[4-1. Pandas 내장 시각화 도구]

1. Pandas 내장 시각화 도구 개요

Pandas 라이브러리는 자체적으로 시각화 기능을 제공합니다. 내장 시각화 기능은 Matplotlib을 기반으로 하며, DataFrame과 Series 데이터 구조에 대한 간편하고 직관적인 시각화 방법을 제공합니다.

이를 사용하기 위해서는 Matplotlib 라이브러리가 설치되어 있어야 합니다. 아나콘다 네비게이터에서 "Environments" 탭에서 현재 사용 중인 가상환경을 선택한 후 라이브러리 이름을 조회하여 설치할 수 있습니다.

터미널 환경에서는 "pip install matplotlib" 명령어를 사용해서 설치할 수 있습니다.

2. 선 그래프

Pandas에서 선 그래프는 시계열 데이터나 연속 데이터를 시각화하는 데 주로 사용됩니다. DataFrame.plot() 또는 Series.plot() 메서드를 사용하여 생성할 수 있습니다.

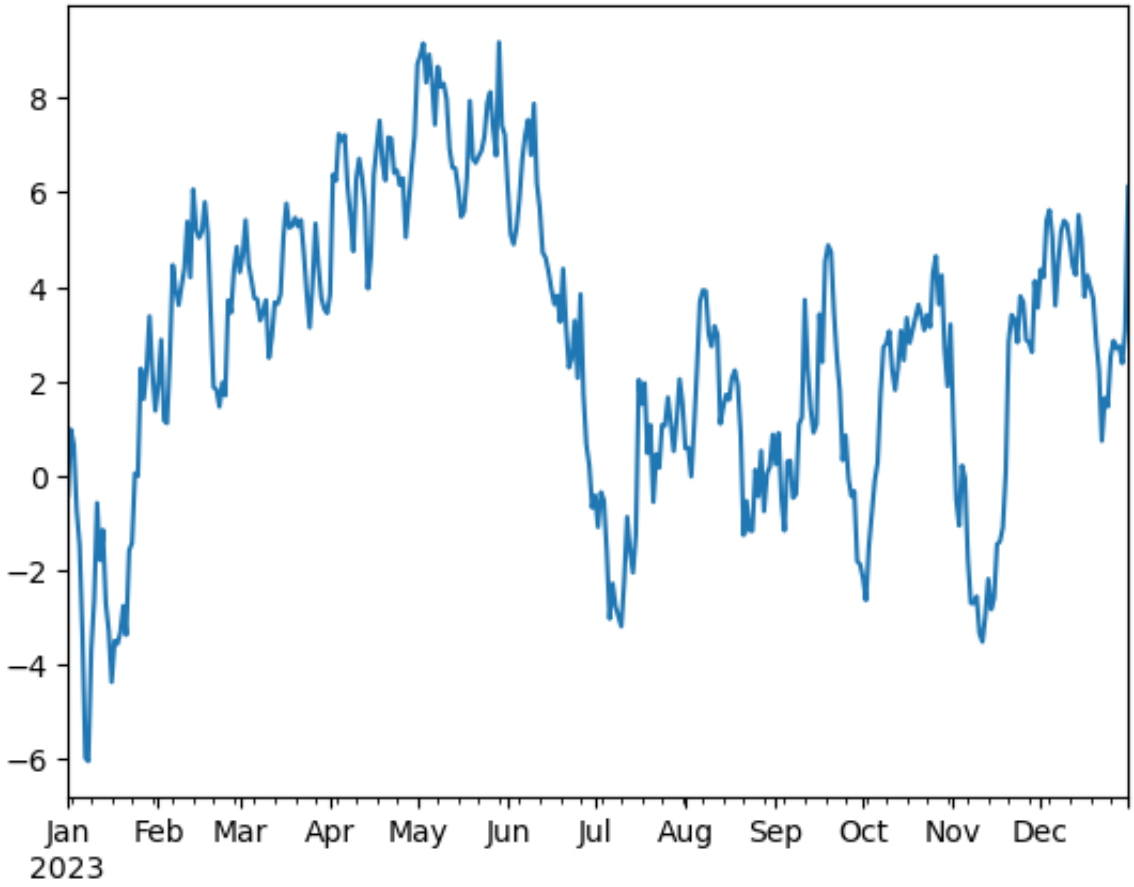
In [ ]:
```
import numpy as np
import pandas as pd

sample_data = np.random.randn(365)
date_index = pd.date_range('2023-01-01', periods=365)

ts = pd.Series(sample_data, index=date_index)
ts = ts.cumsum()

# 선 그래프 생성
ts.plot()
```

Out[ ]:
```
<Axes: >
```



In [ ]:
```
sample_data
```

Out[ ]:
```
array([-0.42109345,  1.39907851, -0.31919688, -1.39484104, -0.72799826,
       -1.94880755, -2.55927534, -0.06849717,  2.35284356,  1.05983974,
        2.05219675, -1.21235902,  0.64336973, -1.55901437, -0.57202776,
       -1.08891333,  0.87134021, -0.06409934,  0.24363021,  0.56084194,
       -0.61606022,  1.78615329,  0.16273751,  1.47336481, -0.06195846,
        2.27877372, -0.64184587,  0.62096986,  1.128046  , -1.19614544,
       -0.80145163,  0.57525895,  0.92404131, -1.70965003, -0.05377913,
        1.44031085,  1.903525  , -0.51279173, -0.32991208,  0.40672984,
        0.36556157,  0.99150525, -1.17853713,  1.86130254, -0.84140326,
       -0.18324783,  0.13046881,  0.62343859, -0.67294899, -1.67588509,
       -1.56622938, -0.05545034, -0.35574316,  0.50909662, -0.27390243,
        2.01487912, -0.26239329,  0.91388069,  0.47688807, -0.53357601,
        0.34796542,  0.75319747, -0.92892562, -0.34299875, -0.379623  ,
       -0.02352247, -0.44834405,  0.1881762 ,  0.24134142, -1.22135017,
        0.43477624,  0.74223513, -0.03637242,  0.20566705,  1.23033251,
        0.68952135, -0.51211604,  0.04103981,  0.1709633 , -0.1845177 ,
        0.13612663, -0.70846885, -0.8560026 , -0.71191063,  0.9391887 ,
        1.26077439, -0.9021614 , -0.67076688, -0.22737103, -0.09668457,
        0.39986322,  2.53397548, -0.12932182,  0.99054667, -0.14693506,
        0.12327721, -1.07456861, -0.62419313, -0.76138615,  1.54424141,
        0.41497044, -0.39724291, -0.61809944, -1.73162724,  0.67722698,
        1.72893917,  0.59596295,  0.55160911, -0.89351246, -0.37181758,
        0.91151987, -0.02695865, -0.72685828,  0.05013031, -0.31094419,
        0.14968789, -1.25110032,  0.74898516,  0.73357236,  0.66426137,
        1.51549305,  0.19275062,  0.24975588, -0.8276483 ,  0.59246021,
```

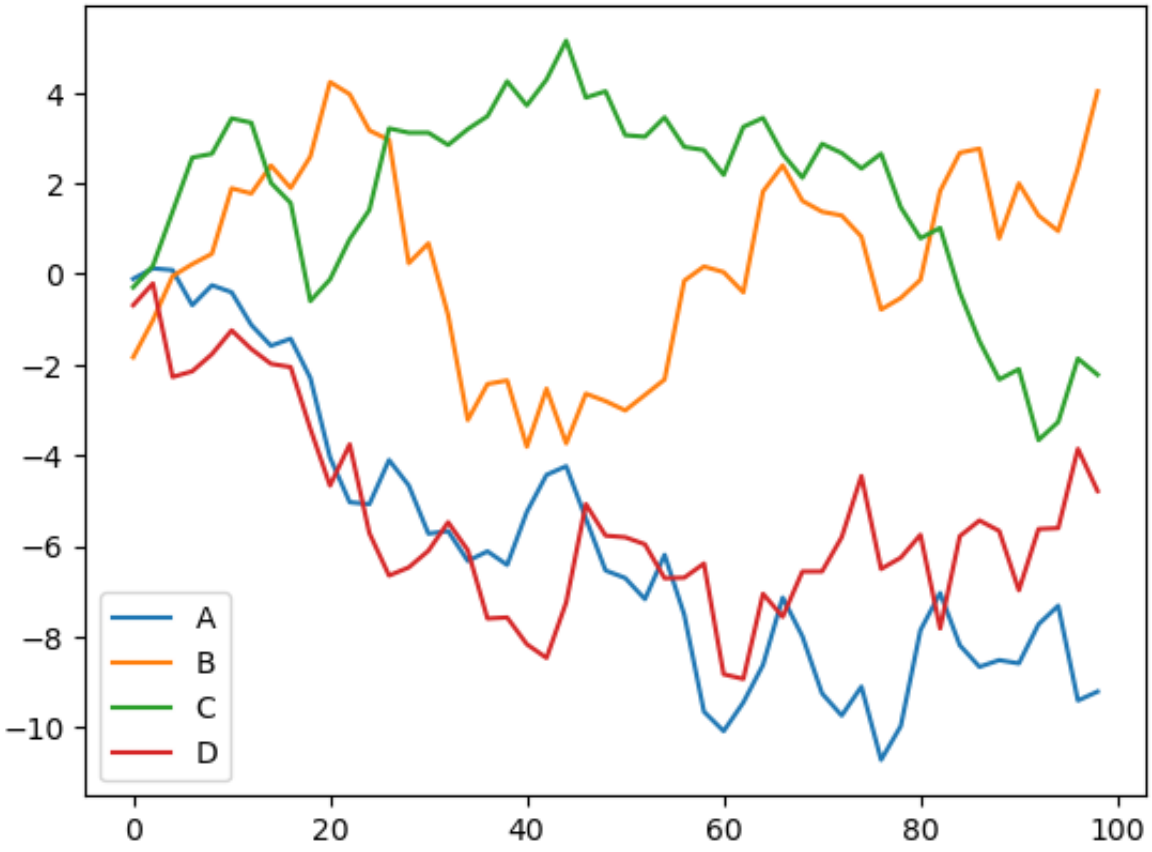
```
       -0.53570959, -0.94965904,  1.22516331, -0.42472468,  0.06599284,
       -0.32297763, -1.04645114, -0.40197141, -0.01806818, -0.42232671,
       -0.60595766,  0.11038795,  0.735517  ,  1.60848251, -1.2138538 ,
       -0.09908751,  0.1399966 ,  0.1199963 ,  0.29965835,  0.71319118,
        0.22761708, -0.77737594, -0.57002886,  2.40146056, -1.7641902 ,
       -0.19717823, -1.10480028, -0.99122364, -0.22126564,  0.33738454,
        0.61366866,  0.85227359,  0.50507505,  0.32533107, -0.74433428,
        1.08853018, -1.68171118, -0.54026121, -0.92577709, -0.11204046,
       -0.29515215, -0.37386552, -0.31540186,  0.18275411, -0.55790512,
        1.12494258, -1.17787438, -0.91375438,  0.19446187,  0.7995718 ,
       -1.21254853,  1.77023196, -2.11443677, -1.07162172, -0.43305825,
       -0.89828867,  0.2580163 , -0.66662776,  0.73608634, -0.17151808,
       -1.07344662, -1.4315497 ,  0.74526354, -0.48186035, -0.22023095,
       -0.20605591,  0.97570596,  1.34016677, -0.71037993, -0.46633074,
        0.80028964,  3.28071093, -0.51179969,  0.43899711, -1.48324943,
        0.59934132, -1.63057992,  1.01292801, -0.28919549,  0.90772647,
       -0.0159089 ,  0.59728294, -0.53544329, -0.59958724,  0.64288439,
        0.87794137, -0.50647488, -0.95707347,  0.01343405, -0.60723035,
        1.07235499,  1.23842783,  1.38996583,  0.24050457, -0.01467757,
       -0.91807767, -0.25736008,  0.43247916, -0.18719603, -1.87917504,
        0.35314477,  0.25872078, -0.10821496,  0.46539539,  0.15761094,
       -0.32441069, -1.02389095, -2.13530794,  0.72038115, -0.62394408,
       -0.00951777,  1.29374774, -0.56239416,  0.96294525, -1.27748779,
        0.78796301,  0.15644605,  0.676128  , -0.62753301,  0.66215724,
       -1.38612929, -0.67768798,  1.45762831,  0.02052553, -0.78712673,
        0.07973967,  1.47290192,  0.1460942 ,  2.49346499, -1.46596077,
       -0.82546447, -0.52006263,  0.1802447 ,  2.32353809, -1.00852786,
        2.12858567,  0.34224396, -0.142481  , -1.26281633, -0.95518339,
       -0.7086957 , -1.4799239 ,  0.52232533, -0.91347759, -0.37340412,
        0.10888644, -1.48492247, -0.06589148, -0.29110272, -0.46692637,
        1.14023025,  0.70913189,  0.67045472,  0.39396741,  1.44121291,
        0.99697287,  0.08483882,  0.25623927, -0.80365879, -0.43967554,
        0.48740332,  0.76682513, -0.62457709,  0.89587932, -0.53005243,
        0.26670421,  0.26603701,  0.28052042, -0.22642429, -0.31762319,
        0.32688772, -0.26141416,  1.13567499,  0.36678533, -1.01823174,
        0.61300708, -1.53418144, -0.8130074 ,  1.30723131, -2.05555995,
       -1.64947952, -0.5524445 ,  1.27195838, -0.27287106, -1.70215781,
       -0.93766754, -0.00387978,  0.14591423, -0.78340271, -0.176561  ,
        0.5622353 ,  0.7670325 , -0.64412587,  0.23851343,  1.13994896,
        0.05788007,  0.29923575,  1.16436647,  2.77015207,  0.56238002,
       -0.09014283, -0.48990343,  0.97606507, -0.12556914, -0.8007381 ,
       -0.04854795, -0.2078943 ,  1.50719267, -0.56761605,  0.80925855,
       -0.17044008,  1.20010386,  0.2289509 , -0.57700087, -1.44594005,
        0.87047277,  0.70234134,  0.22576769, -0.07207145, -0.39777207,
       -0.49325223, -0.18806496,  1.26737552, -0.50413709, -1.22357842,
        0.45043302, -0.24037228, -0.23920104, -0.8723622 , -0.64677377,
       -1.50087327,  0.90804738, -0.18603358,  1.05532864,  0.34204467,
       -0.16482004,  0.04832788, -0.36177983,  0.71122002,  3.02272094])
```

In [ ]:
```
date_index
```

Out[ ]:
```
DatetimeIndex(['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04',
               '2023-01-05', '2023-01-06', '2023-01-07', '2023-01-08',
               '2023-01-09', '2023-01-10',
               ...
               '2023-12-22', '2023-12-23', '2023-12-24', '2023-12-25',
               '2023-12-26', '2023-12-27', '2023-12-28', '2023-12-29',
               '2023-12-30', '2023-12-31'],
              dtype='datetime64[ns]', length=365, freq='D')
```

In [ ]:
```
df = pd.DataFrame(np.random.randn(50, 4).cumsum(axis=0),
                  columns=['A', 'B', 'C', 'D'],
                  index=np.arange(0, 100, 2))
df.plot()
```

Out[ ]: <Axes: >



2. 막대 그래프

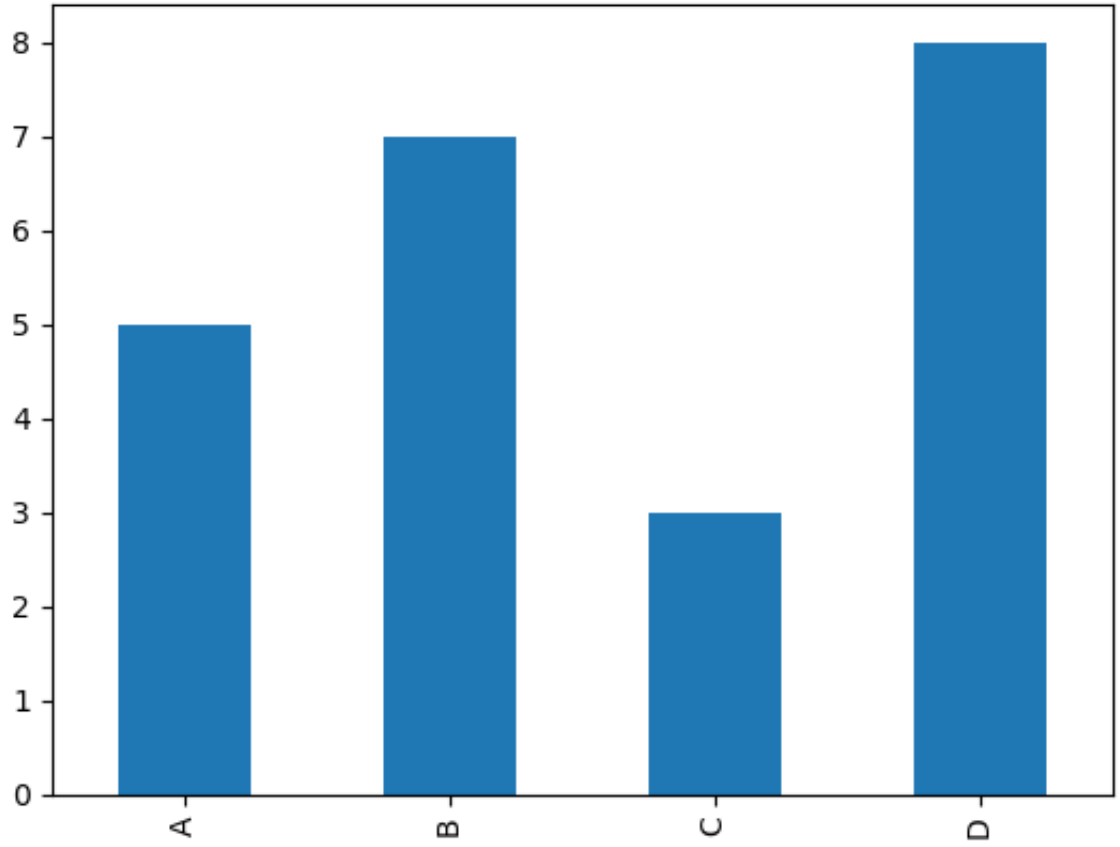
막대 그래프는 범주가 있는 데이터를 시각화하는 데 유용합니다. Series 객체 또는 DataFrame 객체에 대하여 '.plot(kind='bar')' 또는 '.plot.bar()' 메서드를 사용하여 막대 그래프를 그릴 수 있습니다.

In [ ]:
```
import matplotlib.pyplot as plt

# 시리즈 데이터 생성
series_data = pd.Series([5, 7, 3, 8], index=['A', 'B', 'C', 'D'])

series_data.plot(kind='bar')
# plt.title('Bar Chart from Series')
# plt.xlabel('Categories')
# plt.ylabel('Values')
# plt.xticks(rotation=0)
# plt.show()
```

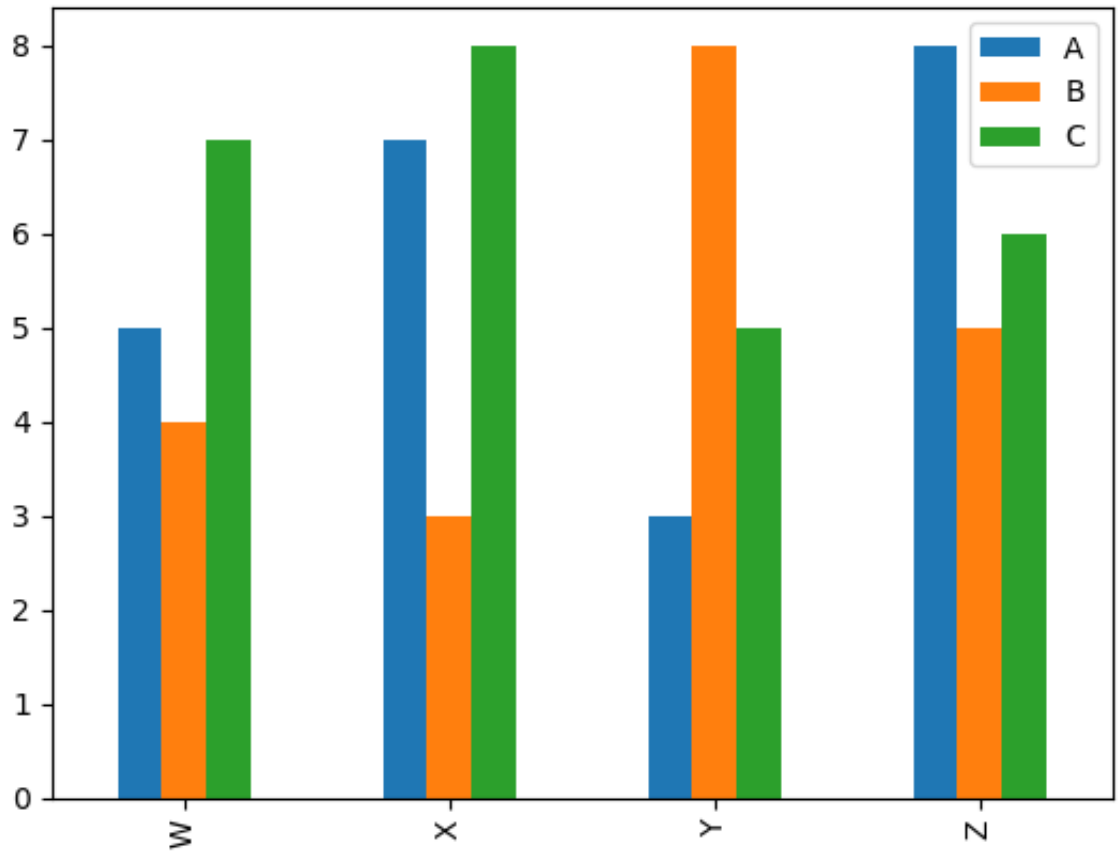
Out[ ]: <Axes: >



In [ ]:
```
# 데이터프레임 데이터 생성
df_data = pd.DataFrame({
    'A': [5, 7, 3, 8],
    'B': [4, 3, 8, 5],
    'C': [7, 8, 5, 6]
}, index=['W', 'X', 'Y', 'Z'])

# 막대 그래프 그리기
df_data.plot(kind='bar')
# plt.title('Bar Chart from DataFrame')
# plt.xlabel('Categories')
# plt.ylabel('Values')
# plt.xticks(rotation=0)
# plt.show()
```

Out[ ]: <Axes: >



3. 히스토그램

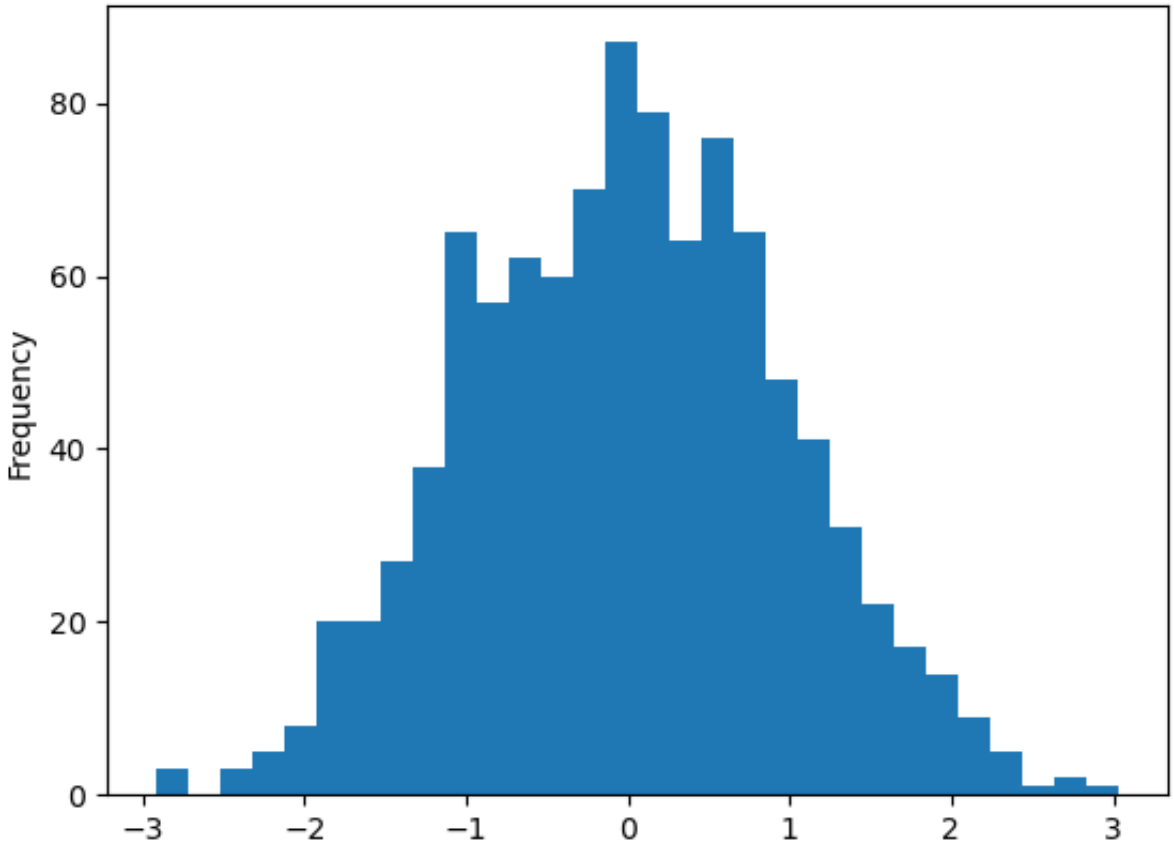
히스토그램은 데이터의 분포를 시각화하는 데 사용됩니다. Series 객체 또는 DataFrame 객체에 대하여 '.plot(kind='hist')' 또는 '.plot.hist()' 메서드를 사용하여 히스토그램을 그릴 수 있습니다.

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

# 시리즈 데이터 생성
series_data = pd.Series(np.random.randn(1000))

# 시리즈 데이터를 이용한 히스토그램 그리기
series_data.plot(kind='hist', bins=30)
# series_data.hist(bins=30)
#plt.title('Histogram from Series')
#plt.xlabel('Value')
#plt.ylabel('Frequency')
#plt.show()
```

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

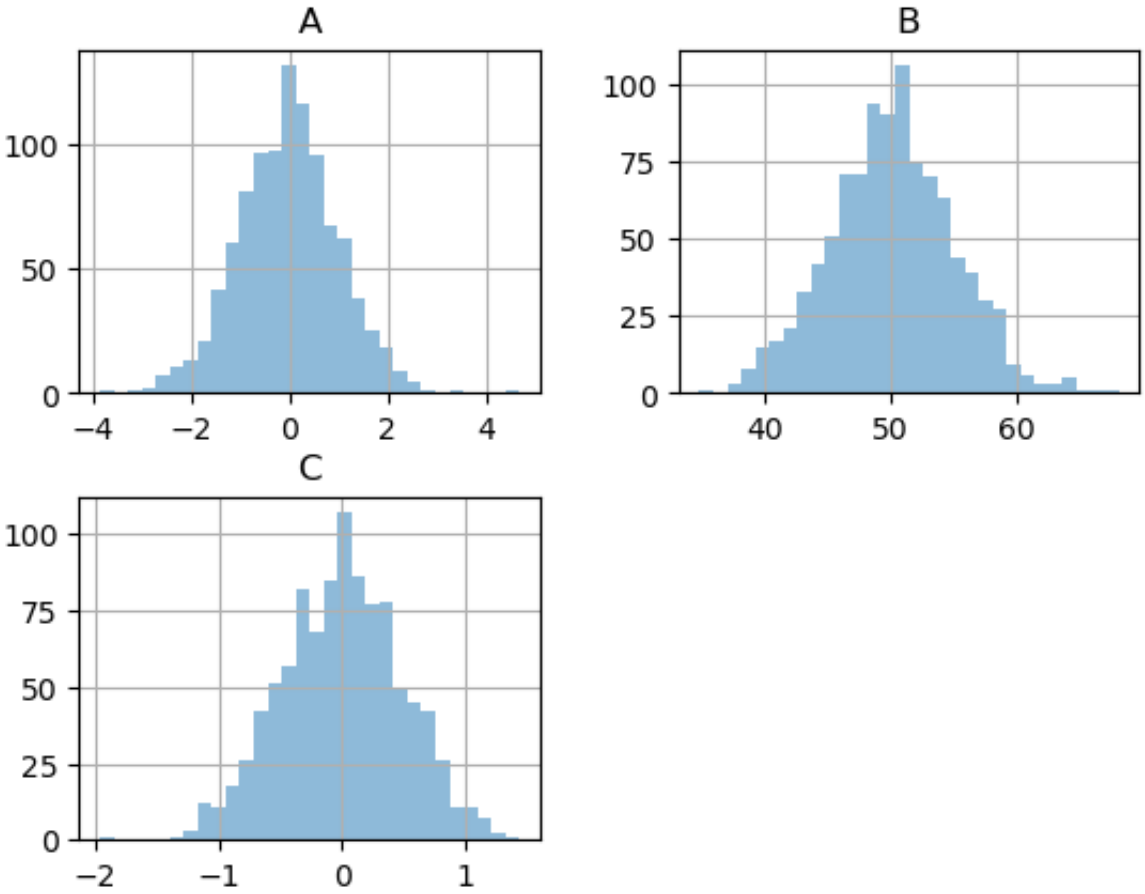


DataFrame 객체에서는 '.plot(kind='hist')' 또는 '.hist()' 메서드를 사용하여 모든 컬럼에 대한 히스토그램을 한 번에 그릴 수 있습니다.

In [ ]:
```
# 데이터프레임 데이터 생성
df_data = pd.DataFrame({
    'A': np.random.randn(1000),
    'B': np.random.randn(1000)*5 + 50,
    'C': np.random.randn(1000)/2
})

# 데이터프레임 데이터를 이용한 히스토그램 그리기
df_data.hist(bins=30, alpha=0.5)
# plt.suptitle('Histograms from DataFrame')
# plt.show()
```

Out[ ]:
```
array([[<Axes: title={'center': 'A'}>, <Axes: title={'center': 'B'}>],
       [<Axes: title={'center': 'C'}>, <Axes: >]], dtype=object)
```



4. 산점도

산점도는 두 변수 간의 관계를 시각적으로 파악하는 데 유용한 도구입니다.
'DataFrame.plot.scatter()' 메서드를 사용하여 산점도를 그릴 수 있습니다. 이 메서드는 x축과 y축에 해당하는 두 컬럼을 지정해야 합니다.

아래 예시에서는 두 개의 난수 배열을 생성하여 DataFrame의 두 컬럼 X와 Y로 사용하고 있습니다. plot.scatter() 메서드를 호출하면서 x와 y 매개변수에 각각 X컬럼과 Y컬럼을 지정함으로써 산점도를 그립니다.

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

# 데이터프레임 생성
df = pd.DataFrame({
    'X': np.random.randn(100),
    'Y': np.random.randn(100) + 1
})

# 산점도 그리기
df.plot.scatter(x='X', y='Y')
# plt.title('Scatter Plot from DataFrame')
# plt.show()
```

Out[ ]: <Axes: xlabel='X', ylabel='Y'>