pandas-2

November 11, 2025

[3]:
```
import pandas as pd
import numpy as np
```

[4]:
```
# DataFrame using array of tuple

arr = [(1, "One"), (2, "Two"), (3, "Three")]
arr
```

[4]:
```
[(1, 'One'), (2, 'Two'), (3, 'Three')]
```

[5]:
```
df = pd.DataFrame(arr, columns=("Number", "Words"))
df
```

[5]:

	Number	Words
0	1	One
1	2	Two
2	3	Three

[6]:
```
data = {1: "One", 2: "Two", 3: "Three"}
data = {"Number": [1, 2, 3], "Words": ["One", "Two", "Three"]}
data
```

[6]:
```
{'Number': [1, 2, 3], 'Words': ['One', 'Two', 'Three']}
```

[7]:
```
df1 = pd.DataFrame(data)
df1
```

[7]:

	Number	Words
0	1	One
1	2	Two
2	3	Three

[8]:
```
# Adding new data

print(pd.concat([df, df1]))
```

```
   Number  Words
0       1    One
1       2    Two
```

```
2       3  Three
0       1    One
1       2    Two
2       3  Three
```

[9]:
```
df = pd.DataFrame([[1, 2], [3, 4]], columns=list('AB'), index=['x', 'y'])
df
```

[9]:
```
   A  B
x  1  2
y  3  4
```

[10]:
```
df2 = pd.DataFrame([[5, 6], [7, 8]], columns=list('AB'), index=['x', 'y'])
df2
```

[10]:
```
   A  B
x  5  6
y  7  8
```

[11]:
```
new_df = pd.concat([df, df2])
```

[12]:
```
new_df
```

[12]:
```
   A  B
x  1  2
y  3  4
x  5  6
y  7  8
```

[13]:
```
df
```

[13]:
```
   A  B
x  1  2
y  3  4
```

[14]:
```
# Save to csv

df.to_csv("sample.csv", index=False)
```

[15]:
```
# Open csv file

new_df = pd.read_csv("sample.csv")
new_df
```

[15]:
```
   A  B
0  1  2
1  3  4
```

[19]:
```
# Slicing in df

df = pd.DataFrame(
    np.arange(18).reshape(6, 3),
    columns=["One", "Two", "Three"],
    index=["a", "b", "c", "d", "e", "f"]
)
df
```

[19]:

	One	Two	Three
a	0	1	2
b	3	4	5
c	6	7	8
d	9	10	11
e	12	13	14
f	15	16	17

[20]:
```
# Selecting single column

df.Two
```

[20]:
```
a     1
b     4
c     7
d    10
e    13
f    16
Name: Two, dtype: int64
```

[23]:
```
df_col2 = df["Two"]
df_col2
```

[23]:
```
a     1
b     4
c     7
d    10
e    13
f    16
Name: Two, dtype: int64
```

[24]:
```
df_u = df[["One", "Three"]]
df_u
```

[24]:

	One	Three
a	0	2
b	3	5
c	6	8
d	9	11

```
e   12     14
f   15     17
```

[25]:
```
df - df_u
```

[25]:
```
   One  Three  Two
a    0      0  NaN
b    0      0  NaN
c    0      0  NaN
d    0      0  NaN
e    0      0  NaN
f    0      0  NaN
```

[26]:
```
df
```

[26]:
```
   One  Two  Three
a    0    1      2
b    3    4      5
c    6    7      8
d    9   10     11
e   12   13     14
f   15   16     17
```

[28]:
```
df.iloc[1:3, 1:3]
```

[28]:
```
   Two  Three
b    4      5
c    7      8
```

[ ]:
```
df.loc["a":"d", "Two":"Three"]
```

[ ]:
```
   Two  Three
a    1      2
b    4      5
c    7      8
d   10     11
e   13     14
f   16     17
```

[34]:
```
df
```

[34]:
```
   One  Two  Three
a    0    1      2
b    3    4      5
c    6    7      8
d    9   10     11
e   12   13     14
f   15   16     17
```

[35]:
```
df.loc["a", "Two"] = 4
```

[36]:
```
df
```

[36]:

	One	Two	Three
a	0	4	2
b	3	4	5
c	6	7	8
d	9	10	11
e	12	13	14
f	15	16	17

[38]:
```
# Calculation of mean column wise

df.mean()
```

[38]:
```
One      7.5
Two      9.0
Three    9.5
dtype: float64
```

[39]:
```
# Calculation of sd column wise

df.std()
```

[39]:
```
One      5.612486
Two      4.898979
Three    5.612486
dtype: float64
```

[43]:
```
# Standarization

std_df = (df - df.mean()) / df.std()
std_df
```

[43]:

	One	Two	Three
a	-1.336306	-1.020621	-1.336306
b	-0.801784	-1.020621	-0.801784
c	-0.267261	-0.408248	-0.267261
d	0.267261	0.204124	0.267261
e	0.801784	0.816497	0.801784
f	1.336306	1.428869	1.336306

[ ]:
```
# Square the data

std_df.pow(2)
```

[ ]:

	One	Two	Three
a	1.785714	1.041667	1.785714
b	0.642857	1.041667	0.642857
c	0.071429	0.166667	0.071429
d	0.071429	0.041667	0.071429
e	0.642857	0.666667	0.642857
f	1.785714	2.041667	1.785714

[45]:
```
# Square root of df

np.sqrt(df)
```

[45]:

	One	Two	Three
a	0.000000	2.000000	1.414214
b	1.732051	2.000000	2.236068
c	2.449490	2.645751	2.828427
d	3.000000	3.162278	3.316625
e	3.464102	3.605551	3.741657
f	3.872983	4.000000	4.123106

0.1 Handling missing data

[47]:
```
import pandas as pd
import numpy as np
import random
```

[63]:
```
values = np.random.randn(21)
values
```

[63]:
```
array([-0.38690101,  1.68458508, -1.1922108 , -0.34103183,  0.43642331,
       -0.05898125,  0.78751423, -0.80324339,  0.02215435, -1.59337394,
        0.53308764,  0.1163521 , -1.96602187,  2.09995047,  0.60930388,
       -0.15209467, -0.61229781, -1.02585561,  0.55976639,  1.06720647,
        1.08440698])
```

[64]:
```
values[random.sample([i for i in range(21)], 5)] = np.nan
values
```

[64]:
```
array([        nan,  1.68458508, -1.1922108 , -0.34103183,  0.43642331,
       -0.05898125,         nan,         nan,  0.02215435, -1.59337394,
        0.53308764,         nan, -1.96602187,  2.09995047,  0.60930388,
               nan, -0.61229781, -1.02585561,  0.55976639,  1.06720647,
        1.08440698])
```

[73]:
```
df = pd.DataFrame(
    values.reshape(7, 3),
    columns=["A", "B", "C"]
)
```

```
df
```

[73]:

	A	B	C
0	NaN	1.684585	-1.192211
1	-0.341032	0.436423	-0.058981
2	NaN	NaN	0.022154
3	-1.593374	0.533088	NaN
4	-1.966022	2.099950	0.609304
5	NaN	-0.612298	-1.025856
6	0.559766	1.067206	1.084407

[66]:
```
# Showing missing data

np.isnan(df)
```

[66]:

	A	B	C
0	True	False	False
1	False	False	False
2	True	True	False
3	False	False	True
4	False	False	False
5	True	False	False
6	False	False	False

[67]:
```
pd.isna(df)
```

[67]:

	A	B	C
0	True	False	False
1	False	False	False
2	True	True	False
3	False	False	True
4	False	False	False
5	True	False	False
6	False	False	False

[69]:
```
pd.isnull(df)
```

[69]:

	A	B	C
0	True	False	False
1	False	False	False
2	True	True	False
3	False	False	True
4	False	False	False
5	True	False	False
6	False	False	False

[70]:
```
df.isnull()
```

[70]:
```
       A      B      C
0   True  False  False
1  False  False  False
2   True   True  False
3  False  False   True
4  False  False  False
5   True  False  False
6  False  False  False
```

[71]:
```
df.notnull()
```

[71]:
```
       A      B      C
0  False   True   True
1   True   True   True
2  False  False   True
3   True   True  False
4   True   True   True
5  False   True   True
6   True   True   True
```

[72]:
```
# Remove null values

df.dropna()
```

[72]:
```
          A         B         C
1 -0.341032  0.436423 -0.058981
4 -1.966022  2.099950  0.609304
6  0.559766  1.067206  1.084407
```

[74]:
```
df
```

[74]:
```
          A         B         C
0       NaN  1.684585 -1.192211
1 -0.341032  0.436423 -0.058981
2       NaN       NaN  0.022154
3 -1.593374  0.533088       NaN
4 -1.966022  2.099950  0.609304
5       NaN -0.612298 -1.025856
6  0.559766  1.067206  1.084407
```

[79]:
```
# Filling the values

df = df.fillna(df.mean())
df
```

[79]:
```
          A         B         C
0 -0.835165  1.684585 -1.192211
1 -0.341032  0.436423 -0.058981
```

```
2 -0.835165  0.868159  0.022154
3 -1.593374  0.533088 -0.093530
4 -1.966022  2.099950  0.609304
5 -0.835165 -0.612298 -1.025856
6  0.559766  1.067206  1.084407
```

[80]:
```
# Sort using index

df.sort_index()
```

[80]:
```
          A         B         C
0 -0.835165  1.684585 -1.192211
1 -0.341032  0.436423 -0.058981
2 -0.835165  0.868159  0.022154
3 -1.593374  0.533088 -0.093530
4 -1.966022  2.099950  0.609304
5 -0.835165 -0.612298 -1.025856
6  0.559766  1.067206  1.084407
```

[85]:
```
# Sorting using index and columns

df.sort_index(axis=1, ascending=False)
```

[85]:
```
          C         B         A
0 -1.192211  1.684585 -0.835165
1 -0.058981  0.436423 -0.341032
2  0.022154  0.868159 -0.835165
3 -0.093530  0.533088 -1.593374
4  0.609304  2.099950 -1.966022
5 -1.025856 -0.612298 -0.835165
6  1.084407  1.067206  0.559766
```

[86]:
```
# Sorting by values

df.sort_values(by="A")
```

[86]:
```
          A         B         C
4 -1.966022  2.099950  0.609304
3 -1.593374  0.533088 -0.093530
2 -0.835165  0.868159  0.022154
0 -0.835165  1.684585 -1.192211
5 -0.835165 -0.612298 -1.025856
1 -0.341032  0.436423 -0.058981
6  0.559766  1.067206  1.084407
```

[87]:
```
df.rank()
```

[87]:
```
     A    B    C
0  4.0  6.0  1.0
1  6.0  2.0  4.0
2  4.0  4.0  5.0
3  2.0  3.0  3.0
4  1.0  7.0  6.0
5  4.0  1.0  2.0
6  7.0  5.0  7.0
```

[88]:
```
df.head()
```

[88]:
```
          A         B         C
0 -0.835165  1.684585 -1.192211
1 -0.341032  0.436423 -0.058981
2 -0.835165  0.868159  0.022154
3 -1.593374  0.533088 -0.093530
4 -1.966022  2.099950  0.609304
```

[89]:
```
df.tail()
```

[89]:
```
          A         B         C
2 -0.835165  0.868159  0.022154
3 -1.593374  0.533088 -0.093530
4 -1.966022  2.099950  0.609304
5 -0.835165 -0.612298 -1.025856
6  0.559766  1.067206  1.084407
```

[90]:
```
df.shape
```

[90]:
```
(7, 3)
```

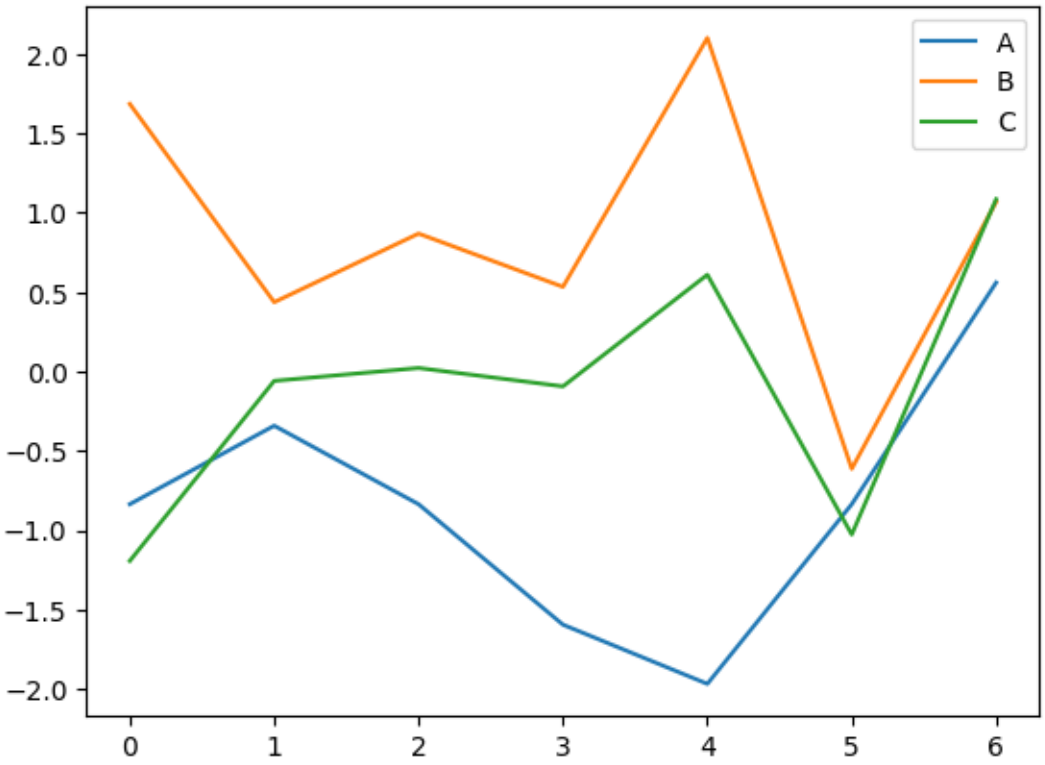
[92]:
```
df.describe()
```

[92]:
```
              A         B         C
count  7.000000  7.000000  7.000000
mean  -0.835165  0.868159 -0.093530
std    0.820947  0.886369  0.813677
min   -1.966022 -0.612298 -1.192211
25%   -1.214270  0.484755 -0.559693
50%   -0.835165  0.868159 -0.058981
75%   -0.588099  1.375896  0.315729
max    0.559766  2.099950  1.084407
```

[99]:
```
%matplotlib inline
```

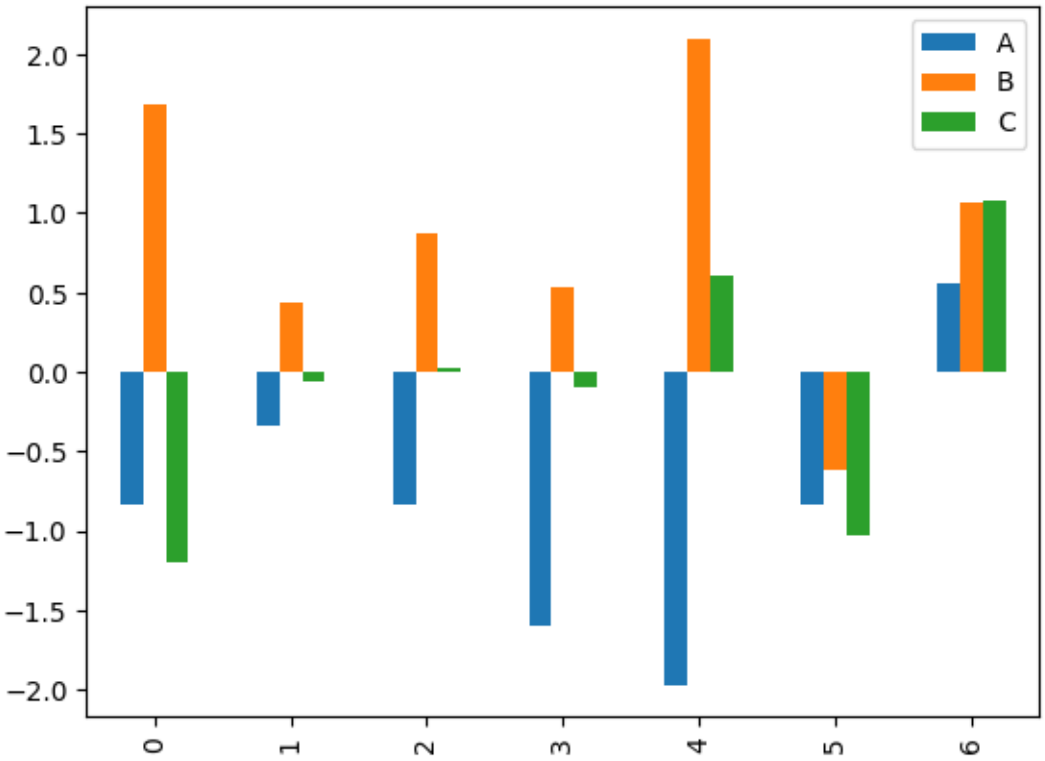
[98]:
```
df.plot(kind="line")
```

[98]:
```
<Axes: >
```



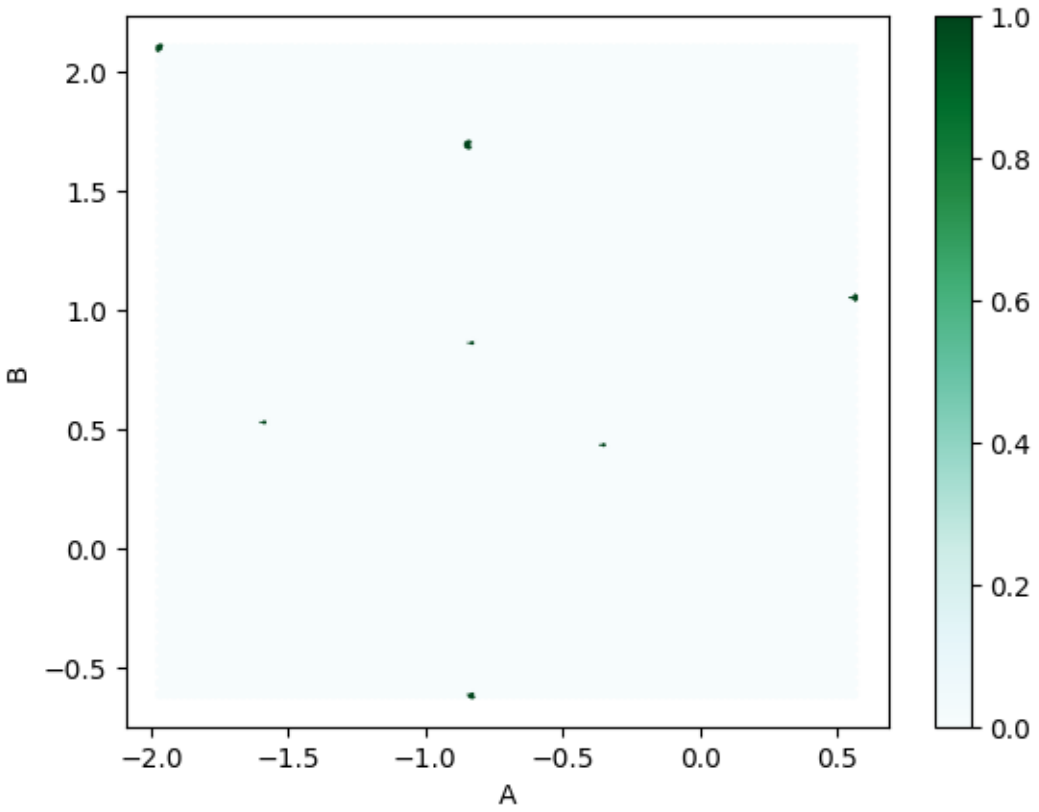
[100]:
```
df.plot(kind="bar")
```

[100]: <Axes: >



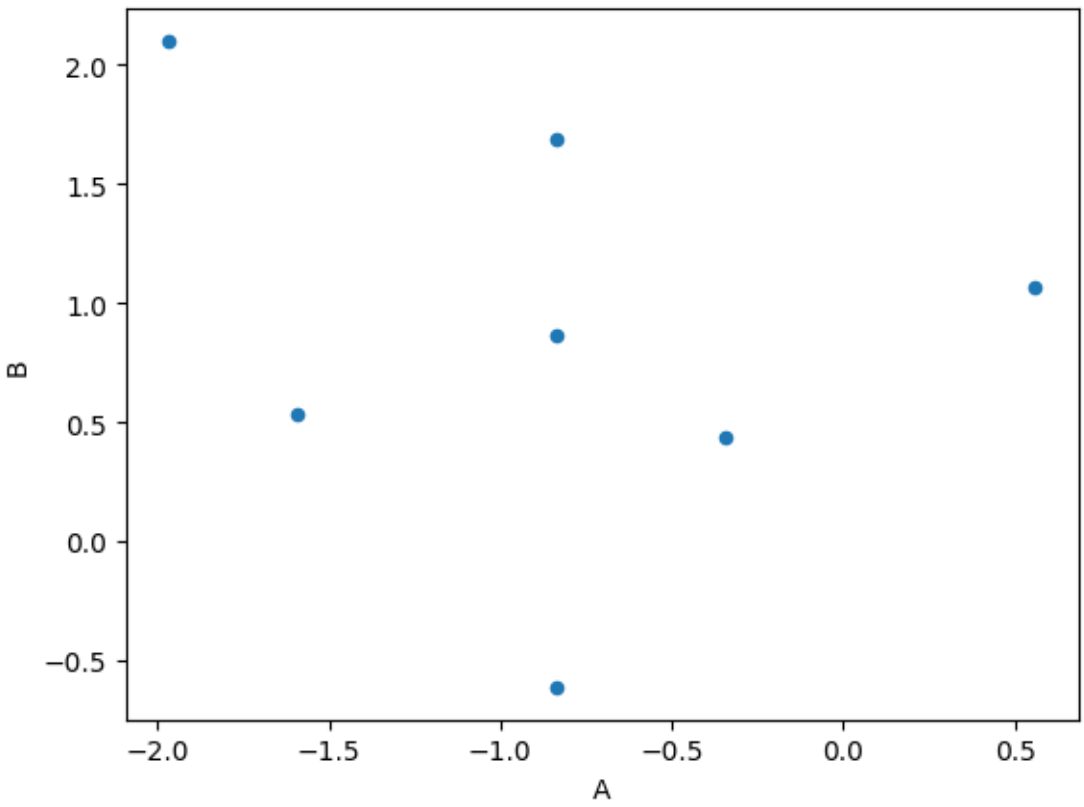
[104]:
```
df.plot(kind="hexbin", x="A", y="B")
```

[104]:
```
<Axes: xlabel='A', ylabel='B'>
```



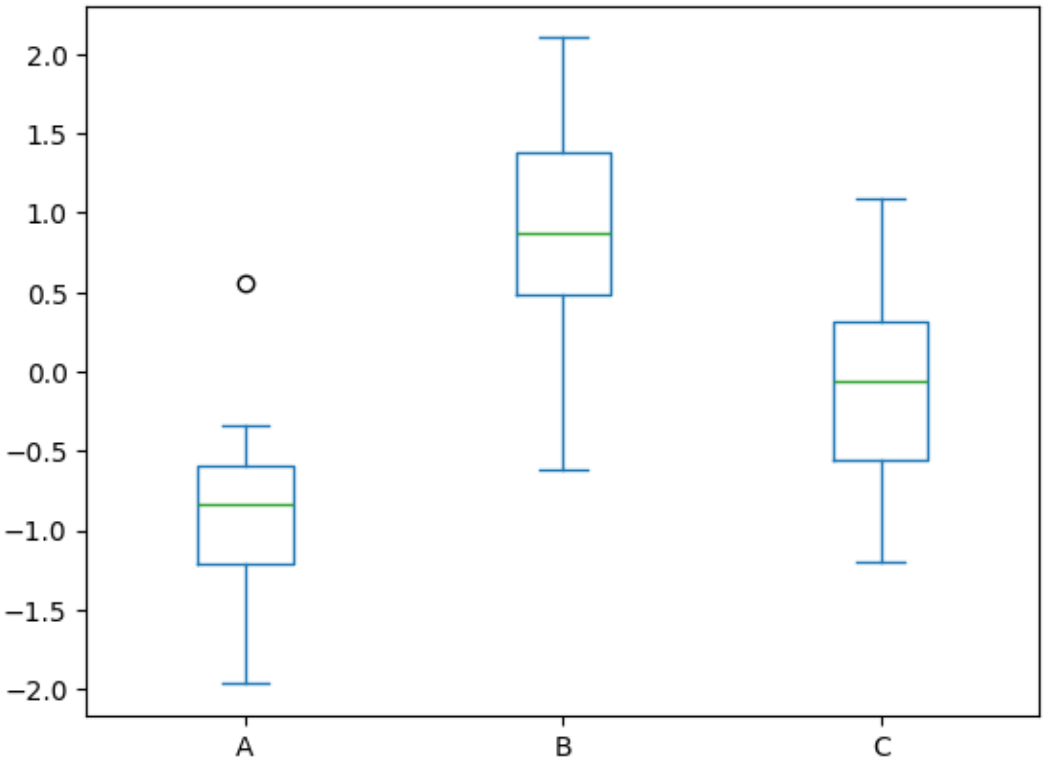
[105]:
```
df.plot(kind="scatter", x="A", y="B")
```

[105]: <Axes: xlabel='A', ylabel='B'>



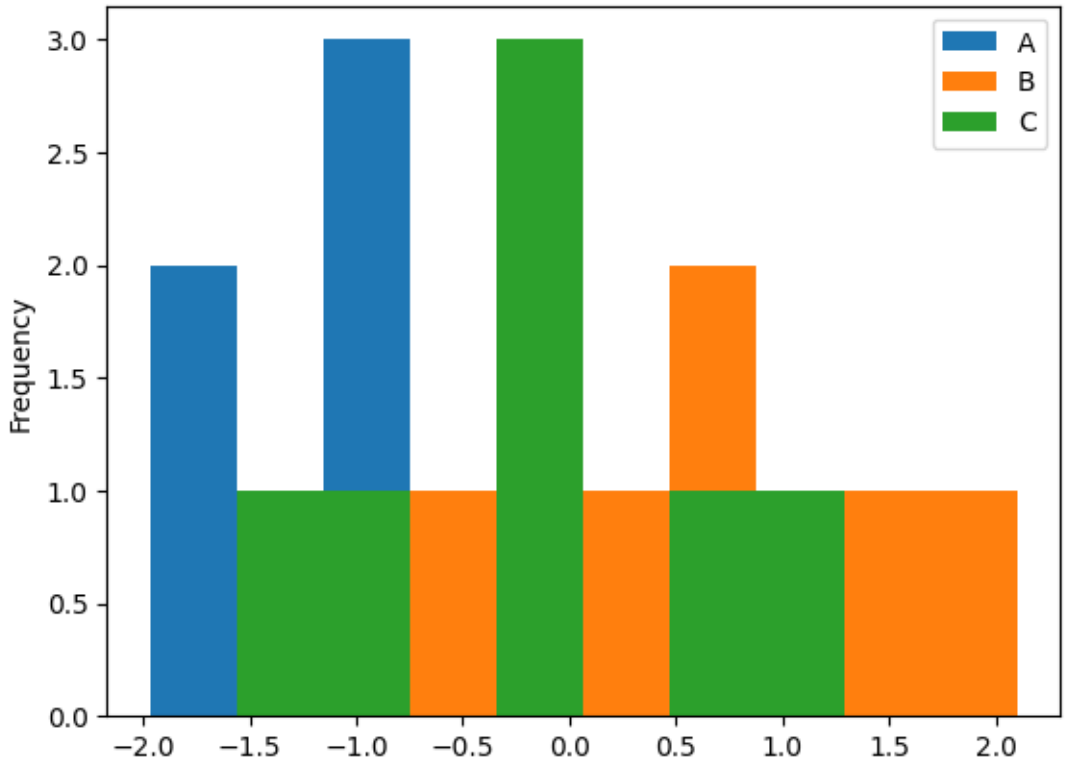
[106]:
```
df.plot(kind="box")
```

[106]: <Axes: >



[107]:
```
df.plot(kind="hist")
```

[107]: <Axes: ylabel='Frequency'>


A
B
C
Frequency
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

[ ]: