yes
i	dog	7.0	2	no
j	dog	3.0	1	no

Q5. Display a summary of the basic information about this DataFrame and its data

CODE :

```
df.info()
df.describe()
```

OUTPUT :

```
[10] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10 entries, a to j
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   animal    10 non-null     object
 1   age       8 non-null      float64
 2   visits    10 non-null     int64
 3   priority  10 non-null     object
dtypes: float64(1), int64(1), object(2)
memory usage: 700.0+ bytes
```

```
df.describe()
```

	age	visits
count	8.000000	10.000000
mean	3.437500	1.900000
std	2.007797	0.875595
min	0.500000	1.000000
25%	2.375000	1.000000
50%	3.000000	2.000000
75%	4.625000	2.750000
max	7.000000	3.000000

Q6. Return the first 3 rows of the DataFrame df.

CODE :

```
df.head(3)
```

OUTPUT :

```
df.head(3)
```

	animal	age	visits	priority
a	cat	2.5	1	yes
b	cat	3.0	3	yes
c	snake	0.5	2	no

Q7.Select just the 'animal' and 'age' columns from the DataFrame df

CODE :

```
df[['animal','age']]
```

OUTPUT :

```
df[['animal','age']]
```

	animal	age
a	cat	2.5
b	cat	3.0
c	snake	0.5
d	dog	NaN
e	dog	5.0
f	cat	2.0
g	snake	4.5
h	cat	NaN
i	dog	7.0
j	dog	3.0

Q8. Select the data in rows [3, 4, 8] and in columns ['animal', 'age'].

CODE :

```
df[['animal','age']].iloc[[3,4,8]]
```

OUTPUT :

```
df[['animal','age']].iloc[[3,4,8]]
```

	animal	age
d	dog	NaN
e	dog	5.0
i	dog	7.0

Q9. Select only the rows where the number of visits is greater than 3.

CODE :

```
df[df['visits']>3]
```

OUTPUT :

```
df[df['visits']>3]
```

	animal	age	visits	priority

Q10. Select the rows where the age is missing, i.e. it is NaN

CODE :

```
df[df['age'].isna()]
```

OUTPUT :

```
df[df['age'].isna()]
```

	animal	age	visits	priority
d	dog	NaN	3	yes
h	cat	NaN	1	yes

Q11. Select the rows where the animal is a cat and the age is less than 3.

CODE :

```
df[(df['animal']=="cat") & (df['age']<3)]
```

OUTPUT :

```
df[(df['animal']=="cat") & (df['age']<3)]
```

	animal	age	visits	priority
a	cat	2.5	1	yes
f	cat	2.0	3	no

Q12. Select the rows the age is between 2 and 4 (inclusive)

CODE :

```
df[(df['age']>=2) & (df['age']<=4)]
```

OUTPUT:

```
df[(df['age']>=2) & (df['age']<=4)]
```

	animal	age	visits	priority
a	cat	2.5	1	yes
b	cat	3.0	3	yes
f	cat	2.0	3	no
j	dog	3.0	1	no

Q13. Change the age in row 'f' to 1.5

CODE ;

```
df.loc[['f'],['age']]=1.5
df
```

OUTPUT :

```
df.loc[['f'],['age']]=1.5
```

```
[23] df
```

	animal	age	visits	priority
a	cat	2.5	1	yes
b	cat	3.0	3	yes
c	snake	0.5	2	no
d	dog	NaN	3	yes
e	dog	5.0	2	no
f	cat	1.5	3	no
g	snake	4.5	1	no
h	cat	NaN	1	yes
i	dog	7.0	2	no
j	dog	3.0	1	no

Q14. Calculate the sum of all visits in df (i.e. find the total number of visits)

CODE :

```
np.sum(df['visits'])
```

OUTPUT :

```
np.sum(df['visits'])

19
```

Q15. Calculate the mean age for each different animal in df.

CODE :

```
df.groupby(['animal']).age.mean()
```

OUTPUT :

```
df.groupby(['animal']).age.mean()

animal
cat      2.333333
dog      5.000000
snake    2.500000
Name: age, dtype: float64
```

Q16. Append a new row 'k' to df with your choice of values for each column.
#Then delete that row to return the original DataFrame

CODE :

```
df.loc['k']=["lion",3,1,"yes"]
df
```

OUTPUT :

```
[26] df.loc['k']=["lion",3,1,"yes"]

df
```

	animal	age	visits	priority
a	cat	2.5	1	yes
b	cat	3.0	3	yes
c	snake	0.5	2	no
d	dog	NaN	3	yes
e	dog	5.0	2	no
f	cat	1.5	3	no
g	snake	4.5	1	no
h	cat	NaN	1	yes
i	dog	7.0	2	no
j	dog	3.0	1	no
k	lion	3.0	1	yes

```
[29] df=df.drop('k')
```

```
df
```

	animal	age	visits	priority
a	cat	2.5	1	yes
b	cat	3.0	3	yes
c	snake	0.5	2	no
d	dog	NaN	3	yes
e	dog	5.0	2	no
f	cat	1.5	3	no
g	snake	4.5	1	no
h	cat	NaN	1	yes
i	dog	7.0	2	no
j	dog	3.0	1	no

Q17. Count the number of each type of animal in df.

CODE :

```
df.groupby(['animal']).animal.count()
```

OUTPUT :

```
df.groupby(['animal']).animal.count()
```

```
animal
cat      4
dog      4
snake    2
Name: animal, dtype: int64
```

Q18. Sort df first by the values in the 'age' in decending order, then by the value in the 'visits' column in ascending order

CODE :

```
df.sort_values(['age','visits'],ascending=[False,True])
```

OUTPUT :

```
df.sort_values(['age','visits'],ascending=[False,True])
```

	animal	age	visits	priority
i	dog	7.0	2	no
e	dog	5.0	2	no
g	snake	4.5	1	no
j	dog	3.0	1	no
b	cat	3.0	3	yes
a	cat	2.5	1	yes
f	cat	1.5	3	no
c	snake	0.5	2	no
h	cat	NaN	1	yes
d	dog	NaN	3	yes

Q19. The 'priority' column contains the values 'yes' and 'no'. Replace this column with a column of boolean values: 'yes' should be True and 'no' should be False

CODE :

```
df['priority']=df['priority'].map({'yes':True,'no':False})

df
```

OUTPUT :

```
[33] df['priority']=df['priority'].map({'yes':True,'no':False})
```

```
df
```

	animal	age	visits	priority
a	cat	2.5	1	True
b	cat	3.0	3	True
c	snake	0.5	2	False
d	dog	NaN	3	True
e	dog	5.0	2	False
f	cat	1.5	3	False
g	snake	4.5	1	False
h	cat	NaN	1	True
i	dog	7.0	2	False
j	dog	3.0	1	False

Q20. In the 'animal' column, change the 'snake' entries to 'python'.

CODE :

```
df['animal']=df['animal'].replace({'snake':"python"})
df
```

OUTPUT ;

```
[37] df['animal']=df['animal'].replace({'snake':"python"})
```

```
df
```

	animal	age	visits	priority
a	cat	2.5	1	True
b	cat	3.0	3	True
c	python	0.5	2	False
d	dog	NaN	3	True
e	dog	5.0	2	False
f	cat	1.5	3	False
g	python	4.5	1	False
h	cat	NaN	1	True
i	dog	7.0	2	False
j	dog	3.0	1	False

Q21.For each animal type and each number of visits, find the mean age. In other words, each row is an animal, each column is a number of visits and the values are the mean ages (hint: use a pivot table).

CODE :

```
table=pd.pivot_table(df,values='age',index=['animal'],columns=['visits'
],aggfunc=np.mean)
```

```
table
```

OUTPUT :

```
[39] table=pd.pivot_table(df,values='age',index=['animal'],columns=['visits'],aggfunc=np.mean)
```

```
table
```

visits	1	2	3
animal			
cat	2.5	NaN	2.25
dog	3.0	6.0	NaN
python	4.5	0.5	NaN

Q22. You have a Data Frame df with a column 'A' of integers. #How do you filter out rows which contain the same integer as the row immediately above?

CODE :

```
df1=pd.DataFrame({'A':[1,2,2,3,4,5,5,5,6,7,7]})
df1.drop_duplicates(subset=None,keep='first',inplace=False)
```

OUTPUT ;

```
df1=pd.DataFrame({'A':[1,2,2,3,4,5,5,5,6,7,7]})
df1.drop_duplicates(subset=None,keep='first',inplace=False)
```

	A
0	1
1	2
3	3
4	4
5	5
8	6
9	7

[

Q23. Given a DataFrame of numeric values, say df = pd.DataFrame(np.random.random(size=(5, 3))) # a 5x3 frame of float values how do you subtract the row mean from each element in the row?

CODE :

```
df=pd.DataFrame(np.random.random(size=(5,3)))
df.sub(df.mean(axis=1),axis=0)
```

OUTPUT :

```
[12] df=pd.DataFrame(np.random.random(size=(5,3)))
```

```
df.sub(df.mean(axis=1),axis=0)
```

	0	1	2
0	-0.404365	0.422881	-0.018516
1	-0.307226	0.406152	-0.098926
2	-0.435510	0.175469	0.260041
3	-0.086662	-0.105491	0.192153
4	-0.158273	0.155755	0.002517

Q24. Suppose you have Data Frame with 10 columns of real numbers. Which column of #numbers has the smallest sum? Return that column's label

CODE :

```
df=pd.DataFrame(np.random.randint(0,10,size=(5,3)))
print(df,"\n")
print("Column labelis:",list(dict(df.sum()).values()).index(min(df.sum(
))))
```

OUTPUT :

```
df=pd.DataFrame(np.random.randint(0,10,size=(5,3)))
print(df,"\n")
print("Column labelis:",list(dict(df.sum()).values()).index(min(df.sum())))
```

```
   0  1  2
0  8  6  9
1  2  6  4
2  1  5  1
3  0  1  9
4  3  8  8

Column labelis: 0
```

Q25. How do you count how many unique rows a Data Frame has (i.e. ignore all rows that are #duplicates)? As input, use a Data Frame of zeros and ones with 10 rows and 3 columns

CODE :

```
df=pd.DataFrame(np.random.randint(0,2,size=(10,3)))
print(df,"\n")
```

```
print("No of unique rows:",len(df.drop_duplicates()))
```

OUTPUT :

```
df=pd.DataFrame(np.random.randint(0,2,size=(10,3)))
print(df,"\n")
print("No of unique rows:",len(df.drop_duplicates()))
```

```
   0  1  2
0  0  1  0
1  0  1  0
2  0  1  0
3  0  1  1
4  1  1  0
5  1  0  1
6  0  1  1
7  0  1  0
8  0  1  0
9  0  1  0

No of unique rows: 4
```

Q26. In the cell below, you have a Data Frame df that consists of 10 columns of floating-point
#numbers. Exactly 5 entries in each row are NaN values.
#For each row of the Data Frame, find the column which contains the third NaN value.
#You should return a Series of column labels: e, c, d, h, d

Example:

nan = np.nan

data = [[0.04, nan, nan, 0.25, nan, 0.43, 0.71, 0.51, nan, nan],

[nan, nan, nan, 0.04, 0.76, nan, nan, 0.67, 0.76, 0.16],

[nan, nan, 0.5 , nan, 0.31, 0.4 , nan, nan, 0.24, 0.01],

[0.49, nan, nan, 0.62, 0.73, 0.26, 0.85, nan, nan, nan],

[nan, nan, 0.41, nan, 0.05, nan, 0.61, nan, 0.48, 0.68]] columns = list('abcdefghij')

df = pd.DataFrame(data, columns=columns) (df.isnull().cumsum(axis=1) == 3).idxmax(axis=1)

0 e

1 c

2 d

3 h

4 d

dtype: object

CODE :

```
nan = np.nan
data = [[0.04,  nan,  nan, 0.25,  nan, 0.43, 0.71, 0.51,  nan,  nan],
        [ nan,  nan,  nan, 0.04, 0.76,  nan,  nan, 0.67, 0.76, 0.16],
        [ nan,  nan, 0.5 ,  nan, 0.31, 0.4 ,  nan,  nan, 0.24, 0.01],
        [0.49,  nan,  nan, 0.62, 0.73, 0.26, 0.85,  nan,  nan,  nan],
        [ nan,  nan, 0.41,  nan, 0.05,  nan, 0.61,  nan, 0.48, 0.68]]

columns = list('abcdefghij')
df = pd.DataFrame(data, columns=columns)
(df.isnull().cumsum(axis=1) == 3).idxmax(axis=1)
```

OUTPUT :

```
nan = np.nan
data = [[0.04,  nan,  nan, 0.25,  nan, 0.43, 0.71, 0.51,  nan,  nan],
        [ nan,  nan,  nan, 0.04, 0.76,  nan,  nan, 0.67, 0.76, 0.16],
        [ nan,  nan, 0.5 ,  nan, 0.31, 0.4 ,  nan,  nan, 0.24, 0.01],
        [0.49,  nan,  nan, 0.62, 0.73, 0.26, 0.85,  nan,  nan,  nan],
        [ nan,  nan, 0.41,  nan, 0.05,  nan, 0.61,  nan, 0.48, 0.68]]

columns = list('abcdefghij')
df = pd.DataFrame(data, columns=columns)
(df.isnull().cumsum(axis=1) == 3).idxmax(axis=1)
```

```
0    e
1    c
2    d
3    h
4    d
dtype: object
```

Q27.A Data Frame has a column of groups 'grps' and column of integer values 'vals':
#For each group, find the sum of the three greatest values.

CODE :

```
df=pd.DataFrame({'grps':list('aaabbcaabcccbbc'),'vals':[12,345,3,1,45,14,4,52,54,23,235,21,57,3,87]})
df.groupby('grps')['vals'].nlargest(3).sum(level=0)
```

OUTPUT :

```
[29] df=pd.DataFrame({'grps':list('aaabbcaabcccbbc'),'vals':[12,345,3,1,45,14,4,52,54,23,235,21,57,3,87]})

df.groupby('grps')['vals'].nlargest(3).sum(level=0)

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Using the level keyword in DataFrame
  """Entry point for launching an IPython kernel.
grps
a    409
b    156
c    345
Name: vals, dtype: int64
```

Q28. The data Frame df constructed below has two integer columns 'A' and 'B'. The values in 'A' are between 1 and 100(inclusive).
#for each group of 10 consecutive integers in 'A' (i.e.'(0,101] (1,201],...)calculate the sum of corresponding values in column'B'.

CODE :

```
df=pd.DataFrame(np.random.RandomState(8765).randint(1,101,size=(100,2))
,columns=["A","B"])
df.groupby(pd.cut(df['A'],np.arange(0,101,10)))['B'].sum()
```

OUTPUT :