Numpy, Pandas
Assignment 7

#NumPy

##1. How to extract all odd numbers from arr?

[ ]:
```
import numpy as np
arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
arr[arr%2!=0]
```

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

##2. Replace all odd numbers in arr with -1 without changing

[ ]:
```
import numpy as np
arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
arr[arr%2!=0]=-1
print(arr)
```

##3. Convert a 1D array to a 2D array with 2 rows

[ ]:
```
import numpy as np
arr=np.arange(10)
arr.reshape(2,5)

#Desired Output:
#> array([[0, 1, 2, 3, 4],
#>        [5, 6, 7, 8, 9]])
```

[ ]: array([[0, 1, 2, 3, 4],
[5, 6, 7, 8, 9]])

##4. Stack arrays a and b vertically

[ ]:
```
import numpy as np
a = np.arange(10).reshape(2,-1)
b = np.repeat(1, 10).reshape(2,-1)


arr_new=np.vstack((a,b))
```

```
print(arr_new)
#Desired Output:

#> array([[0, 1, 2, 3, 4],
#>       [5, 6, 7, 8, 9],
#>       [1, 1, 1, 1, 1],
#>       [1, 1, 1, 1, 1]])
```

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

[ ]:

##5. Stack the arrays a and b horizontally.

```
import numpy as np
a = np.arange(10).reshape(2,-1)
b = np.repeat(1,
10).reshape(2,-1)
arr_new=np.hstack((a,b))
print(arr_new)
#Desired Output:

#> array([[0, 1, 2, 3, 4, 1, 1, 1, 1, 1],
#>       [5, 6, 7, 8, 9, 1, 1, 1, 1, 1]])
```

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

##6.How to get the common items between two python numpy arrays? Get the common items b

[ ]:

```
import numpy as np
a = np.array([1,2,3,2,3,4,3,4,5,6])
b = np.array([7,2,10,2,7,4,9,4,9,8])
```

##7.How to remove from one array those items that exist in another? Q. From array a remove all items present in array b

[ ]:

```
import numpy as np

a = np.array([1,2,3,4,5])
b = np.array([5,6,7,8,9]) a
= a[~np.isin(a, b)] print(a)
```

```
[1 2 3 4]
```

##8. How to get the positions where elements of two arrays match?

Q. Get the positions where elements of a and b match

[ ]:
```
import numpy as np
a = np.array([1,2,3,2,3,4,3,4,5,6])
b = np.array([7,2,10,2,7,4,9,4,9,8])
n =
np.where(a==b
) print(n)
#Desired Output:
#> (array([1, 3, 5, 7]),)
```

(array([1, 3, 5, 7]),)

##9. How to extract all numbers between a given range from a numpy array? Get all items b

[ ]:
```
import numpy as np

an = np.array([2, 6, 1, 9, 10, 3, 27])
ans = np.where(np.logical_and(an>=
an<=10)) print(an[ans])
#Desired Output:

#(array([6, 9, 10]),)
```

[6 9 10]

#Pandas

##Pandas Data Series

##1. Write a Pandas program to convert a dictionary to a Pandas series.

Sample Series:
Original dictionary:
{'a': 100, 'b': 200, 'c': 300, 'd': 400, 'e': 800}

[ ]:

```
Converted series: a
100
b    200
c    300
d    400
```

e 800

dtype: int64

```
import pandas as pd

d={'a': 100, 'b': 200, 'c': 300, 'd': 400, 'e':
res=pd.Series(d)
print(res)
```

a 100
b 200

```
c    300
d    400
e    800
dtype: int64
```

##2. Write a Pandas program to convert a NumPy array to a Pandas series.

[ ]:

```
Sample Series: NumPy array:
[10 20 30 40 50]
Converted Pandas series:
0
1
0
1    20
2    30
```

```
3    40
4    50
```

[ ]:

```
dtype: int64
```

```
np_arr=([10, 20, 30, 40, 50])
nr=pd.Series(np_arr)
print(nr)
```

```
0    10
1    20
2    30
3    40
4    50
dtype: int64
```

##3. Write a Pandas program to change the data type of given a column or a Series.

Sample Series:

```
      O
r
i
g
i
n
a
l
D
a
t
a
S
e
r
i
e
s
:
.
0

1
0
0
1         200
2      python
3      300.12
4         400
dtype: object
C
h
a
n
g
e
t
h
e
s
a
i
d
d
```

atatypetonumeric:0

1000.00

1 200.00
2 NaN
3 300.12
4 400.00
dtype: float64

```
ser = [100, 200, 'python', 300.12, 400]
df = pd.DataFrame(ser, columns=['c1'])
df['c1'] = pd.to_numeric(df['c1'], errors='coerce')
```

```
df
```

[ ]:
```
     c1
0  100.00
1  200.00
2     NaN
3  300.12
4  400.00
```

##4. Write a Pandas program to convert the first column of a DataFrame as a Series.

Sample Output:

```
Original DataFrame
   col1  col2  col3
0     1     4     7
1     2     5     5
2     3     6     8
3     4     9    12
4     7     5     1
5    11     0    11

1st column as a Series:
0     1
1     2
2     3
3     4
4     7
5    11
Name: col1, dtype: int64
<class 'pandas.core.series.Series'>
```

[ ]:
```
import pandas as pd

df = pd.DataFrame({'col1': [1, 2, 3,4,7,11], 'col2': [4, 5,
6,9,5,0],'col3':
[7,5,8,12,1,11]})

# extract the first column of the dataframe as a series
series = df.iloc[:, 0]
series
```

[ ]:
```
0     1
1     2
2     3
3     4
4     7
5    11
Name: col1, dtype: int64
```

##5. Write a Pandas program to convert a given Series to an array.

Sample Output:

Original Data Series:
0 100
1 200
2 python
3 300.12
4 400
dtype: object
Series to an array
['100' '200' 'python' '300.12' '400']
<class 'numpy.ndarray'>

[ ]:
```
ser = [100, 200, 'python', 300.12, 400]
df = pd.Series(ser)
ar=df.to_numpy()
ar
```

[ ]: array([100, 200, 'python', 300.12, 400], dtype=object)

##6 Write a Pandas program to convert Series of lists to one Series.

Sample Output:

Original Series of list
0 [Red, Green, White]
1 [Red, Black]
2 [Yellow]
dtype: object
One Series
0 Red
1 Green
2 White
3 Red
4 Black
5 Yellow
dtype: object

[ ]:
```
s = pd.Series([['Red', 'Green', 'White'],['Red', 'Black'],['Yellow']])
s = s.apply(pd.Series).stack().reset_index(drop=True) print(s)
```

1 Green
2 White
3 Red
4 Black

```
5       Y
e
l
l
o
w
d
t
y
p
e
:
o
b
j
e
c
t
```

[ ]:

```
##7.Write a Pandas program to sort a given Series.
```

```
S
a
m
p
l
e
O
u
t
p
u
t
:
O
r
i
g
i
n
a
l
D
a
t
a
S
e
r
i
e
s
:
0

1
0
0
1       200
2    python
3    300.12
4       400
d
```

type: object

0 100

1 200

3 300.12

4 400

[ ]:

2

python

```
s = pd.Series(['100', '200', 'python', '300.12

srt = pd.Series(s).sort_values()
print(srt)
```

dtype: object

0 100

1 200

3 300.12

4 400

2

python

dtype: object

##Pandas DataFrame

##1.Write a Pandas program to create a dataframe from a dictionary and display it.

Sample data: {'X':[78,85,96,80,86],

'Y':[84,94,89,83,86],'Z':[86,97,96,72,83]}

E

xpect

edOutput:X

```
    X   Y
Z 078   84
86
```

```
    Y
    Z
0   78  84  86
1   85  94  97
2   96  89  96
3   80  83  72
4   86  86  83
```

```
dict={'X':[78,85,96,80,86],  'Y':[84,94,89,83,86
nwd=pd.DataFrame(dic
t) print(nwd)
```

1 85 94 97
2 96 89 96
3 80 83 72
4 86 86 83

##2. Write a Pandas program to create and display a DataFrame from a specified dictionary data which has the index labels.

Sample Python dictionary data and list labels:
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew' 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

[21]:
```
import numpy as np

dt= {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
'Michael',
'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

nw=pd.DataFrame(dt,index = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
'j']) print(nw)
```

attempts name qualify score
a 1 Ana

s
t
a
s
i
a

y

e
s

1
2
.
5

	name	score	attempts	qualify
a	Anastasia	12.5	1	yes
b	Dima	9.0	3	no
c	Katherine	16.5	2	yes
d	James	NaN	3	no
e	Emily	9.0	2	no
f	Michael	20.0	3	yes
g	Matthew	14.5	1	yes
h	Laura	NaN	1	no
i	Kevin	8.0	2	no
j	Jonas	19.0	1	yes

##3. Write a Pandas program to get the first 3 rows of a given DataFrame.

Sample Python dictionary data and list labels:

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew'

'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

Expected Output:

First three rows of the data frame:

	attempts	name	qualify	score
a	1	Anastasia	yes	12.5
b	3	Dima	no	9.0
c	2	Katherine	yes	16.5

[ ]:
```
nw.head(3)
```

[ ]:

	name	score	attempts	qualify
a	Anastasia	12.5	1	yes
b	Dima	9.0	3	no
c	Katherine	16.5	2	yes

##4.Write a Pandas program to select the 'name' and 'score' columns from the following DataFrame.

Sample Python dictionary data and list labels:

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew' 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

Expected Output:

Select specific columns:

```
a   Anastasia    12.5
b         Dima     9.0
c    Katherine    16.5
...
h        Laura     NaN
i        Kevin     8.0
j        Jonas    19.0
              name  score
```

[ ]:
```
df = pd.DataFrame(dt,columns=['name', 'city'],index = ['a', 'b', 'c', 'd', 'e',␣
 ↪'f', 'g', 'h', 'i', 'j'])
df
```

[ ]:

	name	city

a Anastasia NaN b Dima NaN c Katherine NaN d James NaN

e	Emily	NaN
f	Michael	NaN
g	Matthew	NaN
h	Laura	NaN
i	Kevin	NaN
j	Jonas	NaN

##5. Write a Pandas program to select the specified columns and rows from a given data frame.

Sample Python dictionary data and list labels:

Select 'name' and 'score' columns in rows 1, 3, 5, 6 from the following data frame.

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew' 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

[ ]:

Expected Output:

Select specific columns and rows: score qu

	a l i f y	the exami natio n is great er than 2:
b	9.0 no	name
d	NaN no	score
f	20.0 yes	attem pts
g	14.5 yes	qualif y

```
print(nw.iloc[[1, 3, 5, 6], [1, 3]])
```

	s c o r e	q u a l i f y
b	9 . 0	n o
d	NaN	no
f	20.0	yes
g	14.5	yes

##6. Write a Pandas program to select the rows where the number of attempts in the examination is greater than 2.

Sample Python dictionary data and list labels:

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew' 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

Expected Output:

b	Dima	9.0	3	no
d	James	NaN	3	no
f	Michael	20.0	3	yes

Number of attempts in

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

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James',
'Emily',
'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no',
'no', 'yes']}

df = pd.DataFrame(exam_data,index = ['a', 'b', 'c', 'd', 'e', 'f',
'g', 'h',
'i', 'j'])
df = df[df['attempts'] > 2]
print(df)
```

```
b     Dima    9.0         3       no
d    James    NaN         3       no
f  Michael   20.0         3      yes
```

```
          name  score  attempts qualify
```

[22]:

##7. Write a Pandas program to select the rows where the score is missing, i.e. is NaN.

Sample Python dictionary data and list labels:

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew' 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',

Expected Output:

R

o

w

s

w

h

e

r

e

s

c

o

r

e

i

s

m

i

s

s i n g : a t t e m p t s n a m e q u a l i f y s c o r e

d	3	James	no	NaN
h	1	Laura	no	NaN

```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James',
'Emily',␣
'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

df = pd.DataFrame(exam_data,index = ['a', 'b', 'c', 'd', 'e', 'f',
'g', 'h',␣
'i', 'j'])

df = df[df['score'].isnull()]
```

```
print(df)
```

[10]:

	name	score	attempts	qualify
d	James	NaN	3	no
h	Laura	NaN	1	no

##8. Write a Pandas program to select the rows the score is between 15 and 20 (inclusive).

Sample Python dictionary data and list labels:

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew' 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',

Expected Output:

Row wise where score between 15 and 20 (inclusive) : attempts name qualify score

c	2	Katherine	yes	16.5
f	3	Michael	yes	20.0
j	1	Jonas	yes	19.0

```
exam_data = {'name': ['Anastasia',
'Emily',␣
'Michael', 'Matthew', 'Laura', 'Kevin
'score': [12.5, 9, 16.5, np.nan, 9,
'attempts': [1, 3, 2, 3, 2, 3, 1, 1,
'qualify': ['yes', 'no', 'yes', 'no', 'no
'no', 'yes']}

df = pd.DataFrame(exam_data,index
'g', 'h',␣
'i', 'j'])
df =
df[df['score'].between(15,20)]
print(df)
```

	name	score	attempts	qualify
c	Katherine	16.5	2	yes
f	Michael	20.0	3	yes
j	Jonas	19.0	1	yes

##9. Write a Pandas program to select the rows where number of attempts in the examination is less than 2 and score greater than 15.

Sample Python dictionary data and list labels:

```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew' 'score':
[12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
```

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',

Expected Output:

Number of attempts in the examination is less than 2 and score greater than 15 : name score attempts qualify

j Jonas 19.0 1 yes

[12]:

```
exam_data = {'name': ['Anastasia',
'Emily',␣
'Michael', 'Matthew', 'Laura', 'Kevin
'score': [12.5, 9, 16.5, np.nan, 9,
'attempts': [1, 3, 2, 3, 2, 3, 1, 1,
'qualify': ['yes', 'no', 'yes', 'no', 'no
'no', 'yes']}

df = pd.DataFrame(exam_data,index
'g', 'h',␣
'i', 'j'])

print(df[(df['attempts'] < 2) & (df['s
```

n a m e s c o r e a t t e m p t s q u a l i f y

[25]:

```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James',
'Emily',␣
'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no',␣
'no', 'yes']}
```

J o n a s 1 9 . 0 1

y
e
s

##10. Write a Pandas program to change the score in row ‘d’ to 11.5.

Sample Python dictionary data and list labels:

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew' 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

Expected Output:

a	1	Anastasia	yes	12.5
b	3	Dima	no	9.0
c	2	Katherine	yes	16.5
...				
i	2	Kevin	no	8.0
j	1	Jonas	yes	19.0

C
h
a
n
g
e
t
h
e
s
c
o
r
e
i
n
r
o
w
'
d
'
t
o
1
1
.
5
:
a
t
t
e
m
p
t
s
n
a
m
e
q
u
a
l
i
f
y
s
c
o
r
e

```
df = pd.DataFrame(exam_data,index = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',␣
 ↪'i', 'j'])

df.loc['d','score'] = 11.5
print(df)
```

	name	score	attempts	qualify
a	Anastasia	12.5	1	yes
	b	9.0	3	no
	Dima			
c	Katherine	16.5	2	yes
d	James	11.5	3	no
e	Emily	9.0	2	no
f	Michael	20.0	3	yes
	g	14.5	1	yes
	Matthew			
h	Laura	NaN	1	no
i	Kevin	8.0	2	no
j	Jonas	19.0	1	yes

[15]:

##11. Write a Pandas program to calculate the sum of the examination attempts by the students.

Sample Python dictionary data and list labels:

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew' 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

Expected Output:

Sum of the examination attempts by the students: 19

```
exam_data = {'name': ['Anastasia',
'Emily',␣
↪'Michael', 'Matthew', 'Laura', 'Kevin
'score': [12.5, 9, 16.5, np.nan, 9,
'attempts': [1, 3, 2, 3, 2, 3, 1, 1,
'qualify': ['yes', 'no', 'yes', 'no', 'no
↪'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g'

df = pd.DataFrame(exam_data , index

print("Sum of the examination atte
print(df['attempts'].sum())
```

Sum of the examination attempts by the students:
19

##12. Write a Pandas program to calculate the mean of all students' scores. Data is stored in a dataframe.

Sample Python dictionary data and list labels:

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew' 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

Expected Output:

[24]:

```
Mean score for each different student in data frame:
13.5625
```

```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
'Michael', 'Matthew', 'Laura', 'Kevin'
'score': [12.5, 9, 16.5, np.nan, 9,
'attempts': [1, 3, 2, 3, 2, 3, 1, 1,
'qualify': ['yes', 'no', 'yes', 'no', 'no'
'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g'

df = pd.DataFrame(exam_data , index

print("Mean Score by the students:")
print(df['score'].mean())
```

Mean Score by the students:

13.5625

##13. Write a Pandas program to replace the 'qualify' column contains the values 'yes' and 'no' with True and False.

Sample Python dictionary data and list labels:

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew' 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

Expected Output:

[26]:

```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',
'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
```

```
e
5
0
.....
i    2    Kevin    False    8.0
j    1    Jonas    True    19.0
```

Replace the 'qualify' column contains the values 'yes' and 'no' with T rue and False:

```
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

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

	name	score	attempts	qualify
a	Anastasia	12.5	1	True
	b	9.0	3	False
	Dima			
c	Katherine	16.5	2	True
d	James	11.5	3	False
e	Emily	9.0	2	False
f	Michael	20.0	3	True
	g	14.5	1	True
	Matthew			
h	Laura	NaN	1	False
i	Kevin	8.0	2	False
j	Jonas	19.0	1	True

##14. Write a Pandas program to change the name 'James' to 'Suresh' in name column of the DataFrame.

Sample Python dictionary data and list labels:
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew' 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

Expected Output:
C

[27]:
```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James',
'Emily',
'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df ['name'] = df ['name'].replace ('James',
'Suresh') print(df)
```

n
a
m
e
'
J
a
m
e
s

'to\?Suresh\?:attempts na

mequalify score

b	3	Dima	no	9.0
.....				
i	2	Kevin	no	8.0
j	1	Jonas	yes	19.0
a	1	Anastasia	yes	12.5

	name	score	attempts	qualify
a	Anastasia	12.5	1	True

b	Dima	9.0	3	False
c	Katherine	16.5	2	True
d	Suresh	11.5	3	False
e	Emily	9.0	2	False
f	Michael	20.0	3	True
g	Matthew	14.5	1	True
h	Laura	NaN	1	False
i	Kevin	8.0	2	False
j	Jonas	19.0	1	True

##15. Write a Pandas program to delete the 'attempts' column from the DataFrame.

Sample Python dictionary data and list labels:

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew' 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

Expected Output:

Dele

[28]:

t
e
t
h
e
'
a
t
t
e
m
p
t
s
'
c
o
l
u
m
n
f
r
o
m
t
h
e
d
a
t
a
f

r
a
m
e
:
n
a
m
e
q
u
a
l
i
f
y

s
c
o
r
e

a

A
n
a
s
t
a
s

i
a

y
e
s

1
2
.
5
b

D
i
m
a

n
o

9
.
0
.....
i Kevin no 8.0
j Jonas yes 19.0

```
exam_data = {'name': ['Anastasia',
'Emily',␣
'Michael', 'Matthew', 'Laura', 'Kevin
'score': [12.5, 9, 16.5, np.nan, 9,
'attempts': [1, 3, 2, 3, 2, 3, 1, 1,
'qualify': ['yes', 'no', 'yes', 'no', 'no
'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g'

df = pd.DataFrame(exam_data , index
df.pop('attempts')
print(df)
```

	name	score	qualify
a	Anastasia	12.5	yes
b	Dima	9.0	no
c	Katherine	16.5	yes
d	James	NaN	no
e	Emily	9.0	no
f	Michael	20.0	yes
g	Matthew	14.5	yes

h	Laura	NaN	no
i	Kevin	8.0	no
j	Jonas	19.0	yes

##16. Write a Pandas program to insert a new column in existing DataFrame.

Sample Python dictionary data and list labels:

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew' 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19], 'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

Expected Output:

New

[2]:
```
import numpy as np
import pandas as pd


exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James',
'Emily',
'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
color=['red','blue','yellow','red','green','pink','blue','orange','green','r
ed'] df = pd.DataFrame(exam_data)
df['Color'] = color
print(df)
```

Series the 'color' column a

	attempts	name	qualify	score	color
b	3	Dima	no	9.0	Blue
.....					
i	2	Kevin	no	8.0	Green
j	1	Jonas	yes	19.0	Red
a	1	Anastasia	yes	12.5	Red

	name	score	attempts	qualify	Color
0	Anastasia	12.5	1	yes	red
1	Dima	9.0	3	no	blue
2	Katherine	16.5	2	yes	yellow
3	James	NaN	3	no	red
4	Emily	9.0	2	no	green
5	Michael	20.0	3	yes	pink
6	Matthew	14.5	1	yes	blue
7	Laura	NaN	1	no	orange
8	Kevin	8.0	2	no	green
9	Jonas	19.0	1	yes	red

##17. Write a Pandas program to rename columns of a given DataFrame

Sample data:

[3]:
```
df = pd.DataFrame({'col1': [1, 2,3], 'col2': [4, 5, 6], 'col3': [7,8,9]}) print(df)
df.rename(columns={'col1': 'Column1', 'col2': 'Column2', 'col3': 'Column3'},␣
  ↪inplace=True)
print("after renaming")
print(df)
```

```
Original DataFrame
   col1  col2  col3
0     1     4     7
1     2     5     8
2     3     6     9

   0  1  4  7
0  1  2  5  8
1  2  3  6  9
2  3  4  7  9
New DataFrame after rename
```

naming columns: Column1 Column2 Column3

```
   col1  col2  col3
0     1     4     7
1     2     5     8
2     3     6     9
after renaming
```

```
0  1  4  7
1  2  5  8
2  3  6  9
   Column1 Column2 Column3
```

[5]:

##18. Write a Pandas program to select rows from a given DataFrame based on values in some columns.

Sample data:

Original DataFrame

col1 col2 col3
0 1 4 7
1 4 5 8
2 3 6 9
3 4 7 0
4 5 8 1

Rows for column1 value == 4

col1 col2 col3
1 4 5 8
3 4 7 0

```
df = pd.DataFrame({'col1': [1, 4,3,4,5], 'col2': [4, 5, 6,7,8], 'col3':
  [7,8,9,0,1]})
```

```
result = df.loc[df['col1'] == 4]
print(result)
```

[9]:

```
   col1  col2  col3
1     4     5     8
3     4     7     0
```

##19. Write a Pandas program to add one row in an existing DataFrame.

Sample data:

```
Original DataFrame
   col1  col2  co
```

l
3
0 1 4 7
1 4 5 8
2 3 6 9
3 4 7 0
4 5 8 1
A
fteraddonerow:

c
o
l
1

c
o
l
2

c
o
l
3

0 1 4 7
1 4 5 8
2 3 6 9
3 4 7 0
4 5 8 1
5 10 11 12

```
df = pd.DataFrame({'col1': [1, 4,3,4,5], 'col2
  [7,8,9,0,1]})
new_row = {'col1': 10, 'col2': 11, 'col3': 12

df=df.append(new_row,ignore_index=Tru
e) print(df)
```

	col1	col2	col3
0	1	4	7
1	4	5	8
2	3	6	9
3	4	7	0
4	5	8	1
5	10	11	12

<ipython-input-9-faeb192e4fff>:4: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
df=df.append(new_row,ignore_index=True)

##20. Write a Pandas program to replace all the NaN values with Zero's in a column of a dataframe.

Sample data:

	attempts	name	qualify	score
0	1	Anastasia	yes	12.5
1	3	Dima	no	9.0
2	2	Katherine	yes	16.5
.....				
8	2	Kevin	no	8.0
9	1	Jonas	yes	19.0

Original DataFrame

0	1	Anastasia	yes	12.5
1	3	Dima	no	9.0

[10]:

```
import pandas as pd
import numpy as np
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James',
'Emily',␣
'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no',␣
'yes']}
df =
pd.DataFrame(exam_da
ta) print("Original ")
print(df)
df = df.fillna(0)
print("after
replacing") print(df)
```

r a m e r e p l a c i n g a l l N a N w i t h 0

:
attempts

name

qualify

score

	name	score	attempts	qualify
0	Anastasia	12.5	1	yes
1	Dima	9.0	3	no
2	Katherine	16.5	2	yes
3	James	NaN	3	no
4	Emily	9.0	2	no
5	Michael	20.0	3	yes
6	Matthew	14.5	1	yes
7	Laura	NaN	1	no
8	Kevin	8.0	2	no
9	Jonas	19.0	1	yes

Original

	name	score	attempts	qualify
0	Anastasia	12.5	1	yes
1	Dima	9.0	3	no
2	Katherine	16.5	2	yes
3	James	0.0	3	no
4	Emily	9.0	2	no
5	Michael	20.0	3	yes
6	Matthew	14.5	1	yes
7	Laura	0.0	1	no
8	Kevin	8.0	2	no
9	Jonas	19.0	1	yes

after replacing

##21. Write a Pandas program to count the NaN values in one or more columns in DataFrame.

Sample data:
Original DataFrame

	attempts	name	qualify	score
0	1	Anastasia	yes	12.5
1	3	Dima	no	9.0
2	2	Katherine	yes	16.5
3	3	James	no	NaN
4	2	Emily	no	9.0
5	3	Michael	yes	20.0
6	1	Matthew	yes	14.5
7	1	Laura	no	NaN
8	2	Kevin	no	8.0
9	1	Jonas	yes	19.0

[11]:

Number of NaN val

uesinoneormo

recolumn

```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James',
'Emily',␣
'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no',␣
'no', 'yes']}
df = pd.DataFrame(exam_data)
print("Number of NaN values in one or more columns:",
 df.isnull().values.sum())
```

Number of NaN values in one or more columns: 2

##22. Write a Pandas program to drop a list of rows from a specified DataFrame.

Sample data:

Original DataFrame

	col1	col2	col3
0	1	4	7
1	4	5	8

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

[12]:

New DataFrame after removing 2nd & 4th rows:

	col1	col2	col3
0	1	4	7
1	4	5	8
3	4	7	0

```
df = pd.DataFrame({'col1': [1, 4,3,4,5], 'col2
  [7,8,9,0,1]})
rows_to_drop = [2, 4]

df = df.drop(rows_to_drop)
print(df)
```

```
   col1  col2  col3
0     1     4     7
1     4     5     8
3     4     7     0
```

##23. . Write a Pandas program to convert DataFrame column type from string to datetime.

```
Sample data:
String Date:
0    3/11/2000
1    3/12/2000
2    3/13/2000
dtype: object
```

[13]:

```
Original DataFrame(string to datetime):
0 2000-03-11
1 2000-03-12
2 2000-03-13
```

```
import pandas as pd
df = pd.DataFrame({'date': ['3/11/2
']}) print('Original DataFrame:')
print(df)
```

```
print(df)
```

```
Original DataFrame:
d
a
t
e
0

3
/
1
1
/
2
0
0
0
1

3
```

[14]:
```
import pandas as pd

d = {'col1': [1, 2, 3, 4, 7], 'col2': [4, 5, 6, 9, 5], 'col3': [7, 8,
12, 1,␣
↪11]}
df =
pd.DataFrame(d)
print(df)

print("Row where col1 has max value:")
print(df['col1'].argmax())

print("Row where col2 has max value:")
print(df['col2'].argmax())

print("Row where col3 has max value:")
print(df['col3'].argmax())
```

```
e
0
2
0
0
0
-
0
3
-
1
1
1 2000-03-12
2 2000-03-13
```

##24. Write a Pandas program to find the row for where

the value of a given column is maximum.

Sample Output:

```
0    1    4    7
1    2    5    8
2    3    6   12
3    4    9    1
4    7    5   11
Row 4 where col1 has maximum value:
Row 3 where col2 has maximum value:
Row 2 where col3 has maximum value:
```

Original DataFrame

```
   col1  col2  col3
0     1     4     7
1     2     5     8
```

```
2      3      6     12
3      4      9      1
4      7      5     11
Ro  where col1    has     value:
w                  max
4
Ro  where col2    has     value:
w                  max
3
Ro  where col3    has     value:
w                  max
2
```

##25. Write a Pandas program to get the datatypes of columns of a DataFrame.

Sample data:

Original DataFrame:

```
    attempts          name qualify   score

1          3         Dima       no     9.0
.......
8          2        Kevin       no     8.0
9          1        Jonas      yes    19.0
       0          1   Anastasia       yes    12.5
```

[15]:

```
Data types of the columns of the said DataFrame:
attempts      int64
name         object
qualify      object
score
```

```
import pandas as pd
import numpy as np

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James',
'Emily',
'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no',
'no', 'yes']}

df =
pd.DataFrame(exam_da
ta) print(df.dtypes)
```

```
name          object
score        float64
attempts       int64
qualify       object
dtype: object
```

##26. Write a Pandas program to group by the first column and get second column as lists in rows.

Sample data:

Original DataFrame
col1 col2
0 C1 1
1 C1 2
2 C2 3
3 C2 3
4 C2 4
5 C3 6
6 C2 5

[16]:

Grouponthecol1:col1
C1 [1, 2]
C2 [3, 3, 4, 5]
C3 [6]
Name: col2, dtype: object

```
import pandas as pd

df = pd.DataFrame({'col1': ['C1', 'C
'col2':
[1, 2, 3, 3, 4, 6, 5]})
print(df)

df = df.groupby('col1')['col2'].apply(
 print("after grouping")
print(df)
```

0 C1 1
1 C1 2
2 C2 3
3 C2 3
4 C2 4
5 C3 6
6 C2 5

col1 col2

```
after grouping
col1
C1          [1, 2]
C2      [3, 3, 4, 5]
C3             [6]
Name: col2, dtype: object
```

##27 Write a Pandas program to count number of columns of a DataFrame.

Sample Output:

```
Original DataFrame
   col1  col2  col3
0   1      4     7
```

1 2 5 8
2 3 6 12
3 4 9 1
4 7 5 11

[17]:
```
Number of columns:
3
```

```
import pandas as pd
d = {'col1': [1, 2, 3, 4, 7], 'col2': [2, 3, 6, 12, 1, 11]}
df = pd.DataFrame(d)
num_cols = len(df.columns)
print(num_cols)
```

3

##28. Write a Pandas program to get first n records of a DataFrame.

Sample Output:

[19]:
```
Original DataFrame
   col1  col2
0     1     2
1     2     3
2     3     6
3     4     9
4     7     5
```

l
3
0 1 4 7
1 2 5 5
2 3 6 8
3 4 9 12
4 7 5 1
5 11 0 11

F
i
r
s
t
3
r
o
w
s
o
f
t
h
e
s
a
i

d
D
a
t
a
F
r
a
m
e
'
.
c
o
l
1
c
o
l
2
c
o
l
3
0 1 4 7
1 2 5 5
2 3 6 8

```
import pandas as pd

d = {'col1': [1, 2, 3, 4, 7, 11], 'c
'col3': [7, 5, 8,
12, 1,11]}
df = pd.DataFrame(d)

print(df)
print("first 3
rows")

s1 =
df.iloc[0:3]
print(s1)
```

col1 col2 col3

0 1 4 7
1 2 5 5
2 3 6 8
3 4 9 12
4 7 5 1
5 11 0 11
first 3 rows
c
o
l
1

c
o
l
2

c
o
l
3
0

1

[21]:

4

7
1 2 5 5
2 3 6 8

##29. Write a Pandas program to get last n records of a DataFrame.

Sample Output:

O
r
i
g
i
n
a
l
D
a
t
a
F
r
a
m
e
c
o
l
1
c
o

l
2
c
o
l
3
0 1 4 7
1 2 5 5
2 3 6 8
3 4 9 12
4 7 5 1
5 11 0 11

L
a
s
t
3
r
o
w
s
o
f
t
h
e
s

a
i
d
D
a
t
a
F
r
a
m
e
'
:
c
o
l
1
c
o
l
2
c
o
l
3
3 4 9 12
4 7 5 1
5 11 0 11

```
import pandas as pd

d = {'col1': [1, 2, 3, 4, 7, 11], 'c
'col3': [7, 5, 8,␣
12, 1,11]}
df = pd.DataFrame(d)

print(df)
print("last 3
rows")

s1 =
df.iloc[3:]
print(s1)
```

	col1	col2	col3
0	1	4	7
1	2	5	5
2	3	6	8
3	4	9	12
4	7	5	1
5	11	0	11

last 3 rows

c

o

l

1

c

o

l

2

c

o

l

3

3

4

9

1

2

4 7 5 1

5 11 0 11

##30. Write a Pandas program to get topmost n records within each group of a DataFrame.

[1]:

Sample Output:

O

r

i

g

i

n

a

l

D

a

t

a

F

r

a

m

e

c

o

l

1

c

o

l

2

c

o

l

3

0 1 4 7

1 2 5 5

2 3 6 8
3 4 9 12
4 7 5 1
5 11 0 11

topmo
st n
record
s
within
each
group
of a
DataFr
ame:
col1
col2
col3
5 11 0 11
4 7 5 1
3 4 9 12
c
o
l
1
c
o
l
2
c
o
l
3
3
4
9
1
2
2 3 6 8
1 2 5 5
4 7 5 1
c
o
l
1
c
o
l
2
c
o
l
3
3
4
9
1
2
5 11 0 11
2 3 6 8

```
import pandas as pd
d = {'col1': [1, 2, 3, 4, 7, 11], 'c
'col3': [7, 5, 8,␣
12, 1,11]}
df =
pd.DataFrame(data=d)
print("Original
DataFrame") print(df)
print("topmost n records within each
DataFrame:") df1 = df.nlargest(3, '
print(df1)
df2 = df.nlargest(3, 'col2')
print(df2)
df3 = df.nlargest(3, 'col3')
print(df3)
```

```
Original DataFrame
    col1    col2    col3
```

	0	1	4	7
	1	2	5	5
	2	3	6	8
	3	4	9	12
	4	7	5	1
	5	11	0	11

	col1	col2	col3
5	11	0	11
4	7	5	1
3	4	9	12

	col1	col2	col3
3	4	9	12
2	3	6	8
1	2	5	5

	col1	col2	col3
3	4	9	12
5	11	0	11
2	3	6	8

topmost n records within each group of a DataFrame:

##31. Write a Pandas program to add a prefix or suffix to all columns of a given DataFrame.

Sample Output:
Original DataFrame
W X Y Z
0 68 78 84 86
1 75 85 94 97
2 86 96 89 96
3 80 80 83 72
4 66 86 86 83

Add prefix:

0 68 78 84 86
1 75 85 94 97
2 86 96 89 96
3 80 80 83 72
4 66 86 86 83

A_W A_X A_Y A_Z

Add suffix:

0 68 78 84 86
1 75 85 94 97
2 86 96 89 96
3 80 80 83 72
4 66 86 86 83
W_1 X_1 Y_1 Z_1

[2]:
```
import pandas as pd
df = pd.DataFrame({'W':[68,75,86,80,66],'X':[78,85,96,80,86], 'Y':
[84,94,89,83,86],'Z':[86,97,96,72,83]});
print("
DataFrame")
print(df)
print("Add prefix:")
print(df.add_prefix("A_"))
print("Add suffix:")
print(df.add_suffix("_1"))
```

```
DataFrame
    W   X   Y
Z 068  78  84
86
1   75  85  94  97
2   86  96  89  96
3   80  80  83  72
4   66  86  86  83
```

```
  A_W  A_X  A_Y  A_Z
0   68      84   86
       78
1   75      94   97
       85
2   86      89   96
       96
3   80      83   72
       80
4   66      86   83
       86
Add suffix:
  W_1  X_1  Y_1  Z_1
0   68      84   86
       78
1   75      94   97
       85
2   86      89   96
       96
3   80      83   72
       80
4   66      86   83
       86
Add prefix:
```

##32. Write a Pandas program to convert continuous values of a column in a given DataFrame to categorical.

Input:

{ 'Name': ['Alberto Franco','Gino Mcneill','Ryan Parkes', 'Eesha Hinton', 'Syed Wharton'], 'Age': [18, 22, 40, 50, 80, 5] }

Output:

Age group:

0 kids
1 adult
2 elderly
3 adult
4 elderly
5 kids

Name: age_groups, dtype: category
Categories (3, object): [kids < adult < elderly]

[3]:

```
import pandas as pd
df = pd.DataFrame({'name': ['Albert
Parkes',
'Eesha Hinton', 'Syed Wharton', 'Ki
 85, 50, 80,
5]})
print(" DataFrame:")
print(df)
print('Age group:')
df["age_groups"] = pd.cut(df["age"], b
["kids",
 "adult", "elderly"])
print(df["age_groups"])
```

```
 DataFrame:
              name  age
0   Alberto Franco   18
1      Gino Mcneill   22
2       Ryan Parkes   85
3      Eesha Hinton   50
4      Syed          80
Wharton
5     Kierra Gentry    5
Age group:
0          kids
1         adult
2       elderly
3         adult
4       elderly
5          kids
Name:                dtype: category
age_groups,
Categories (3, object): ['kids' < 'adult' < 'elderly']
```

##33. Write a Pandas program to append rows to an existing DataFrame and display the combined data.

Test Data:

tudent_data1

	student_id	name	marks
0	S1	Danniella Fenton	200
1	S2	Ryder Storey	210
2	S3	Bryce Jensen	190
3	S4	Ed Bernal	222
4	S5	Kwame Morin	199

New Row(s)

student_id	S6	
name	Scarlette	Fisher
marks	205	

dtype: object

[5]:
```
import pandas as pd
std1 = pd.DataFrame({'student_id': ['S1', 'S2', 'S3', 'S4',
                                    'S5'],'name':
['Danniella Fenton', 'Ryder Storey', 'Bryce Jensen', 'Ed
                                    Bernal', 'Kwame
Morin'],'marks': [200, 210, 190, 222, 199]})

s6 = pd.Series(['S6', 'Scarlette Fisher', 205], index=['student_id',
'name',
'marks'])

dct = [{'student_id': 'S6', 'name': 'Scarlette Fisher', 'marks': 203},
   {'student_id': 'S7', 'name': 'Bryce Jensen', 'marks':
                                207}]

print("Original
DataFrames:")
print(std1)
print("Dictionary:")
print(s6)
comb_data = std1.append(dct,
ignore_index=True, sort=False) print("Combined
Data:")
print(comb_data)
```

Original DataFrames:

	student_id	name	marks
0	S1	Danniella Fenton	200
1	S2	Ryder Storey	210
2	S3	Bryce Jensen	190
3	S4	Ed Bernal	222
4	S5	Kwame Morin	199

Dictionary:

student_id	S6
name	Scarlette Fisher
marks	205

dtype: object

Combined Data:

	student_id	name	marks
0	S1	Danniella Fenton	200
1	S2	Ryder Storey	210
2	S3	Bryce Jensen	190
3	S4	Ed Bernal	222
4	S5	Kwame Morin	199
5	S6	Scarlette Fisher	203
6	S7	Bryce Jensen	207

<ipython-input-5-b267fc108719>:14: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  comb_data = std1.append(dct, ignore_index=True, sort=False)

##34 Write a Pandas program to join the two given dataframes along rows and merge with another dataframe along the common column id.

Test Data:
student_data1:

	student_id	name	marks
0	S1	Danniella Fenton	200
1	S2	Ryder Storey	210
2	S3	Bryce Jensen	190
3	S4	Ed Bernal	222

student_data2:

	student_id	name	marks
0	S4	Scarlette Fisher	201
1	S5	Carla Williamson	200
2	S6	Dante Morse	198
3	S7	Kaiser William	219
4	S8	Madeeha Preston	201
4	S5	Kwame Morin	199

[6]:

exam_data:

	student_id	exam_id
0	S1	23
1	S2	45
2	S3	12
3	S4	67

4	S5	21
5	S7	55
6	S8	33
7	S9	14
8	S10	56
9	S11	83
10	S12	88
11	S13	12

```
import pandas as pd
std1 = pd.DataFrame({'student_id': ['S1', 'S2', 'S3', 'S4',
'S5'],'name':␣
↪['Danniella Fenton', 'Ryder Storey', 'Bryce Jensen', 'Ed Bernal',
  'Kwame␣
↪Morin'],'marks': [200, 210, 190, 222, 199]})

std2 = pd.DataFrame({'student_id': ['S4', 'S5', 'S6', 'S7', 'S8'],
'name':␣
↪['Scarlette Fisher', 'Carla Williamson', 'Dante Morse', 'Kaiser
  William',␣
↪'Madeeha Preston'], 'marks': [201, 200, 198, 219, 201]})

exam_data = pd.DataFrame({'student_id': ['S1', 'S2', 'S3', 'S4',
'S5', 'S7',␣
↪'S8', 'S9', 'S10', 'S11', 'S12', 'S13'],'exam_id': [23, 45, 12, 67,
  21, 55,␣
↪33, 14, 56, 83, 88, 12]})
```

```
print("Original DataFrames:")
 print(std1)
print(std2)
 print(exam_d
 ata)

print("Join  first  two
 dataframes") res =
 pd.concat([std1, std2])
 print(res)

print("joining res and exam_data")
finaldata  =  pd.merge(res,  exam_data,
 on='student_id') print(finaldata)
```

```
Original DataFrames:
  student_id                 name  marks
0         S1   Danniella Fenton      200
1         S2        Ryder Storey     210
2         S3        Bryce Jensen     190
3         S4           Ed Bernal     222
4         S5         Kwame Morin     199
  student_id                   name  marks
0         S4   Scarlette Fisher       201
1         S5   Carla Williamson       200
2         S6         Dante Morse      198
3         S7      Kaiser William      219


   student_id  exam_id
0          S1       23
1          S2       45
2          S3       12
3          S4       67
4          S5       21
5          S7       55
6          S8       33
7          S9       14
8         S10       56
9         S11       83
10        S12       88
11        S13       12
  Join first two  dataframes
  student_id                 name  marks
0         S1   Danniella Fenton      200
1         S2        Ryder Storey     210
2         S3        Bryce Jensen     190
3         S4           Ed Bernal     222
4         S5         Kwame Morin     199
0         S4   Scarlette Fisher       201
```

4	S8	Madeeha Preston	201

1	S5	Carla Williamson	200	
2	S6	Dante Morse	198	
3	S7	Kaiser William	219	
4	S8	Madeeha Preston	201	

joining res and exam_data

	student_id	name	marks	exam_id
0	S1	Danniella Fenton	200	23
1	S2	Ryder Storey	210	45
2	S3	Bryce Jensen	190	12
3	S4	Ed Bernal	222	67
4	S4	Scarlette Fisher	201	67
5	S5	Kwame Morin	199	21
6	S5	Carla Williamson	200	21
7	S7	Kaiser William	219	55
8	S8	Madeeha Preston	201	33