In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
info=np.array(['P','a','n','d','a','s'])
a=pd.Series(info)
print(a)
```

```
0    P
1    a
2    n
3    d
4    a
5    s
dtype: object
```

In [3]:
```
x=('python','pandas')
df=pd.DataFrame(x)
print(df)
```

```
        0
0  python
1  pandas
```

In [4]:
```
df=pd.Series()
print(df)
```

```
Series([], dtype: float64)

C:\Users\samir\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: DeprecationWarning: The default dtype for empty Series will be 'object'
instead of 'float64' in a future version. Specify a dtype explicitly to silence this warning.
  """Entry point for launching an IPython kernel.
```

In [5]:
```
info={'x':0.,'y':1.,'z':3.}
a=pd.Series(info)
print(a)
```

```
x    0.0
y    1.0
z    3.0
dtype: float64
```

In [6]:
```
x=pd.Series(4,index=[0,1,2,3])
print(x)
```

```
0    4
1    4
2    4
3    4
dtype: int64
```

In [7]:
```
x=pd.Series([1,2,3],index=['a','b','c'])
print(x[1])
```

```
2
```

In [8]:
```
x=pd.Series(data=[2,4,6,8])
y=pd.Series(data=[11.9,22,23,44.9],index=['a','b','c','d'])
print(x.index)
print(x.values)
print(y.index)
print(y.values)
```

```
RangeIndex(start=0, stop=4, step=1)
[2 4 6 8]
Index(['a', 'b', 'c', 'd'], dtype='object')
[11.9 22.  23.  44.9]
```

In [9]:
```
a=pd.Series(data=[2,3,4,5])
b=pd.Series(data=[2.4,6,7.9],index=['a','b','c'])
print(a.shape)
print(b.shape)
```

```
(4,)
(3,)
```

In [10]:
```
a=pd.Series(data=[1,2,3,4])
b=pd.Series(data=[2,9.8,89],
index=['a','b','c'])
print(a.ndim,b.ndim)
print(a.size,b.size)
print(a.nbytes,b.nbytes)
```

```
1 1
4 3
32 24
```

In [11]:
```
a=pd.Series(data=[1,2,3,np.NaN])
b=pd.Series(data=[4.3,5.6,9.6],index=['x','y','z'])
c=pd.Series()
print(a.empty,b.empty,c.empty)
print(a.hasnans,b.hasnans,c.hasnans)
print(len (a),len (b))
print(a.count(),b.count())
```

```
False False True
True False False
4 3
3 3

C:\Users\samir\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: DeprecationWarning: The default dtype for empty Series will be 'object'
instead of 'float64' in a future version. Specify a dtype explicitly to silence this warning.
  This is separate from the ipykernel package so we can avoid doing imports until
```

In [12]:
```
a=pd.Series(['Java','C','C++',np.NaN])
a.map({'Java':'Core'})
```

Out[12]:
```
0    Core
1     NaN
2     NaN
3     NaN
dtype: object
```

In [13]:
```
a=pd.Series(['Java','C','C++',np.NaN])
a.map({'Java':'Core'})
a.map('I like {}'.format,na_action='ignore')
```

Out[13]:
```
0    I like Java
1       I like C
2     I like C++
3            NaN
dtype: object
```

In [14]:
```
a=pd.Series(['java','C','C++',np.NaN])
a.map({'java':'Core'})
a.map('I like {}'.format)
a.map('i like {}'.format,na_action='ignore')
```

Out[14]:
```
0    i like java
1       i like C
2     i like C++
3            NaN
dtype: object
```

In [15]:
```
print(np.std([4,7,2,1,6,3]))
print(np.std([6,9,15,2,-17,15,4]))
```

```
2.1147629234082532
10.077252622027656
```

In [16]:
```
info={'Name':['parker','Smith','john','williams'],
'sub1_Marks':[52,38,42,37],
'sub2_marks':[41,35,29,36]}
data=pd.DataFrame(info)
data.std()
```

Out[16]:
```
sub1_Marks    6.849574
sub2_marks    4.924429
dtype: float64
```

In [17]:
```
s=pd.Series(['a','b','c'],
name='vals')
s.to_frame()
```

Out[17]:

	vals
0	a
1	b
2	c

In [18]:
```
emp=['Parker','Smith','john','Williams']
id=[102,107,109,114]
emp_series=pd.Series(emp)
id_series=pd.Series(id)
frame={'Emp':emp_series,'ID':id_series}
result=pd.DataFrame(frame)
print(result)
```

```
        Emp   ID
0    Parker  102
1     Smith  107
2      john  109
3  Williams  114
```

In [19]:
```
pd.unique(pd.Series([2,1,3,3]))
pd.unique(pd.Series([pd.Timestamp('20160101'),
pd.Timestamp('20160101')]))
```

Out[19]:
```
array(['2016-01-01T00:00:00.000000000'], dtype='datetime64[ns]')
```

In [20]:
```
pd.unique(pd.Index([pd.Timestamp('20160101',tz='US/Eastern'),
pd.Timestamp('20160101',tz='US/Eastern')]))
```

Out[20]:
```
DatetimeIndex(['2016-01-01 00:00:00-05:00'], dtype='datetime64[ns, US/Eastern]', freq=None)
```

In [21]:
```
index = pd.Index([2, 1, 1, np.nan, 3])
index.value_counts()
```

Out[21]:
```
1.0    2
3.0    1
2.0    1
dtype: int64
```

In [22]:
```
index = pd.Index([2, 1, 1, np.nan, 3])
a = pd.Series([2, 1, 1, np.nan, 3])
a.value_counts(normalize=True)
```

Out[22]:
```
1.0    0.50
3.0    0.25
2.0    0.25
dtype: float64
```

In [23]:
```
index = pd.Index([1, 3, 2, 2, 1, np.nan])
index.value_counts()
a = pd.Series([1, 3, 2, 2, 1, np.nan])
a.value_counts(bins=2)
```

Out[23]:
```
(0.997, 2.0]    4
(2.0, 3.0]      1
dtype: int64
```

In [24]:
```
index = pd.Index([1, 3, 2, 2, 1, np.nan])
index.value_counts()
a = pd.Series([1, 3, 2, 2, 1, np.nan])
a.value_counts(dropna=False)
```

Out[24]:
```
2.0    2
1.0    2
NaN    1
3.0    1
dtype: int64
```

In [25]:
```
df = pd.DataFrame()
print (df)
```

```
Empty DataFrame
Columns: []
Index: []
```

In [26]:
```
info = {'ID' :[101, 102, 103],'Department' :['B.Sc','B.Tech','M.Tech',]}
df = pd.DataFrame(info)
print (df)
```

```
    ID Department
0  101       B.Sc
1  102     B.Tech
2  103     M.Tech
```

In [27]:
```
info = {'one' : pd.Series([1, 2, 3, 4, 5, 6], index=['a', 'b', 'c', 'd', 'e', 'f']),
'two' : pd.Series([1, 2, 3, 4, 5, 6, 7, 8], index=['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])}
d1 = pd.DataFrame(info)
print (d1)
```

```
   one  two
a  1.0    1
b  2.0    2
c  3.0    3
d  4.0    4
e  5.0    5
f  6.0    6
g  NaN    7
h  NaN    8
```

In [28]:
```
info = {'one' : pd.Series([1, 2, 3, 4, 5, 6], index=['a', 'b', 'c', 'd', 'e', 'f']),
'two' : pd.Series([1, 2, 3, 4, 5, 6, 7, 8], index=['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])}
d1 = pd.DataFrame(info)
print (d1 ['one'])
```

```
a    1.0
b    2.0
c    3.0
d    4.0
e    5.0
f    6.0
g    NaN
h    NaN
Name: one, dtype: float64
```

In [29]:
```
info = {'one' : pd.Series([1, 2, 3, 4, 5], index=['a', 'b', 'c', 'd', 'e']),
'two' : pd.Series([1, 2, 3, 4, 5, 6], index=['a', 'b', 'c', 'd', 'e', 'f'])}
df = pd.DataFrame(info)
print ("Add new column by passing series")
df['three']=pd.Series([20,40,60],index=['a','b','c'])
print (df)
print ("Add new column using existing DataFrame columns")
df['four']=df['one']+df['three']
print(df)
```

```
Add new column by passing series
   one  two  three
a  1.0    1   20.0
b  2.0    2   40.0
c  3.0    3   60.0
d  4.0    4    NaN
e  5.0    5    NaN
f  NaN    6    NaN
Add new column using existing DataFrame columns
   one  two  three  four
a  1.0    1   20.0  21.0
b  2.0    2   40.0  42.0
c  3.0    3   60.0  63.0
d  4.0    4    NaN   NaN
e  5.0    5    NaN   NaN
f  NaN    6    NaN   NaN
```

In [30]:
```
info = {'one' : pd.Series([1, 2], index= ['a', 'b']),
'two' : pd.Series([1, 2, 3], index=['a', 'b', 'c'])}
df = pd.DataFrame(info)
print ("The DataFrame:")
print (df)
print ("Delete the first column:")
del df['one']
print (df)
print ("Delete the another column:")
df.pop('two')
print (df)
```

```
The DataFrame:
   one  two
a  1.0    1
b  2.0    2
c  NaN    3
Delete the first column:
   two
a    1
b    2
c    3
Delete the another column:
Empty DataFrame
Columns: []
Index: [a, b, c]
```

In [31]:
```
info = {'one' : pd.Series([1, 2, 3, 4, 5], index=['a', 'b', 'c', 'd', 'e']),
'two' : pd.Series([1, 2, 3, 4, 5, 6], index=['a', 'b', 'c', 'd', 'e', 'f'])}
df = pd.DataFrame(info)
print (df.loc['b'])
```

```
one    2.0
two    2.0
Name: b, dtype: float64
```

In [32]:
```
info = {'one' : pd.Series([1, 2, 3, 4, 5], index=['a', 'b', 'c', 'd', 'e']),
'two' : pd.Series([1, 2, 3, 4, 5, 6], index=['a', 'b', 'c', 'd', 'e', 'f'])}
df = pd.DataFrame(info)
print (df.iloc[3])
```

```
one    4.0
two    4.0
Name: d, dtype: float64
```

In [33]:
```
info = {'one' : pd.Series([1, 2, 3, 4, 5], index=['a', 'b', 'c', 'd', 'e']),
'two' : pd.Series([1, 2, 3, 4, 5, 6], index=['a', 'b', 'c', 'd', 'e', 'f'])}
df = pd.DataFrame(info)
print (df[2:5])
```

```
   one  two
c  3.0    3
d  4.0    4
e  5.0    5
```

In [34]:
```
d = pd.DataFrame([[7, 8], [9, 10]], columns = ['x','y'])
d2 = pd.DataFrame([[11, 12], [13, 14]], columns = ['x','y'])
d = d.append(d2)
print (d)
```

```
    x   y
0   7   8
1   9  10
0  11  12
1  13  14
```

In [35]:
```
a_info = pd.DataFrame([[4, 5], [6, 7]], columns = ['x','y'])
b_info = pd.DataFrame([[8, 9], [10, 11]], columns = ['x','y'])
a_info = a_info.append(b_info)
a_info = a_info.drop(0)
print(a_info)
```

```
    x   y
1   6   7
1  10  11
```

In [45]:
```
info1 = pd.DataFrame({"x":[25,15,12,19],
"y":[47, 24, 17, 29]})
Info2 = pd.DataFrame({"x":[25, 15, 12],
"y":[47, 24, 17],
"z":[38, 12, 45]})
info.append(info2, ignore_index = True)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-45-0bebc4b68eee> in <module>()
      4 "y":[47, 24, 17],
      5 "z":[38, 12, 45]})
----> 6 info.append(info2, ignore_index = True)

AttributeError: 'dict' object has no attribute 'append'
```

In [46]:
```
import pandas as pd
info1 = info = pd.DataFrame({"x":[15, 25, 37, 42],
"y":[24, 38, 18, 45]})
info2 = pd.DataFrame({"x":[15, 25, 37],
"y":[24, 38, 45]})
print(info1, "\n")
info2
info1.append(df2)
info.append(info2, ignore_index = True)
```

```
    x   y
0  15  24
1  25  38
2  37  18
3  42  45


---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-46-381fdc7d40c7> in <module>()
      6 print(info1, "\n")
      7 info2
----> 8 info1.append(df2)
      9 info.append(info2, ignore_index = True)

NameError: name 'df2' is not defined
```

In [48]:
```
info = pd.DataFrame([[2, 7]] * 4, columns=['P', 'Q'])
info.apply(np.sqrt)
info.apply(np.sum, axis=0)
info.apply(np.sum, axis=1)
info.apply(lambda x: [1, 2], axis=1)
info.apply(lambda x: [1, 2], axis=1, result_type='expand')
info.apply(lambda x: pd.Series([1, 2], index=['foo', 'bar']), axis=1)
info.apply(lambda x: [1, 2], axis=1, result_type='broadcast')
info
```

Out[48]:

	P	Q
0	2	7
1	2	7
2	2	7
3	2	7

In [49]:
```
info=pd.DataFrame([[1,5,7],[10,12,15],[18,21,24],[np.nan,np.nan,np.nan]],columns=['X','Y','Z'])
info.agg(['sum','min'])
```

Out[49]:

	X	Y	Z
sum	29.0	38.0	46.0
min	1.0	5.0	7.0

In [50]:
```
info=pd.DataFrame([[1,5,7],[10,12,15],[18,21,24],[np.nan,np.nan,np.nan]],columns=['X','Y','Z'])
df.agg({'A' : ['sum', 'min'], 'B' : ['min', 'max']})
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-50-23e0370cf09a> in <module>()
      1 info=pd.DataFrame([[1,5,7],[10,12,15],[18,21,24],[np.nan,np.nan,np.nan]],columns=['X','Y','Z'])
----> 2 df.agg({'A' : ['sum', 'min'], 'B' : ['min', 'max']})

~\Anaconda3\lib\site-packages\pandas\core\frame.py in aggregate(self, func, axis, *args, **kwargs)
   7364         result = None
   7365         try:
-> 7366             result, how = self._aggregate(func, axis, *args, **kwargs)
   7367         except TypeError as err:
   7368             exc = TypeError(

~\Anaconda3\lib\site-packages\pandas\core\frame.py in _aggregate(self, arg, axis, *args, **kwargs)
   7389             result = result.T if result is not None else result
   7390             return result, how
-> 7391         return super()._aggregate(arg, *args, **kwargs)
   7392
   7393     agg = aggregate

~\Anaconda3\lib\site-packages\pandas\core\base.py in _aggregate(self, arg, *args, **kwargs)
    340                         raise SpecificationError("nested renamer is not supported")
    341                     elif isinstance(obj, ABCDataFrame) and k not in obj.columns:
--> 342                         raise KeyError(f"Column '{k}' does not exist!")
    343
    344                 arg = new_arg

KeyError: "Column 'A' does not exist!"
```

In [52]:
```
import pandas as pd
info = pd.DataFrame()
info['ID'] = [101, 102, 103]
info
info.assign(Name = ['Smith', 'Parker', 'John'])
```

Out[52]:

	ID	Name
0	101	Smith
1	102	Parker
2	103	John

In [53]:
```
import pandas as pd
info = pd.DataFrame({'temp_c': [17.0, 25.0]},
index=['Canada', 'Australia'])
info
info.assign(temp_f=lambda x: x.temp_c * 7 / 2 + 24)
info.assign(temp_f=lambda x: x['temp_c'] * 6 / 2 + 21,
temp_k=lambda x: (x['temp_f'] + 342.27) * 6 / 4)
```

Out[53]:

	temp_c	temp_f	temp_k
Canada	17.0	72.0	621.405
Australia	25.0	96.0	657.405

In [55]:
```
a = {'col1': [1, 2], 'col2': [3, 4]}
info = pd.DataFrame(data=a)
info.dtypes
info.astype('int64').dtypes
info.astype({'col1': 'int64'}).dtypes
x = pd.Series([1, 2], dtype='int64')
x.astype('category')
cat_dtype = pd.api.types.CategoricalDtype(
categories=[2, 1], ordered=True)
x.astype(cat_dtype)
x1 = pd.Series([1,2])
x2 = x1.astype('int64', copy=False)
x2[0] = 10
x1
```

Out[55]:
```
0    10
1     2
dtype: int64
```

In [57]:
```
info = pd.DataFrame({"Person":["Parker", "Smith", "William", "John"],
"Age": [27., 29, np.nan, 32]
info.count()
```

```
  File "<ipython-input-57-391ecfe091d6>", line 3
    info.count()
       ^
SyntaxError: invalid syntax
```

In [58]:
```
info = pd.DataFrame({"Person":["Parker", "Smith", "William", "John"],
"Age": [27., 29, np.nan, 32]
info.count(axis='columns')
```

```
  File "<ipython-input-58-525e14218a0e>", line 3
    info.count(axis='columns')
       ^
SyntaxError: invalid syntax
```

In [59]:
```
info_nums = pd.DataFrame({'num': np.random.randint(1, 50, 11)})
print(info_nums)
info_nums['num_bins'] = pd.cut(x=df_nums['num'], bins=[1, 25, 50])
print(info_nums)
print(info_nums['num_bins'].unique())
```

```
    num
0    35
1    20
2    39
3    49
4    20
5    38
6    49
7    35
8    23
9     9
10   38

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-59-76461cb77308> in <module>()
      1 info_nums = pd.DataFrame({'num': np.random.randint(1, 50, 11)})
      2 print(info_nums)
----> 3 info_nums['num_bins'] = pd.cut(x=df_nums['num'], bins=[1, 25, 50])
      4 print(info_nums)
      5 print(info_nums['num_bins'].unique())

NameError: name 'df_nums' is not defined
```

In [60]:
```
info_nums = pd.DataFrame({'num': np.random.randint(1, 10, 7)})
print(info_nums)
info_nums['nums_labels'] = pd.cut(x=info_nums['num'], bins=[1, 7, 10], labels=['Lows', 'Highs'], right=False)
print(info_nums)
print(info_nums['nums_labels'].unique())
```

```
   num
0    4
1    1
2    1
3    4
4    6
5    2
6    9
   num nums_labels
0    4        Lows
1    1        Lows
2    1        Lows
3    4        Lows
4    6        Lows
5    2        Lows
6    9       Highs
['Lows', 'Highs']
Categories (2, object): ['Lows' < 'Highs']
```

In [61]:
```
a1 = pd.Series([1, 2, 3])
a1.describe()
```

Out[61]:
```
count    3.0
mean     2.0
std      1.0
min      1.0
25%      1.5
50%      2.0
75%      2.5
max      3.0
dtype: float64
```

In [62]:
```
a1 = pd.Series(['p', 'q', 'q', 'r'])
a1.describe()
```

Out[62]:
```
count     4
unique    3
top       q
freq      2
dtype: object
```

In [63]:
```
a1 = pd.Series([1, 2, 3])
a1.describe()
a1 = pd.Series(['p', 'q', 'q', 'r'])
a1.describe()
info = pd.DataFrame({'categorical': pd.Categorical(['s','t','u']),
'numeric': [1, 2, 3],
'object': ['p', 'q', 'r']
})
info.describe(include=[np.number])
info.describe(include=[np.object])
info.describe(include=['category'])
```

```
C:\Users\samir\Anaconda3\lib\site-packages\ipykernel_launcher.py:10: DeprecationWarning: `np.object` is a deprecated alias for the builtin `
object`. To silence this warning, use `object` by itself. Doing this will not modify any behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  # Remove the CWD from sys.path while we load stuff.
```

Out[63]:

	categorical
count	3
unique	3
top	u
freq	1

In [64]:
```
a1 = pd.Series([1, 2, 3])
a1.describe()
a1 = pd.Series(['p', 'q', 'q', 'r'])
a1.describe()
info = pd.DataFrame({'categorical': pd.Categorical(['s','t','u']),
'numeric': [1, 2, 3],
'object': ['p', 'q', 'r']
})
info.describe()
info.describe(include='all')
info.numeric.describe()
info.describe(include=[np.number])
info.describe(include=[np.object])
info.describe(include=['category'])
info.describe(exclude=[np.number])
info.describe(exclude=[np.object])
```

```
C:\Users\samir\Anaconda3\lib\site-packages\ipykernel_launcher.py:13: DeprecationWarning: `np.object` is a deprecated alias for the builtin `
object`. To silence this warning, use `object` by itself. Doing this will not modify any behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  del sys.path[0]
C:\Users\samir\Anaconda3\lib\site-packages\ipykernel_launcher.py:16: DeprecationWarning: `np.object` is a deprecated alias for the builtin `
object`. To silence this warning, use `object` by itself. Doing this will not modify any behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  app.launch_new_instance()
```

Out[64]:

	categorical	numeric
count	3	3.0
unique	3	NaN
top	u	NaN
freq	1	NaN
mean	NaN	2.0
std	NaN	1.0
min	NaN	1.0
25%	NaN	1.5
50%	NaN	2.0
75%	NaN	2.5
max	NaN	3.0

In [65]:
```
emp = {"Name": ["Parker", "Smith", "William", "Parker"],
"Age": [21, 32, 29, 21]}
info = pd.DataFrame(emp)
print(info)
```

```
      Name  Age
0   Parker   21
1    Smith   32
2  William   29
3   Parker   21
```

In [66]:
```
emp = {"Name": ["Parker", "Smith", "William", "Parker"],
"Age": [21, 32, 29, 21]}
info = pd.DataFrame(emp)
info = info.drop_duplicates()
print(info)
```

```
      Name  Age
0   Parker   21
1    Smith   32
2  William   29
```

In [67]:
```
data = {'Name': ['Parker', 'Smith', 'John', 'William'],
'Percentage': [82, 98, 91, 87],
'Course': ['B.Sc','B.Ed','M.Phill','BA']}
df = pd.DataFrame(data)
grouped = df.groupby('Course')
print(grouped['Percentage'].agg(np.mean))
```

```
Course
B.Ed       98
B.Sc       82
BA         87
M.Phill    91
Name: Percentage, dtype: int64
```

In [68]:
```
data = {'Name': ['Parker', 'Smith', 'John', 'William'],
'Percentage': [82, 98, 91, 87],
'Course': ['B.Sc','B.Ed','M.Phill','BA']}
df = pd.DataFrame(data)
grouped = df.groupby('Course')
Percentage = lambda x: (x - x.mean()) / x.std()*10
print(grouped.transform(Percentage))
```

```
   Percentage
0         NaN
1         NaN
2         NaN
3         NaN
```

In [69]:
```
data = {'Name': ['Parker', 'Smith', 'John', 'William'],
'Percentage': [82, 98, 91, 87],
'Course': ['B.Sc','B.Ed','M.Phill','BA']}
df = pd.DataFrame(data)
grouped = df.groupby('Course')
print (df.groupby('Course').filter(lambda x: len(x) >= 1))
```

```
      Name  Percentage   Course
0   Parker          82     B.Sc
1    Smith          98     B.Ed
2     John          91  M.Phill
3  William          87       BA
```

In [71]:
```
info = pd.DataFrame({'Name': ['Parker', 'Smith','John', 'William'],'Percentage': [92., 98., 89.,86.]})
info
```

Out[71]:

	Name	Percentage
0	Parker	92.0
1	Smith	98.0
2	John	89.0
3	William	86.0

In [72]:
```
data = {'Name': ['Parker', 'Smith', 'John', 'William'],
'Percentage': [82, 98, 91, 87],}
info = pd.DataFrame(data)
print (info)
```

```
      Name  Percentage
0   Parker          82
1    Smith          98
2     John          91
3  William          87
```

In [73]:
```
info = pd.DataFrame({'language':['C', 'C++', 'Python', 'Java','PHP']})
info.head()
info.head(3)
```

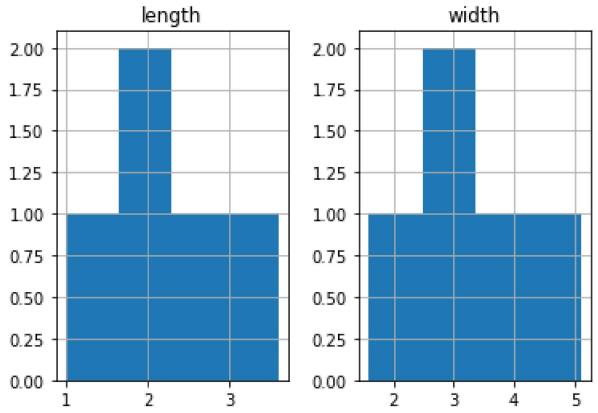
Out[73]:

	language
0	C
1	C++
2	Python

In [74]:
```
import pandas as pd
data = pd.read_csv("aa.csv")
data_top = data.head(2)
data_top
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-74-c26f73e44889> in <module>()
      1 import pandas as pd
----> 2 data = pd.read_csv("aa.csv")
      3 data_top = data.head(2)
      4 data_top

~\Anaconda3\lib\site-packages\pandas\io\parsers.py in read_csv(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols, squeez
e, prefix, mangle_dupe_cols, dtype, engine, converters, true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows, na_values,
keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date_parser, dayfirst, cache_date
s, iterator, chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting, doublequote, escapechar, comment, encoding, dia
lect, error_bad_lines, warn_bad_lines, delim_whitespace, low_memory, memory_map, float_precision)
    686     )
    687
--> 688     return _read(filepath_or_buffer, kwds)
    689
    690

~\Anaconda3\lib\site-packages\pandas\io\parsers.py in _read(filepath_or_buffer, kwds)
    452
    453     # Create the parser.
--> 454     parser = TextFileReader(fp_or_buf, **kwds)
    455
    456     if chunksize or iterator:

~\Anaconda3\lib\site-packages\pandas\io\parsers.py in __init__(self, f, engine, **kwds)
    946             self.options["has_index_names"] = kwds["has_index_names"]
    947
--> 948         self._make_engine(self.engine)
    949
    950     def close(self):

~\Anaconda3\lib\site-packages\pandas\io\parsers.py in _make_engine(self, engine)
   1178     def _make_engine(self, engine="c"):
   1179         if engine == "c":
-> 1180             self._engine = CParserWrapper(self.f, **self.options)
   1181         else:
   1182             if engine == "python":

~\Anaconda3\lib\site-packages\pandas\io\parsers.py in __init__(self, src, **kwds)
   2008         kwds["usecols"] = self.usecols
   2009
-> 2010         self._reader = parsers.TextReader(src, **kwds)
   2011         self.unnamed_cols = self._reader.unnamed_cols
   2012

pandas\_libs\parsers.pyx in pandas._libs.parsers.TextReader.__cinit__()

pandas\_libs\parsers.pyx in pandas._libs.parsers.TextReader._setup_parser_source()

FileNotFoundError: [Errno 2] No such file or directory: 'aa.csv'
```

In [75]:
```
info = pd.DataFrame({
'length': [2, 1.7, 3.6, 2.4, 1],
'width': [4.2, 2.6, 1.6, 5.1, 2.9]})
hist = info.hist(bins=4)
```



In [79]:
```
import pandas as pd
info = pd.DataFrame({'A': {0: 'p', 1: 'q', 2: 'r'},
'B': {0: 40, 1: 55, 2: 25},
'C': {0: 56, 1: 62, 2: 42}})
pd.melt(info, id_vars=['A'], value_vars=['C'])
pd.melt(info, id_vars=['A'], value_vars=['B', 'C'])
pd.melt(info, id_vars=['A'], value_vars=['C'],
var_name='myVarname', value_name='myValname')
```

Out[79]:

	A	myVarname	myValname
0	p	C	56
1	q	C	62
2	r	C	42

In [80]:
```
import pandas as pd
left = pd.DataFrame({
'id':[1,2,3,4],
'Name': ['John', 'Parker', 'Smith', 'Parker'],
'subject_id':['sub1','sub2','sub4','sub6']})
right = pd.DataFrame({
'id':[1,2,3,4],
'Name': ['William', 'Albert', 'Tony', 'Allen'],
'subject_id':['sub2','sub4','sub3','sub6']})
print (left)
print (right)
```

```
   id    Name subject_id
0   1    John       sub1
1   2  Parker       sub2
2   3   Smith       sub4
3   4  Parker       sub6
   id     Name subject_id
0   1  William       sub2
1   2   Albert       sub4
2   3     Tony       sub3
3   4    Allen       sub6
```

In [82]:
```
info = pd.DataFrame({'P': ['Smith', 'John', 'William', 'Parker'],
'Q': ['Python', 'C', 'C++', 'Java'],
'R': [19, 24, 22, 25]})
info
table = pd.pivot_table(info, index =['P', 'Q'])
table
```

Out[82]:

		R
P	**Q**	
John	**C**	24
Parker	**Java**	25
Smith	**Python**	19
William	**C++**	22

In [83]:
```
info = pd.DataFrame({'X': range(1, 6),
'Y': range(10, 0, -2),
'Z Z': range(10, 5, -1)})
info
info.query('X > Y')
info[info.X > info.Y]
info[info.Y == info['Z Z']]
```

Out[83]:

	X	Y	Z Z
0	1	10	10

In [86]:
```
info = {'name': ['Parker', 'Smith', 'William', 'Robert'],
'age': [38, 47, 44, 34],
'language': ['Java', 'Python', 'JavaScript', 'Python']}
info_pd = pd.DataFrame(info)
print(info_pd)
info_pd.rename(columns = {'name':'Name'}, inplace = True)
print("\nAfter modifying first column:\n", info_pd.columns)
```

```
      name  age    language
0   Parker   38        Java
1    Smith   47      Python
2  William   44  JavaScript
3   Robert   34      Python

After modifying first column:
 Index(['Name', 'age', 'language'], dtype='object')
```

In [89]:
```
info = pd.DataFrame({'data1': [2, 4, 8, 0],
'data2': [2, 0, 0, 0],
'data3': [10, 2, 1, 8]},
index=['John', 'Parker', 'Smith', 'William'])
info
info['data1'].sample(n=3, random_state=1)
info.sample(frac=0.5, replace=True, random_state=1)
info.sample(n=2, weights='data3', random_state=1)
```

Out[89]:

	data1	data2	data3
John	2	2	10
William	0	0	8

In [90]:
```
info= pd.DataFrame({'a_data': [45, 28, 39, 32, 18],
'b_data': [26, 37, 41, 35, 45],
'c_data': [22, 19, 11, 25, 16]})
info.shift(periods=2)
```

Out[90]:

	a_data	b_data	c_data
0	NaN	NaN	NaN
1	NaN	NaN	NaN
2	45.0	26.0	22.0
3	28.0	37.0	19.0
4	39.0	41.0	11.0

In [91]:
```
info= pd.DataFrame({'a_data': [45, 28, 39, 32, 18],
'b_data': [26, 38, 41, 35, 45],
'c_data': [22, 19, 11, 25, 16]})
info.shift(periods=2)
info.shift(periods=2,axis=1,fill_value= 70)
```

Out[91]:

	a_data	b_data	c_data
0	70	70	45
1	70	70	28
2	70	70	39
3	70	70	32
4	70	70	18

In [92]:
```
info=pd.DataFrame(np.random.randn(10,2),index=[1,3,7,2,4,5,9,8,0,6],columns=['col2','col1'])
print(info)
```

```
       col2      col1
1  0.941541 -0.848997
3 -2.019275 -0.199827
7  0.968417  0.693776
2  0.163754 -0.183727
4  0.168173 -0.798187
5 -0.373903  0.400373
9  0.452809 -0.735786
8 -0.614082 -0.442762
0  1.389327 -0.572569
6  1.732545 -0.016950
```

In [93]:
```
info=pd.DataFrame(np.random.randn(10,2),index=[1,2,5,4,8,7,9,3,0,6],columns = ['col4','col3'])
info2=info.sort_index()
print(info2)
```

```
       col4      col3
0 -0.876527  0.894678
1  0.399854  1.421984
2  0.606392 -1.016643
3  0.016354 -0.983710
4 -0.547883 -1.205958
5 -0.474872  0.504485
6 -1.241455  1.374123
7 -1.982797  0.916754
8 -1.074007 -0.552876
9  0.227106  0.195191
```

In [94]:
```
info = pd.DataFrame({'col1':[7,1,8,3],'col2':[8,12,4,9]})
info_2 = info.sort_values(by='col2')
print(info_2)
```

```
   col1  col2
2     8     4
0     7     8
3     3     9
1     1    12
```

In [95]:
```
info = {'Name': ['Parker', 'Smith', 'William'],
'age' : [32, 28, 39]}
data = pd.DataFrame(info)
data['total'] = data['age'].sum()
print(data)
```

```
      Name  age  total
0   Parker   32     99
1    Smith   28     99
2  William   39     99
```

In [96]:
```
info_marks = pd.DataFrame({'name': ['Parker', 'Smith', 'William', 'Terry'],
'Maths': [78, 84, 67, 72],
'Science': [89, 92, 61, 77],
'English': [72, 75, 64, 82]})
writer = pd.ExcelWriter('output.xlsx')
info_marks.to_excel(writer)
writer.save()
print('DataFrame is written successfully to the Excel File.')
```

```
DataFrame is written successfully to the Excel File.
```

In [97]:
```
info =pd.DataFrame({"P":[8, 2, 9, None, 3],
"Q":[4, 14, 12, 22, None],
"R":[2, 5, 7, 16, 13],
"S":[16, 10, None, 19, 18]})
index_ =['A_Row', 'B_Row', 'C_Row', 'D_Row', 'E_Row']
info.index =index_
print(info)
```

```
         P     Q   R     S
A_Row  8.0   4.0   2  16.0
B_Row  2.0  14.0   5  10.0
C_Row  9.0  12.0   7   NaN
D_Row  NaN  22.0  16  19.0
E_Row  3.0   NaN  13  18.0
```

In [98]:
```
info = pd.DataFrame({"A":[8, 2, 7, None, 6],
"B":[4, 3, None, 9, 2],
"C":[17, 42, 35, 18, 24],
"D":[15, 18, None, 11, 12]})
index_ = ['Row1', 'Row2', 'Row3', 'Row4', 'Row5']
info.index = index_
print(info)
result = info.transpose()
print(result)
```

```
        A    B   C     D
Row1  8.0  4.0  17  15.0
Row2  2.0  3.0  42  18.0
Row3  7.0  NaN  35   NaN
Row4  NaN  9.0  18  11.0
Row5  6.0  2.0  24  12.0
   Row1  Row2  Row3  Row4  Row5
A   8.0   2.0   7.0   NaN   6.0
B   4.0   3.0   NaN   9.0   2.0
C  17.0  42.0  35.0  18.0  24.0
D  15.0  18.0   NaN  11.0  12.0
```

In [99]:
```
a = pd.Series(range(5))
a.where(a > 0)
a.mask(a > 0)
a.where(a > 1, 10)
info = pd.DataFrame(np.arange(10).reshape(-1, 2), columns=['A', 'B'])
info
b = info % 3 == 0
info.where(b, -info)
info.where(b, -info) == np.where(b, info, -info)
info.where(b, -info) == info.mask(~b, -info)
```

Out[99]:

	A	B
0	True	True
1	True	True
2	True	True
3	True	True
4	True	True

In [100]:
```
pd.DataFrame({"P": [2, 3], "Q": [4, 5]}).to_numpy()
info = pd.DataFrame({"P": [2, 3], "Q": [4.0, 5.8]})
info.to_numpy()
info['R'] = pd.date_range('2000', periods=2)
info.to_numpy()
```

Out[100]:
```
array([[2, 4.0, Timestamp('2000-01-01 00:00:00')],
       [3, 5.8, Timestamp('2000-01-02 00:00:00')]], dtype=object)
```

In [101]:
```
info = pd.DataFrame([[17, 62, 35],[25, 36, 54],[42, 20, 15],[48, 62, 76]],
columns=['x', 'y', 'z'])
print('DataFrame\n----------\n', info)
arr = info.to_numpy()
print('\nNumpy Array\n----------\n', arr)
```

```
DataFrame
----------
     x   y   z
0  17  62  35
1  25  36  54
2  42  20  15
3  48  62  76

Numpy Array
----------
 [[17 62 35]
 [25 36 54]
 [42 20 15]
 [48 62 76]]
```

In [102]:
```
data = {'Name': ['Smith', 'Parker'], 'ID': [101, 102], 'Language': ['Python', 'JavaScript']}
info = pd.DataFrame(data)
print('DataFrame Values:\n', info)
csv_data = info.to_csv()
print('\nCSV String Values:\n', csv_data)
```

```
DataFrame Values:
      Name   ID    Language
0   Smith  101      Python
1  Parker  102  JavaScript

CSV String Values:
 ,Name,ID,Language
0,Smith,101,Python
1,Parker,102,JavaScript
```

In [103]:
```
data = {'Name': ['Smith', 'Parker'], 'ID': [101, pd.NaT], 'Language': [pd.NaT, 'JavaScript']}
info = pd.DataFrame(data)
print('DataFrame Values:\n', info)
csv_data = info.to_csv()
print('\nCSV String Values:\n', csv_data)
csv_data = info.to_csv(na_rep="None")
print('CSV String with Null Data Values:\n', csv_data)
```

```
DataFrame Values:
      Name   ID    Language
0   Smith  101         NaT
1  Parker  NaT  JavaScript

CSV String Values:
 ,Name,ID,Language
0,Smith,101,
1,Parker,,JavaScript

CSV String with Null Data Values:
 ,Name,ID,Language
0,Smith,101,None
1,Parker,None,JavaScript
```

In [105]:
```
data = {'Name': ['Smith', 'Parker'], 'ID': [101, pd.NaT], 'Language': ['Python', 'JavaScript']}
info = pd.DataFrame(data)
print('DataFrame:\n', info)
csv_data = info.to_csv(sep='|')
print(csv_data)
```

```
DataFrame:
      Name   ID    Language
0   Smith  101      Python
1  Parker  NaT  JavaScript
|Name|ID|Language
0|Smith|101|Python
1|Parker||JavaScript
```

In [107]:
```
one = pd.DataFrame({
'Name': ['Parker', 'Smith', 'Allen', 'John', 'Parker'],
'subject_id':['sub1','sub2','sub4','sub6','sub5'],
'Marks_scored':[98,90,87,69,78]},
index=[1,2,3,4,5])
two = pd.DataFrame({
'Name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],
'subject_id':['sub2','sub4','sub3','sub6','sub5'],
'Marks_scored':[89,80,79,97,88]},
index=[1,2,3,4,5])
print (one.append(two))
```

```
     Name subject_id  Marks_scored
1  Parker       sub1            98
2   Smith       sub2            90
3   Allen       sub4            87
4    John       sub6            69
5  Parker       sub5            78
1   Billy       sub2            89
2   Brian       sub4            80
3    Bran       sub3            79
4   Bryce       sub6            97
5   Betty       sub5            88
```

In [108]:
```
info = pd.Series([11, 14, 17, 24, 19, 32, 34, 27],
index = [['x', 'x', 'x', 'x', 'y', 'y', 'y', 'y'],
['obj1', 'obj2', 'obj3', 'obj4', 'obj1', 'obj2', 'obj3', 'obj4']])
data
```

Out[108]:
```
{'Name': ['Smith', 'Parker'],
 'ID': [101, NaT],
 'Language': ['Python', 'JavaScript']}
```

In [113]:
```
info = pd.DataFrame({'Language known': ['Python', 'Android', 'C', 'Android', 'Python', 'C++', 'C']},
index=['Parker', 'Smith', 'John', 'William', 'Dean', 'Christina', 'Cornelia'])
print(info)
dictionary = {"Python": 1, "Android": 2, "C": 3, "Android": 4, "C++": 5}
info1 = info.replace({"Language known": dictionary})
print("\n\n")
print(info1)
```

```
          Language known
Parker            Python
Smith            Android
John                   C
William          Android
Dean              Python
Christina            C++
Cornelia               C



           Language known
Parker                  1
Smith                   4
John                    3
William                 4
Dean                    1
Christina               5
Cornelia                3
```

In [114]:
```
info = pd.DataFrame({
'name':['Parker','Smith','John'],
'age':[27,34,31],
'city':['US','Belgium','London']})
info.replace([29],38)
```

Out[114]:

	name	age	city
0	Parker	27	US
1	Smith	34	Belgium
2	John	31	London

In [115]:
```
info = pd.DataFrame({
'name':['Parker','Smith','John'],
'age':[27,34,31],
'city':['US','Belgium','London']})
info.replace({34:29,'Smith':'William'})
```

Out[115]:

	name	age	city
0	Parker	27	US
1	William	29	Belgium
2	John	31	London

In [117]:
```
info = pd.DataFrame({'x': [1, 2], 'y': [3, 7]})
p = info.isin(range(1,8))
print('DataFrame\n----------\n',info)
print('\nDataFrame.isin(range(1,6))\n----------\n',p)
```

```
DataFrame
----------
    x  y
0  1  3
1  2  7

DataFrame.isin(range(1,6))
----------
       x     y
0  True  True
1  True  True
```

In [122]:
```
data = pd.DataFrame({
'EmpCode': ['Emp001', 'Emp002', 'Emp003', 'Emp004', 'Emp005'],
'Name': ['Parker', 'Smith', 'Jones', 'Terry', 'Palin'],
'Occupation': ['Tester', 'Developer', 'Statistician',
'Tester', 'Developer'],
'Date Of Join': ['2019-01-17', '2019-01-26', '2019-01-29', '2019-02-02',
'2019-02-11'],
'Age': [29, 22, 25, 38, 27]})
print("\nUseisin operator\n")
print(data.loc[data['Occupation'].isin(['Tester','Developer'])])
(data['Name'] == 'John') &(data['Age'] < 27)
```

```
Useisin operator

  EmpCode    Name Occupation Date Of Join  Age
0  Emp001  Parker     Tester   2019-01-17   29
1  Emp002   Smith  Developer   2019-01-26   22
3  Emp004   Terry     Tester   2019-02-02   38
4  Emp005   Palin  Developer   2019-02-11   27
```

Out[122]:
```
0    False
1    False
2    False
3    False
4    False
dtype: bool
```

In [123]:
```
info = pd.DataFrame({'Age':[32, 41, 44, 38, 33],
'Name':['Phill', 'William', 'Terry', 'Smith', 'Parker']})
# Create the index
index_ = ['Row_1', 'Row_2', 'Row_3', 'Row_4', 'Row_5']
info.index = index_
final = info.loc['Row_2', 'Name']
print(final)
```

```
William
```

In [124]:
```
info = pd.DataFrame({"P":[28, 17, 14, 42, None],
"Q":[15, 23, None, 15, 12],
"R":[11, 23, 16, 32, 42],
"S":[41, None, 34, 25, 18]})
index_ = ['A', 'B', 'C', 'D', 'E']
info.index = index_
print(info)
```

```
      P     Q   R     S
A  28.0  15.0  11  41.0
B  17.0  23.0  23   NaN
C  14.0   NaN  16  34.0
D  42.0  15.0  32  25.0
E   NaN  12.0  42  18.0
```

In [125]:
```
result = info.loc[:, ['P', 'S']]
print(result)
```

```
      P     S
A  28.0  41.0
B  17.0   NaN
C  14.0  34.0
D  42.0  25.0
E   NaN  18.0
```

In [ ]:
```

```