1.How to extract all odd numbers from arr?

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

```
[1 3 5 7 9]
```

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

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

```
[ 0 -1  2 -1  4 -1  6 -1  8 -1]
```

3. Convert a 1D array to a 2D array with 2 rows [ ]

In [3]:
```
import numpy as np
arr=np.arange(10)

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

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

4. Stack arrays a and b vertically

In [4]:
```
import numpy as np
a = np.arange(10).reshape(2,-1)
b = np.repeat(1, 10).reshape(2,-1)

#Desired Output:

#> array([[0, 1, 2, 3, 4],
#>        [5, 6, 7, 8, 9],
#>        [1, 1, 1, 1, 1],
#>        [1, 1, 1, 1, 1]])
np.vstack((a,b))
```

Out[4]:
```
array([[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

In [5]:
```
import numpy as np
a = np.arange(10).reshape(2,-1)
b = np.repeat(1, 10).reshape(2,-1)

#Desired Output:

#> array([[0, 1, 2, 3, 4, 1, 1, 1, 1, 1],
#>        [5, 6, 7, 8, 9, 1, 1, 1, 1, 1]])
np.hstack((a,b))
```

Out[5]:
```
array([[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?

In [6]:
```
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])
c = np.intersect1d(a,b)
print(c)
```

```
[2 4]
```

7.How to remove from one array those items that exist in another?

In [8]:
```
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?

In [9]:
```
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])

#Desired Output:

#> (array([1, 3, 5, 7]),)
np.where(a==b)
```

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

9. How to extract all numbers between a given range from a numpy array?

In [10]:
```
import numpy as np

a = np.array([2, 6, 1, 9, 10, 3, 27])
indices = np.where((a >=5) & (a <= 10))
a[indices]
#Desired Output:

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

Out[10]:
```
array([ 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

In [11]:
```
import pandas as pd
dic = {'a': 100, 'b': 200, 'c': 300, 'd': 400, 'e': 800}
pd.Series(dic)
```

Out[11]:
```
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 10 1 20 2 30 3 40 4 50 dtype: int64

In [12]:
```
import numpy as np
import pandas as pd
a = np.array([10,20 ,30, 40, 50])
pd.Series(a)
```

Out[12]:
```
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: Original Data Series: 0 100 1 200 2 python 3 300.12 4 400 dtype: object Change the said data type to numeric: 0 100.00 1 200.00 2 NaN 3 300.12 4 400.00 dtype: float64

In [13]:
```
import pandas as pd
s1 = pd.Series(['100', '200', 'python', '300.12', '400'])
print("Original Data Series:")
print(s1)
print("Change the said data type to numeric:")
s2 = pd.to_numeric(s1, errors='coerce')
print(s2)
```

```
Original Data Series:
0       100
1       200
2    python
3    300.12
4       400
dtype: object
Change the said data type to numeric:
0    100.00
1    200.00
2       NaN
3    300.12
4    400.00
dtype: float64
```

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'>

In [14]:
```
import pandas as pd
d = {'col1': [1, 2, 3, 4, 7, 11], 'col2': [4, 5, 6, 9, 5, 0], 'col3': [7, 5, 8, 12, 1,11]}
df = pd.DataFrame(data=d)
print("Original DataFrame")
print(df)
s1 = df.iloc[:,0]
print("\n1st column as a Series:")
print(s1)
print(type(s1))
```

```
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'>
```

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'>

In [15]:
```
import pandas as pd
import numpy as np
s1 = pd.Series(['100', '200', 'python', '300.12', '400'])
print("Original Data Series:")
print(s1)
print("Series to an array")
a = s1.values
print(a)
print(type(a))
```

```
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'>
```

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

In [16]:
```
import pandas as pd
s = pd.Series([
    ['Red', 'Green', 'White'],
    ['Red', 'Black'],
    ['Yellow']])
print("Original Series of list")
print(s)
s = s.apply(pd.Series).stack().reset_index(drop=True)
print("One Series")
print(s)
```

```
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
```

7.Write a Pandas program to sort a given Series. Sample Output: Original Data Series: 0 100 1 200 2 python 3 300.12 4 400 dtype: object 0 100 1 200 3 300.12 4 400 2 python dtype: object

In [17]:
```
s = pd.Series(['100', '200', 'python', '300.12', '400'])
srt = pd.Series(s).sort_values()
print(srt)
```

```
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]} Expected Output: X Y Z

0 78 84 86

1 85 94 97

2 96 89 96

3 80 83 72

4 86 86 83

In [18]:
```
import pandas as pd
df = pd.DataFrame({'X':[78,85,96,80,86], 'Y':[84,94,89,83,86],'Z':[86,97,96,72,83]});
print(df)
```

```
    X   Y   Z
0  78  84  86
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', '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'] Expected Output: attempts name qualify score
   a 1 Anastasia yes 12.5
   b 3 Dima no 9.0
   ....
   i 2 Kevin no 8.0
   j 1 Jonas yes 19.0

In [19]:
```
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']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print(df)
```

```
        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', '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'] 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

In [20]:
```
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']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("First three rows of the data frame:")
print(df.iloc[:3])
```

```
First three rows of the data frame:
        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', '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'] Expected Output: Select specific columns:
name score
a Anastasia 12.5
b Dima 9.0
c Katherine 16.5
...
h Laura NaN
i Kevin 8.0
j Jonas 19.0

In [21]:
```
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']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("Select specific columns:")
print(df[['name', 'score']])
```

```
Select specific columns:
        name  score
a  Anastasia   12.5
b       Dima    9.0
c  Katherine   16.5
d      James    NaN
e      Emily    9.0
f    Michael   20.0
g    Matthew   14.5
h      Laura    NaN
i      Kevin    8.0
j      Jonas   19.0
```

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', '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'] Expected Output: Select specific columns and rows: score qualify b 9.0 no d NaN no f 20.0 yes g 14.5 yes

In [22]:
```
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']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("Select specific columns and rows:")
print(df.iloc[[1, 3, 5, 6], [1, 3]])
```

```
Select specific columns and rows:
   score qualify
b    9.0      no
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', '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'] Expected Output: Number of attempts in the examination is greater than 2: name score attempts qualify b Dima 9.0 3 no d James NaN 3 no f Michael 20.0 3 yes

In [23]:
```
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']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("Number of attempts in the examination is greater than 2:")
print(df[df['attempts'] > 2])
```

```
Number of attempts in the examination is greater than 2:
      name  score  attempts qualify
b     Dima    9.0         3      no
d    James    NaN         3      no
f  Michael   20.0         3     yes
```

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', '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', labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'] Expected Output: Rows where score is missing: attempts name qualify score d 3 James no NaN h 1 Laura no NaN

In [24]:
```
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']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("Rows where score is missing:")
print(df[df['score'].isnull()])
```

```
Rows where score is missing:
    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', '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', labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'] Expected Output: Rows 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

In [25]:
```
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']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("Rows where score between 15 and 20 (inclusive):")
print(df[df['score'].between(15, 20)])
```

```
Rows where score between 15 and 20 (inclusive):
        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', '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', labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'] 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

In [26]:
```
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']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
print("Number of attempts in the examination is less than 2 and score greater than 15 :")
print(df[(df['attempts'] < 2) & (df['score'] > 15)])
```

```
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
```

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', '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'] Expected Output: Change the score in row 'd' to 11.5:
    attempts name qualify score
    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

In [27]:
```
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']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("\nOriginal data frame:")
print(df)
print("\nChange the score in row 'd' to 11.5:")
df.loc['d', 'score'] = 11.5
print(df)
```

```
Original data frame:
        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

Change the score in row 'd' to 11.5:
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d      James   11.5         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
```

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', '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'] Expected Output: Sum of the examination attempts by the students:
    19

In [28]:
```
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']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("\nSum of the examination attempts by the students:")
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', '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'] Expected Output: Mean score for each different student in data frame:
    13.5625

In [29]:
```
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']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("\nMean score for each different student in data frame:")
print(df['score'].mean())
```

```
Mean score for each different student in data frame:
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', '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'] Expected Output: Replace the 'qualify' column contains the values 'yes' and 'no' with T rue and False:
    attempts name qualify score
    a 1 Anastasia True 12.5
    b 3 Dima False 9.0
    .....
    i 2 Kevin False 8.0
    j 1 Jonas True 19.0

In [30]:

```
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']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
print("Original rows:")
print(df)
print("\nReplace the 'qualify' column contains the values 'yes' and 'no'  with True and  False:")
df['qualify'] = df['qualify'].map({'yes': True, 'no': False})
print(df)
```

```
Original rows:
        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

Replace the 'qualify' column contains the values 'yes' and 'no'  with True and  False:
        name  score  attempts  qualify
a  Anastasia   12.5         1     True
b       Dima    9.0         3    False
c  Katherine   16.5         2     True
```

```
d      James    NaN         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
```

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', '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'] Expected Output: Change the name 'James' to ?Suresh?:
    attempts name qualify score
    a 1 Anastasia yes 12.5
    b 3 Dima no 9.0
    .....
    i 2 Kevin no 8.0
    j 1 Jonas yes 19.0

In [32]:
```
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)
```

```
        name  score  attempts  qualify
a  Anastasia   12.5         1     True
b       Dima    9.0         3    False
c  Katherine   16.5         2     True
d     Suresh    NaN         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', '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'] Expected Output: Delete the 'attempts' column from the data frame:
    name qualify score
    a Anastasia yes 12.5
    b Dima no 9.0
    .....
    i Kevin no 8.0
    j Jonas yes 19.0

In [33]:
```
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']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
print("Original rows:")
print(df)
print("\nDelete the 'attempts' column from the data frame:")
df.pop('attempts')
print(df)
```

```
Original rows:
        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

Delete the 'attempts' column from the data frame:
        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', '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'] Expected Output: New DataFrame after inserting the 'color' column
attempts name qualify score color
a 1 Anastasia yes 12.5 Red
b 3 Dima no 9.0 Blue
.....
i 2 Kevin no 8.0 Green
j 1 Jonas yes 19.0 Red

In [34]:

```
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']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
print("Original rows:")
print(df)
color = ['Red','Blue','Orange','Red','White','White','Blue','Green','Green','Red']
df['color'] = color
print("\nNew DataFrame after inserting the 'color' column")
print(df)
```

```
Original rows:
        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

New DataFrame after inserting the 'color' column
        name  score  attempts qualify   color
a  Anastasia   12.5         1     yes     Red
b       Dima    9.0         3      no    Blue
c  Katherine   16.5         2     yes  Orange
```

```
d      James    NaN         3      no     Red
e      Emily    9.0         2      no   White
f    Michael   20.0         3     yes   White
g    Matthew   14.5         1     yes    Blue
h      Laura    NaN         1      no   Green
i      Kevin    8.0         2      no   Green
j      Jonas   19.0         1     yes     Red
```

17. Write a Pandas program to rename columns of a given DataFrame Sample data: Original DataFrame col1 col2 col3 0 1 4 7 1 2 5 8 2 3 6 9 New DataFrame after renaming columns: Column1 Column2 Column3 0 1 4 7 1 2 5 8 2 3 6 9

In [35]:
```
import pandas as pd
d = {'col1': [1, 2, 3], 'col2': [4, 5, 6], 'col3': [7, 8, 9]}
df = pd.DataFrame(data=d)
print("Original DataFrame")
print(df)
df.columns = ['Column1', 'Column2', 'Column3']
df = df.rename(columns={'col1': 'Column1', 'col2': 'Column2', 'col3': 'Column3'})
print("New DataFrame after renaming columns:")
print(df)
```

```
Original DataFrame
   col1  col2  col3
0     1     4     7
1     2     5     8
2     3     6     9
New DataFrame after renaming columns:
   Column1  Column2  Column3
0        1        4        7
1        2        5        8
2        3        6        9
```

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 colum1 value == 4 col1 col2 col3 1 4 5 8 3 4 7 0

In [36]:
```
import pandas as pd
import numpy as np
d = {'col1': [1, 4, 3, 4, 5], 'col2': [4, 5, 6, 7, 8], 'col3': [7, 8, 9, 0, 1]}
df = pd.DataFrame(data=d)
print("Original DataFrame")
print(df)
print('Rows for colum1 value == 4')
print(df.loc[df['col1'] == 4])
```

```
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 colum1 value == 4
   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 col3 0 1 4 7 1 4 5 8 2 3 6 9 3 4 7 0 4 5 8 1 After add one row: 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

In [37]:
```
import pandas as pd
import numpy as np
d = {'col1': [1, 4, 3, 4, 5], 'col2': [4, 5, 6, 7, 8], 'col3': [7, 8, 9, 0, 1]}
df = pd.DataFrame(data=d)
print("Original DataFrame")
print(df)
print('After add one row:')
df2 = {'col1': 10, 'col2': 11, 'col3': 12}
df = df.append(df2, ignore_index=True)
print(df)
```

```
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
After add one row:
   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

/var/folders/n3/2c719lpd7ps4sg23bj71nfsh0000gn/T/ipykernel_4616/385303473.py:9: FutureWarning: The fram
e.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat in
stead.
  df = df.append(df2, 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: Original DataFrame

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

New DataFrame replacing all NaN with 0: 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

In [38]:
```
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_data)
print("Original DataFrame")
print(df)
df =  df.fillna(0)
print("\nNew DataFrame replacing all NaN with 0:")
print(df)
```

```
Original DataFrame
        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
```

```
New DataFrame replacing all NaN with 0:
        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
```

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

Number of NaN values in one or more columns: 2

In [39]:
```
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_data)
print("Original DataFrame")
print(df)
print("\nNumber of NaN values in one or more columns:")
print(df.isnull().values.sum())
```

```
Original DataFrame
        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

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

New DataFrame after removing 2nd & 4th rows: col1 col2 col3 0 1 4 7 1 4 5 8 3 4 7 0

In [40]:
```
import pandas as pd
import numpy as np
d = {'col1': [1, 4, 3, 4, 5], 'col2': [4, 5, 6, 7, 8], 'col3': [7, 8, 9, 0, 1]}
df = pd.DataFrame(d)
print("Original DataFrame")
print(df)
print("New DataFrame after removing 2nd & 4th rows:")
df = df.drop(df.index[[2,4]])
print(df)
```

```
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
New DataFrame after removing 2nd & 4th rows:
   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

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

In [41]:
```
import pandas as pd
import numpy as np
s = pd.Series(['3/11/2000', '3/12/2000', '3/13/2000'])
print("String Date:")
print(s)
r = pd.to_datetime(pd.Series(s))
df = pd.DataFrame(r)
print("Original DataFrame (string to datetime):")
print(df)
```

```
String Date:
0    3/11/2000
1    3/12/2000
2    3/13/2000
dtype: object
Original DataFrame (string to datetime):
           0
0 2000-03-11
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: 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 Row where col1 has maximum value: 4 Row where col2 has maximum value: 3 Row where col3 has maximum value: 2

In [42]:
```
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(data=d)
print("Original DataFrame")
print(df)
print("Row where col1 has maximum value:")
print(df['col1'].argmax())
print("Row where col2 has maximum value:")
print(df['col2'].argmax())
print("Row where col3 has maximum value:")
print(df['col3'].argmax())
```

```
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
Row where col1 has maximum value:
4
Row where col2 has maximum value:
3
Row where col3 has maximum value:
2
```

25. Write a Pandas program to get the datatypes of columns of a DataFrame. Sample data: Original DataFrame: attempts name qualify score 0 1 Anastasia yes 12.5 1 3 Dima no 9.0 ..... 8 2 Kevin no 8.0 9 1 Jonas yes 19.0 Data types of the columns of the said DataFrame: attempts int64 name object qualify object score float64 dtype: object

In [43]:
```
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_data)
print("Original DataFrame:")
print(df)
print("Data types of the columns of the said DataFrame:")
print(df.dtypes)
```

```
Original DataFrame:
        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
Data types of the columns of the said DataFrame:
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

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

In [44]:
```
  import pandas as pd
df = pd.DataFrame( {'col1':['C1','C1','C2','C2','C2','C3','C2'], 'col2':[1,2,3,3,4,6,5]})
print("Original DataFrame")
print(df)
df = df.groupby('col1')['col2'].apply(list)
print("\nGroup on the col1:")
print(df)
```

```
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

Group on the col1:
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

Number of columns: 3

In [45]:
```
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(data=d)
print("Original DataFrame")
print(df)
print("\nNumber of columns:")
print(len(df.columns))
```

```
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

Number of columns:
3
```

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

First 3 rows of the said DataFrame': col1 col2 col3 0 1 4 7 1 2 5 5 2 3 6 8

In [46]:
```
import pandas as pd
d = {'col1': [1, 2, 3, 4, 7, 11], 'col2': [4, 5, 6, 9, 5, 0], 'col3': [7, 5, 8, 12, 1,11]}
df = pd.DataFrame(data=d)
print("Original DataFrame")
print(df)
print("\nFirst 3 rows of the said DataFrame':")
df1 = df.head(3)
print(df1)
```

```
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

First 3 rows of the said DataFrame':
   col1  col2  col3
0     1     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: 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

Last 3 rows of the said DataFrame': col1 col2 col3 3 4 9 12 4 7 5 1 5 11 0 11

In [47]:
```
import pandas as pd
d = {'col1': [1, 2, 3, 4, 7, 11], 'col2': [4, 5, 6, 9, 5, 0], 'col3': [7, 5, 8, 12, 1,11]}
df = pd.DataFrame(data=d)
print("Original DataFrame")
print(df)
print("\nFirst 3 rows of the said DataFrame':")
df1 = df.tail(3)
print(df1)
```

```
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

First 3 rows of the said DataFrame':
   col1  col2  col3
3     4     9    12
4     7     5     1
5    11     0    11
```

30. Write a Pandas program to get topmost n records within each group of a DataFrame. 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 topmost n records within each group of a DataFrame: 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 4 7 5 1 col1 col2 col3 3 4 9 12 5 11 0 11 2 3 6 8

In [48]:

```
import pandas as pd
d = {'col1': [1, 2, 3, 4, 7, 11], 'col2': [4, 5, 6, 9, 5, 0], 'col3': [7, 5, 8, 12, 1,11]}
df = pd.DataFrame(data=d)
print("Original DataFrame")
print(df)
print("\ntopmost n records within each group of a DataFrame:")
df1 = df.nlargest(3, 'col1')
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

topmost n records within each group of a DataFrame:
   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

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: A_W A_X A_Y A_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 suffix: W_1 X_1 Y_1 Z_1 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

In [49]:
```
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("Original DataFrame")
print(df)
print("\nAdd prefix:")
print(df.add_prefix("A_"))
print("\nAdd suffix:")
print(df.add_suffix("_1"))
```

```
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:
   A_W  A_X  A_Y  A_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 suffix:
   W_1  X_1  Y_1  Z_1
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
```

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]

In [50]:
```
import pandas as pd
df = pd.DataFrame({
    'name': ['Alberto Franco','Gino Mcneill','Ryan Parkes', 'Eesha Hinton', 'Syed Wharton', 'Kierra Gent
      'age': [18, 22, 85, 50, 80, 5]
})
print("Original DataFrame:")
print(df)
print('\nAge group:')
df["age_groups"] = pd.cut(df["age"], bins = [0, 18, 65, 99], labels = ["kids", "adult", "elderly"])
print(df["age_groups"])
```

```
Original DataFrame:
             name  age
0  Alberto Franco   18
1    Gino Mcneill   22
2     Ryan Parkes   85
3    Eesha Hinton   50
4    Syed Wharton   80
5   Kierra Gentry    5

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']
```

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

In [51]:
```
import pandas as pd
student_data1  = 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'])


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

print("Original DataFrames:")
print(student_data1)
print("\nDictionary:")
print(s6)
combined_data =  student_data1.append(dicts, ignore_index=True, sort=False)
print("\nCombined Data:")
print(combined_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

/var/folders/n3/2c719lpd7ps4sg23bj71nfsh0000gn/T/ipykernel_4616/907218911.py:17: FutureWarning: The fra
me.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat i
nstead.
  combined_data =  student_data1.append(dicts, 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 4 S5 Kwame Morin 199 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

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

In [52]:

```
import pandas as pd
student_data1 = 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]})

student_data2 = pd.DataFrame({
        'student_id': ['S4', 'S5', 'S6', 'S7', 'S8'],
        'name': ['Scarlette Fisher', 'Carla Williamson', 'Dante Morse', 'Kaiser William', 'Madeeha Prest
        '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(student_data1)
print(student_data2)
print(exam_data)

print("\nJoin first two said dataframes along rows:")
result_data = pd.concat([student_data1, student_data2])
print(result_data)

print("\nNow join the said result_data and df_exam_data along student_id:")
final_merged_data = pd.merge(result_data, exam_data, on='student_id')
print(final_merged_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
  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
   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 said dataframes along rows:
  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
1         S5  Carla Williamson    200
2         S6       Dante Morse    198
3         S7    Kaiser William    219
4         S8   Madeeha Preston    201

Now join the said result_data and df_exam_data along student_id:
```

	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

In [ ]: