

## 데이터 시각화

- 지도시각화
- 네트워크 시각화
- 텍스트 시각화
- 이미지 시각화

In [ ]: !pip install folium requests

```
Requirement already satisfied: folium in c:\Users\colab\Anaconda3\lib\site-packages (0.19.5)
Requirement already satisfied: requests in c:\Users\colab\Anaconda3\lib\site-packages (2.31.0)
Requirement already satisfied: branca>=0.6.0 in c:\Users\colab\Anaconda3\lib\site-packages (from folium) (0.8.1)
Requirement already satisfied: jinja2>=2.9 in c:\Users\colab\Anaconda3\lib\site-packages (from folium) (3.1.2)
Requirement already satisfied: numpy in c:\Users\colab\Anaconda3\lib\site-packages (from folium) (1.24.3)
Requirement already satisfied: xyzservices in c:\Users\colab\Anaconda3\lib\site-packages (from folium) (2022.9.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\Users\colab\Anaconda3\lib\site-packages (from requests) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\Users\colab\Anaconda3\lib\site-packages (from requests) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\Users\colab\Anaconda3\lib\site-packages (from requests) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\Users\colab\Anaconda3\lib\site-packages (from requests) (2023.7.22)
Requirement already satisfied: MarkupSafe>=2.0 in c:\Users\colab\Anaconda3\lib\site-packages (from jinja2>=2.9->folium) (2.1.1)
```

In [3]: import folium  
import random  
from IPython.display import display

```
# 날씨 데이터를 표시할 도시 목록
cities = [
    {"name": "서울", "lat": 37.5665, "lon": 126.9780},
    {"name": "부산", "lat": 35.1796, "lon": 129.0756},
    {"name": "대구", "lat": 35.8714, "lon": 128.6014},
    {"name": "인천", "lat": 37.4563, "lon": 126.7052},
    {"name": "광주", "lat": 35.1595, "lon": 126.8526},
    {"name": "대전", "lat": 36.3504, "lon": 127.3845},
    {"name": "울산", "lat": 35.5384, "lon": 129.3114},
    {"name": "세종", "lat": 36.4800, "lon": 127.2890},
    {"name": "제주", "lat": 33.4996, "lon": 126.5312},
]

# 날씨 상태 목록
weather_conditions = ["맑음", "흐림", "비", "눈", "안개"]

# 지도 생성
m = folium.Map(location=[36.5, 127.5], zoom_start=7)

# 각 도시의 가상 날씨 데이터 생성 및 마커 추가
for city in cities:
    # 가상의 온도 (10° C ~ 30° C)
```

```

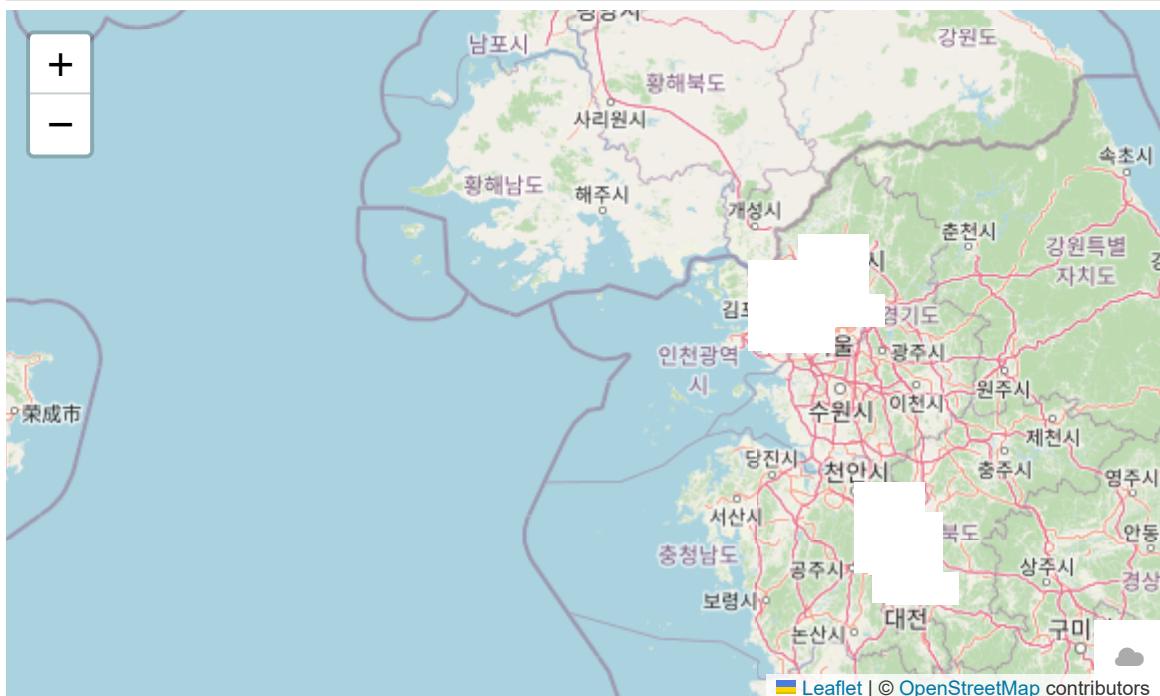
temp = round(random.uniform(10, 30), 1)
# 무작위 날씨 상태 선택
weather = random.choice(weather_conditions)

# 날씨에 따른 아이콘 선택
if weather == "맑음":
    icon = "sun"
elif weather == "흐림":
    icon = "cloud"
elif weather == "비":
    icon = "rain"
elif weather == "눈":
    icon = "snow"
else:
    icon = "fog"

# 마커 및 팝업 생성
folium.Marker(
    location=[city['lat'], city['lon']],
    popup=f'{city['name']}: {temp}° C, {weather}',
    tooltip=city['name'],
    icon=folium.Icon(color="red", icon=icon)
).add_to(m)

# 지도 표시
display(m)

```



## 이미지 처리 및 컴퓨터 비전 도구

### 1. OpenCV:

- 컴퓨터 비전과 이미지 처리를 위한 강력한 오픈소스 라이브러리입니다.
- 이미지 및 비디오 처리, 객체 감지, 얼굴 인식, 움직임 추적, 기계 학습 등 다양한 컴퓨터 비전 기능을 제공합니다.
- C++로 작성되었지만 Python, Java 등 다양한 언어에서 사용 가능한 바인딩을 제공합니다.
- 실시간 처리에 최적화되어 있으며, 멀티 코어 처리를 지원합니다.

- 500개 이상의 알고리즘과 함수를 포함하고 있어 이미지 필터링, 기하학적 변환, 색상 공간 변환, 윤곽선 검출 등 다양한 작업을 수행할 수 있습니다.
- OpenCV는 opencv-python 패키지로 설치할 수 있으며, GUI 기능이 제외된 opencv-python-headless 버전도 있습니다.

## 2. opencv-python-headless:

- OpenCV의 헤드리스 버전으로, GUI 관련 기능(예: `imshow`)이 제외된 OpenCV 패키지입니다.
- 주로 서버 환경이나 GUI가 필요 없는 환경에서 이미지 처리 및 컴퓨터 비전 작업을 수행할 때 사용됩니다.

## 3. pillow:

- Python Imaging Library(PIL)의 현대적인 대체 라이브러리입니다.
- 이미지 열기, 편집, 저장 등의 작업을 지원하며, 다양한 이미지 포맷을 처리할 수 있습니다.

## 4. numpy:

- 다차원 배열 및 행렬 연산을 위한 라이브러리입니다.
- 수학적 연산, 선형 대수, 통계, 배열 조작 등 과학 계산에 필수적인 기능을 제공합니다.

## 5. matplotlib:

- 데이터 시각화를 위한 라이브러리로, 2D, 3D 그래프와 플롯을 생성하는 데 사용됩니다.
- 선 그래프, 막대 그래프, 산점도 등 다양한 시각화 도구를 제공합니다.

```
In [5]: !pip install opencv-python-headless pillow numpy matplotlib
```

```
Requirement already satisfied: opencv-python-headless in c:\Users\colab\Anaconda3\lib\site-packages (4.11.0.86)
Requirement already satisfied: pillow in c:\Users\colab\Anaconda3\lib\site-packages (9.4.0)
Requirement already satisfied: numpy in c:\Users\colab\Anaconda3\lib\site-packages (1.24.3)
Requirement already satisfied: matplotlib in c:\Users\colab\Anaconda3\lib\site-packages (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\Users\colab\Anaconda3\lib\site-packages (from matplotlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\Users\colab\Anaconda3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\Users\colab\Anaconda3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\Users\colab\Anaconda3\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\Users\colab\Anaconda3\lib\site-packages (from matplotlib) (23.1)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\Users\colab\Anaconda3\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\Users\colab\Anaconda3\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\Users\colab\Anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

```
In [4]: import cv2
import numpy as np
from PIL import Image, ImageDraw, ImageFont
import matplotlib.pyplot as plt
import urllib.request

# 이미지 다운로드 함수
def download_image(url, filename):
```

```

urllib.request.urlretrieve(url, filename)

# 이미지 URL (공개 도메인 이미지)
image_url = "https://upload.wikimedia.org/wikipedia/commons/3/3c/Giant_Panda_2004-03-
image_filename = "panda.jpg"

# 이미지 다운로드
download_image(image_url, image_filename)

# 이미지 로드
image = cv2.imread(image_filename)
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# 1. 이미지 필터 적용 (예: 가우시안 블러)
blurred = cv2.GaussianBlur(image_rgb, (15, 15), 0)

# 2. 엣지 감지
edges = cv2.Canny(image_rgb, 100, 200)

# 3. 객체 감지 (여기서는 얼굴 감지를 예로 들겠습니다)
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalfac-
gray = cv2.cvtColor(image_rgb, cv2.COLOR_RGB2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.3, 5)

# 결과 이미지 생성
result = image_rgb.copy()
for (x, y, w, h) in faces:
    cv2.rectangle(result, (x, y), (x+w, y+h), (255, 0, 0), 2)

# Pillow를 사용하여 결과 표시
pil_image = Image.fromarray(result)
draw = ImageDraw.Draw(pil_image)
font = ImageFont.load_default()
draw.text((10, 10), "Detected Objects: " + str(len(faces)), font=font, fill=(255, 255, 255))

# 결과 표시
plt.figure(figsize=(20, 10))

plt.subplot(2, 2, 1)
plt.title("Original Image")
plt.imshow(image_rgb)
plt.axis('off')

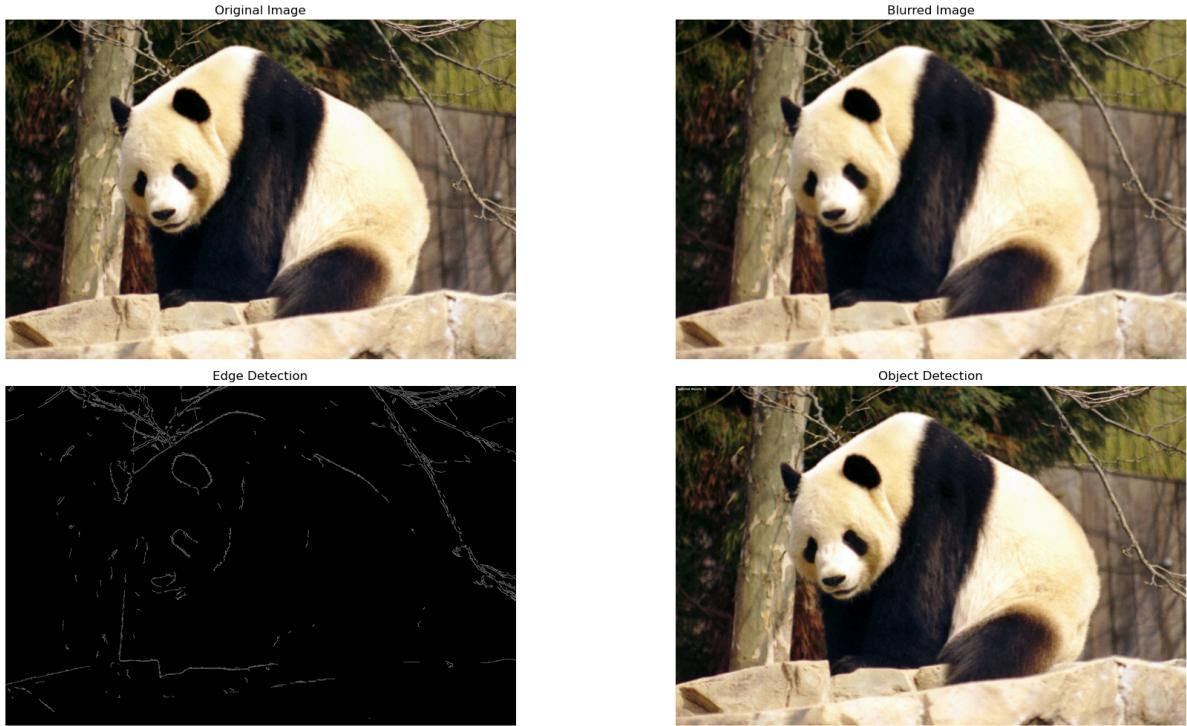
plt.subplot(2, 2, 2)
plt.title("Blurred Image")
plt.imshow(blurred)
plt.axis('off')

plt.subplot(2, 2, 3)
plt.title("Edge Detection")
plt.imshow(edges, cmap='gray')
plt.axis('off')

plt.subplot(2, 2, 4)
plt.title("Object Detection")
plt.imshow(pil_image)
plt.axis('off')

plt.tight_layout()
plt.show()

```



- 판다 이미지를 다운로드합니다.
- 원본 이미지, 블러 처리된 이미지, 엣지 감지 결과를 생성합니다.
- 얼굴 감지를 시도합니다 (판다 이미지에서는 사람 얼굴이 감지되지 않을 것입니다).
- 결과를 4개의 서브플롯으로 표시합니다.

### 1. 가우시안 블러 (Gaussian Blur)

- 가우시안 블러는 **이미지를 부드럽게 하여 노이즈를 줄이는 기법입니다.**
- 각 픽셀의 값을 주변 픽셀들과 가우시안 분포에 따라 가중 평균하여 계산합니다. 이를 통해 이미지의 세부 사항을 흐리게 만들어 노이즈를 감소시키고, 엣지 감지와 같은 후속 처리의 정확도를 높입니다.

### 2. 엣지 감지 (Edge Detection)

- 엣지 감지는 이미지에서 밝기 변화가 급격한 부분을 찾아내는 기법으로, 물체의 경계를 식별하는 데 사용됩니다.
- 대표적인 방법으로는 **캐니 엣지 감지기(Canny Edge Detector)** 가 있으며, 이는 이미지의 그라디언트를 계산하고, 비최대 억제와 이중 임계값을 적용하여 정확한 엣지를 검출합니다.

### 1. 객체 감지 (Object Detection)

- 객체 감지는 이미지나 비디오에서 특정 객체를 찾아내고 그 위치를 식별하는 기법입니다.
- 머신 러닝과 딥 러닝 알고리즘을 활용하여 객체의 특징을 학습하고, 새로운 이미지에서 해당 객체를 탐지합니다.
- 예를 들어, 얼굴 인식을 위해 **하르 캐스케이드 분류기(Haar Cascade Classifier)** 나 딥 러닝 기반의 **YOLO(You Only Look Once)** 와 같은 모델이 사용됩니다.

```
In [5]: import cv2
import numpy as np
from PIL import Image, ImageDraw, ImageFont
import matplotlib.pyplot as plt
import urllib.request
```

```

# 이미지 다운로드 함수
def download_image(url, filename):
    urllib.request.urlretrieve(url, filename)

# 이미지 URL (공개 도메인 이미지)
image_url = "https://upload.wikimedia.org/wikipedia/commons/3/37/Dagestani_man_and_woman.jpg"
image_filename = "people.jpg"

# 이미지 다운로드
download_image(image_url, image_filename)

# 이미지 로드
image = cv2.imread(image_filename)
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# 1. 이미지 필터 적용 (예: 가우시안 블러)
blurred = cv2.GaussianBlur(image_rgb, (15, 15), 0)

# 2. 엣지 감지
edges = cv2.Canny(image_rgb, 100, 200)

# 3. 얼굴 감지
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
gray = cv2.cvtColor(image_rgb, cv2.COLOR_RGB2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.3, 5)

# 결과 이미지 생성
result = image_rgb.copy()
for (x, y, w, h) in faces:
    cv2.rectangle(result, (x, y), (x+w, y+h), (255, 0, 0), 2)

# Pillow를 사용하여 결과 표시
pil_image = Image.fromarray(result)
draw = ImageDraw.Draw(pil_image)
font = ImageFont.load_default()
draw.text((10, 10), f"Detected Faces: {len(faces)}", font=font, fill=(255, 255, 255))

# 결과 표시
plt.figure(figsize=(20, 15))

plt.subplot(2, 2, 1)
plt.title("Original Image")
plt.imshow(image_rgb)
plt.axis('off')

plt.subplot(2, 2, 2)
plt.title("Blurred Image")
plt.imshow(blurred)
plt.axis('off')

plt.subplot(2, 2, 3)
plt.title("Edge Detection")
plt.imshow(edges, cmap='gray')
plt.axis('off')

plt.subplot(2, 2, 4)
plt.title("Face Detection")
plt.imshow(pil_image)
plt.axis('off')

plt.tight_layout()
plt.show()

```



## 코드 작업 수행

- 두 명의 사람이 있는 이미지를 다운로드합니다.
- 원본 이미지, 블러 처리된 이미지, 엣지 감지 결과를 생성합니다.
- 얼굴 감지를 수행하고 감지된 얼굴 주위에 사각형을 그립니다.
- 결과를 4개의 서브플롯으로 표시합니다.

## 성능 개선

## 남자의 얼굴이 인식되지 않은 주요 원인

- 얼굴 각도: 남자의 얼굴이 카메라를 정면으로 바라보고 있지 않을 수 있습니다. 옆모습이나 각도가 있는 얼굴은 인식하기 어려울 수 있습니다.
  - 조명 조건: 그림자나 불균형한 조명으로 인해 얼굴의 특징이 잘 보이지 않을 수 있습니다.
  - 얼굴 가림: 수염, 모자, 안경 등이 얼굴의 일부를 가리고 있을 수 있습니다.
  - 해상도 및 화질: 이미지의 해상도가 낮거나 화질이 좋지 않으면 얼굴 감지가 어려울 수 있습니다.
  - 알고리즘의 한계: 사용된 Haar Cascade 분류기는 간단하고 빠르지만, 복잡한 상황에서는 정확도가 떨어질 수 있습니다.
- 
- cv2.createCLAHE를 사용하여 **CLAHE (Contrast Limited Adaptive Histogram Equalization)** 를 적용함으로써 이미지의 대비를 향상
  - 조명 조건이 좋지 않은 이미지에서 객체 검출 성능을 향상시키는데 도움
  - 얼굴 검출 시 detectMultiScale 함수의 파라미터 값이 변경
    - 첫 번째 코드에서는 scaleFactor=1.3, minNeighbors=5를 사용하였으나, 두 번째 코드에서는 scaleFactor=1.1, minNeighbors=5, minSize=(30, 30)로 설정되었습니다. 이러한 조정은 검출기의 민감도를 조절하여 더 정확한 얼굴 검출을 가능하게 합니다.

```
In [6]: import cv2
import numpy as np
from PIL import Image, ImageDraw, ImageFont
import matplotlib.pyplot as plt
import urllib.request

def download_image(url, filename):
    urllib.request.urlretrieve(url, filename)

image_url = "https://upload.wikimedia.org/wikipedia/commons/3/37/Dagestani_man_and_woman.jpg"
image_filename = "people.jpg"

download_image(image_url, image_filename)

image = cv2.imread(image_filename)
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# 이미지 전처리: 대비 향상
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
gray = cv2.cvtColor(image_rgb, cv2.COLOR_RGB2GRAY)
gray = clahe.apply(gray)

blurred = cv2.GaussianBlur(image_rgb, (15, 15), 0)
edges = cv2.Canny(blurred, 100, 200)

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

result = image_rgb.copy()
for (x, y, w, h) in faces:
    cv2.rectangle(result, (x, y), (x+w, y+h), (255, 0, 0), 2)

pil_image = Image.fromarray(result)
draw = ImageDraw.Draw(pil_image)
font = ImageFont.load_default()
draw.text((10, 10), f"Detected Faces: {len(faces)}", font=font, fill=(255, 255, 255))
```

```
plt.figure(figsize=(20, 15))

plt.subplot(2, 2, 1)
plt.title("Original Image")
plt.imshow(image_rgb)
plt.axis('off')

plt.subplot(2, 2, 2)
plt.title("Blurred Image")
plt.imshow(blurred)
plt.axis('off')

plt.subplot(2, 2, 3)
plt.title("Edge Detection")
plt.imshow(edges, cmap='gray')
plt.axis('off')

plt.subplot(2, 2, 4)
plt.title("Face Detection")
plt.imshow(pil_image)
plt.axis('off')

plt.tight_layout()
plt.show()
```



Original Image



Blurred Image



Edge Detection



Face Detection

- 이 코드에서는 CLAHE(Contrast Limited Adaptive Histogram Equalization)를 사용하여 이미지의 대비를 향상시켰고, detectMultiScale 함수의 파라미터를 조정.

## 네트워크 그래스 생성

```
In [7]: import networkx as nx
import matplotlib.pyplot as plt

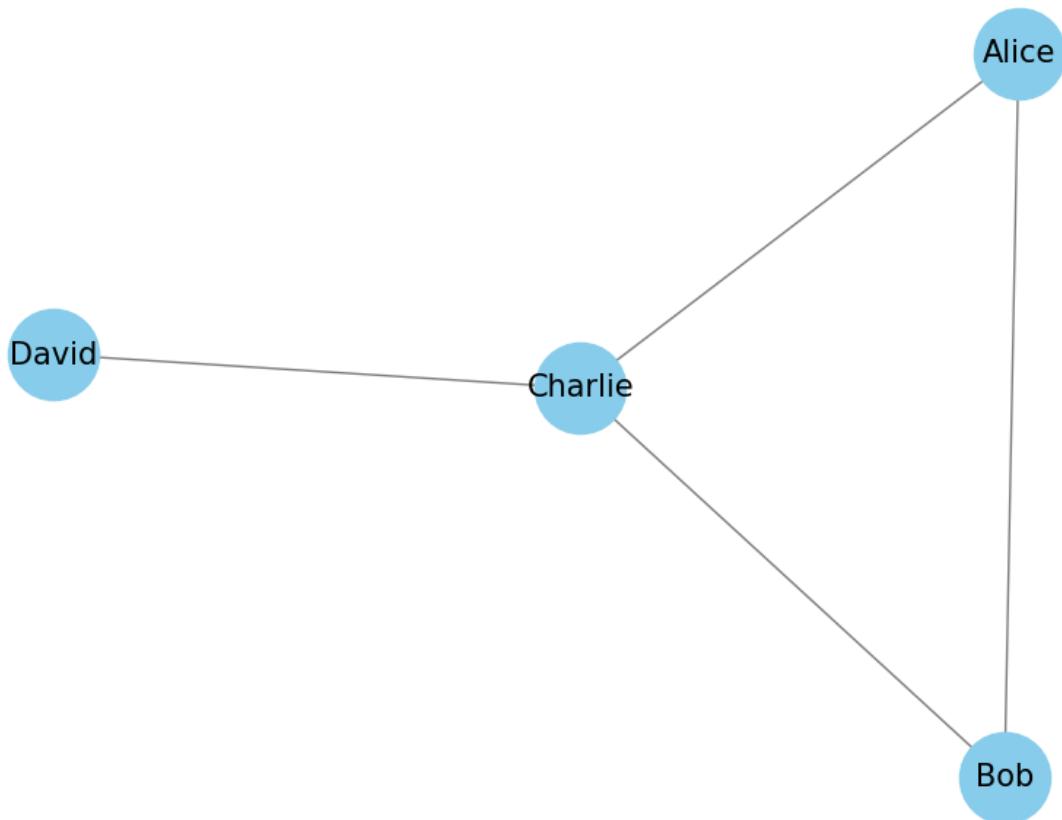
# 빈 그래프 생성
G = nx.Graph()

# 노드 추가
G.add_node("Alice")
G.add_node("Bob")
G.add_node("Charlie")
G.add_node("David")

# 엣지(연결) 추가
G.add_edges_from([( "Alice", "Bob"), ("Alice", "Charlie"), ("Bob", "Charlie"), ("Charlie", "David")])

# 그래프 시각화
plt.figure(figsize=(8, 6))
nx.draw(G, with_labels=True, node_color='skyblue', node_size=2000, font_size=15, font_weight='bold')
plt.title("Social Network")
plt.show()
```

Social Network



## 워드 클라우드 예제

```
In [6]: !pip install wordcloud
```

Requirement already satisfied: wordcloud in c:\Users\colab\Anaconda3\lib\site-packages (1.9.4)  
Requirement already satisfied: numpy>=1.6.1 in c:\Users\colab\Anaconda3\lib\site-packages (from wordcloud) (1.24.3)  
Requirement already satisfied: pillow in c:\Users\colab\Anaconda3\lib\site-packages (from wordcloud) (9.4.0)  
Requirement already satisfied: matplotlib in c:\Users\colab\Anaconda3\lib\site-packages (from wordcloud) (3.7.2)  
Requirement already satisfied: contourpy>=1.0.1 in c:\Users\colab\Anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.0.5)  
Requirement already satisfied: cycler>=0.10 in c:\Users\colab\Anaconda3\lib\site-packages (from matplotlib->wordcloud) (0.11.0)  
Requirement already satisfied: fonttools>=4.22.0 in c:\Users\colab\Anaconda3\lib\site-packages (from matplotlib->wordcloud) (4.25.0)  
Requirement already satisfied: kiwisolver>=1.0.1 in c:\Users\colab\Anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.4.4)  
Requirement already satisfied: packaging>=20.0 in c:\Users\colab\Anaconda3\lib\site-packages (from matplotlib->wordcloud) (23.1)  
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\Users\colab\Anaconda3\lib\site-packages (from matplotlib->wordcloud) (3.0.9)  
Requirement already satisfied: python-dateutil>=2.7 in c:\Users\colab\Anaconda3\lib\site-packages (from matplotlib->wordcloud) (2.8.2)  
Requirement already satisfied: six>=1.5 in c:\Users\colab\Anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)

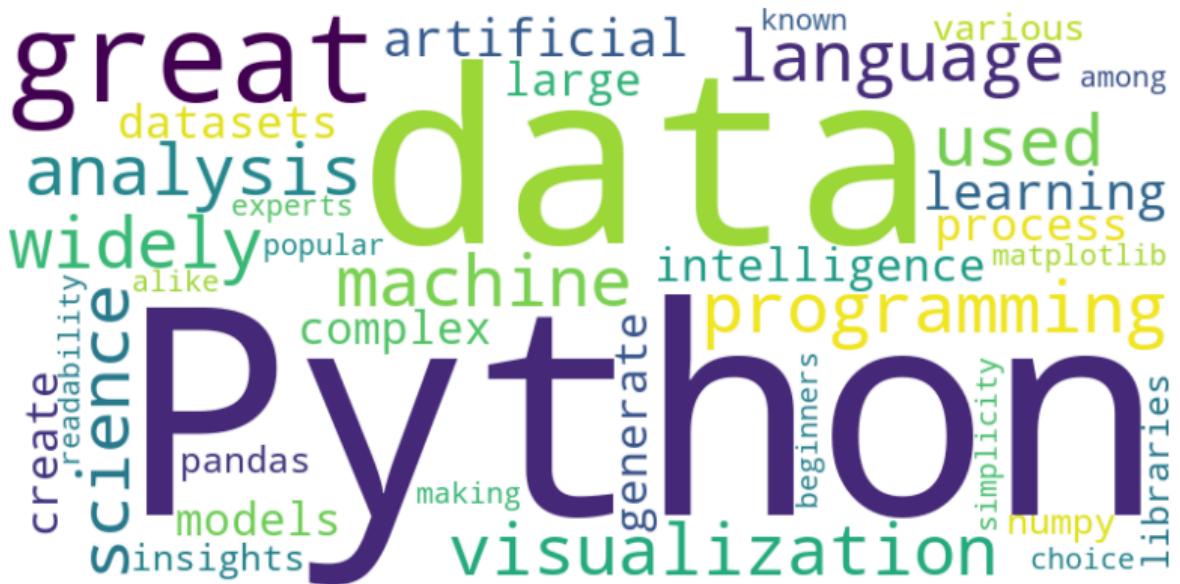
In [7]:

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# 텍스트 데이터
text = """
Python is a great programming language for data analysis and visualization.
It is widely used in data science, machine learning, and artificial intelligence.
With Python, you can create complex models, process large datasets, and generate
insights through various libraries like pandas, numpy, and matplotlib.
Python is also known for its simplicity and readability, making it a popular choice
among beginners and experts alike.
"""

# 워드 클라우드 생성
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text)

# 워드 클라우드 시각화
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



In [ ]: