

CNN 설명

[\[딥러닝 모델\] CNN \(Convolutional Neural Network\) 설명](#)

머신러닝(딥러닝) 함수 설명

활성화 함수 (Activation Function)

활성화 함수는 각 노드에서 가중치가 곱해진 값을 다른 특정한 값으로 바꾸어준다. 활성화 함수를 쓰는 이유는 활성화 함수를 쓰지 않으면 hidden layer가 많아져도 하나의 선형연산으로 나타낼 수 있어 망이 깊어지지 않기 때문이다. 활성화 함수의 종류는 다음과 같다.

Rectified Linear Units (ReLU)

- 뉴런의 On-Off를 흉내내지는 않지만
- 딥러닝 성능을 크게 향상시키는 경향이 있어 널리 활용됨

하이퍼볼릭 탄젠트 (tanh)

- 시그모이드에 비해 뉴런의 On-Off를 더 잘 흉내낼 수 있음
- 최근 음수값을 사용하지 않으려는 추세로 선호도 낮아짐

Sigmoid 함수

- 가장 무난한 형태로 널리 활용됨
- binary 예측에 사용
- 정규화된 회귀에서도 사용 가능

Softmax Activation Function

- 출력층에서 사용
- 카테고리 예측에 사용
- 총합은 1이며, 출력은 0~1 사이의 값이므로 확률로 해석할 수 있다.

손실함수 (Loss Function)

모델은 오차를 최소화하는 방향으로 학습하는데, 이 오차를 정의하는 함수가 손실함수이다.

손실함수의 종류에는 다음 것들이 있다. 평균 오차 정도는 이해가 쉽게 되는데, 엔트로피 오차는 원지도 모르겠다. 사실 대학원생 아니면 그냥 이런게 있구나 이름 정도만 알면 된다. 어차피 이름 하나만 쓰면 알아서 계산해준다.

평균 제곱 오차 (MSE, Mean Squared Error)

- 회귀 수치 구할 때

평균 제곱근 오차 (RMSE, Root Mean Squared Error)

- 위와 사실상 같음 (위에 루트만 씌움)

이진 교차 엔트로피 오차 (BCEE, Binary Cross Entropy Error)

- 이진 분류 구할 때
- 활성화 함수로 주로 sigmoid를 사용

범주형 교차 엔트로피 오차 (CCEE, Categorical Cross Entropy Error)

- 다항 분류 구할때
- 활성화 함수로 주로 softmax를 사용

YOLO8 시연 예제

```
pip install ultralytics
pip install opencv-python
```

최신 YOLO 버전인 YOLOv8 (현재 기준으로 가장 널리 사용되는 Ultralytics의 최신 안정화 버전 중 하나)을 사용하여 실시간 카메라 영상에서 객체 인식을 하고, 객체별 이름과 카운팅 정보를 표시하는 파이썬 코드 예제

```
import cv2
from ultralytics import YOLO
from collections import defaultdict

# YOLOv8 모델 로드 (기본적으로 'yolov8n.pt'는 나노 버전으로 빠르고 가볍습니다)
model = YOLO("yolov8s.pt") # 'yolov8s.pt', 'yolov8m.pt' 등으로 더 높은 성능 모델
# 사용 가능

...

yolov8n (Nano): 가장 작고 빠른 모델, 리소스가 제한된 환경에 적합.
yolov8s (Small): 속도와 정확도의 균형이 좋은 모델.
yolov8m (Medium): 중간 크기, 더 높은 정확도를 제공.
yolov8l (Large): 높은 정확도를 목표로 하는 대형 모델.
yolov8x (Extra Large): 최대 정확도를 위한 가장 큰 모델.

yolov8n: 3.2M 매개변수, 8.7 GFLOPs.
yolov8s: 11.2M 매개변수, 28.6 GFLOPs.
yolov8m: 25.9M 매개변수, 78.9 GFLOPs.
yolov8l: 43.7M 매개변수, 165.2 GFLOPs.
yolov8x: 68.2M 매개변수, 257.8 GFLOPs.

COCO 80개 클래스 목록 (참고용, 일부 예시)
사람(person), 자전거(bicycle), 자동차(car), 오토바이(motorbike),
비행기(airplane), 버스(bus), 기차(train), 트럭(truck), 보트(boat), 신호등(traffic
light), 소화전(fire hydrant), 표지판(stop sign), 주차 미터기(parking meter),
벤치(bench), 새(bird), 고양이(cat), 개(dog), 말(horse), 양(sheep), 소(cow),
코끼리(elephant), 곰(bear), 얼룩말(zebra), 기린(giraffe), 배낭(backpack),
우산(umbrella), 핸드백(handbag), 넥타이(tie), 여행가방(suitcase),
```

프리스비(frisbee), 스키(skis), 스노보드(snowboard), 공(sports ball), 연(kite), 야구 배트(baseball bat), 야구 글러브(baseball glove), 스케이트보드(skateboard), 서핑보드(surfboard), 테니스 라켓(tennis racket), 병(bottle), 와인잔(wine glass), 컵(cup), 포크(fork), 나이프(knife), 숟가락(spoon), 그릇(bowl), 바나나(banana), 사과(apple), 샌드위치(sandwich), 오렌지(orange), 브로콜리(broccoli), 당근(carrot), 핫도그(hot dog), 피자(pizza), 도넛(donut), 케이크(cake), 의자(chair), 소파(couch), 화분(potted plant), 침대.bed), 식탁(dining table), 화장실(toilet), TV, 노트북(laptop), 마우스(mouse), 리모컨(remote), 키보드(keyboard), 휴대폰(cell phone), 전자레인지(microwave), 오븐(oven), 토스터(toaster), 싱크대(sink), 냉장고(refrigerator), 책(book), 시계(clock), 꽃병(vase), 가위(scissors), 테디베어(teddy bear), 헤어드라이어(hair drier), 칫솔(toothbrush).

YOLOv8은 NVIDIA GPU에서 CUDA와 cuDNN을 지원하므로 GPU 가속을 권장합니다.

최소 사양:

GPU: NVIDIA GTX 1650 (4GB VRAM).

CPU: Intel Core i5 또는 AMD Ryzen 5.

RAM: 16GB.

용도: yolov8n, yolov8s로 실시간 추론 또는 소규모 학습.

권장 사양 (모델별):

yolov8n (Nano):

GPU: NVIDIA GTX 1660 Ti (6GB VRAM).

RAM: 16GB.

실시간 추론: 30~60 FPS 가능 (640x640 해상도 기준).

yolov8s (Small):

GPU: NVIDIA RTX 3060 (6~12GB VRAM).

RAM: 16~32GB.

학습: 소규모 데이터셋(1,000~5,000 이미지) 처리 가능.

yolov8m (Medium):

GPU: NVIDIA RTX 3070 (8GB VRAM).

RAM: 32GB.

학습: 중간 크기 데이터셋(5,000~10,000 이미지).

yolov8l (Large):

GPU: NVIDIA RTX 3080 (10GB VRAM) 또는 A4000.

RAM: 32~64GB.

학습 및 추론: 고해상도 영상 또는 대규모 데이터셋.

yolov8x (Extra Large):

GPU: NVIDIA RTX 3090 (24GB VRAM) 또는 A100 (40GB 이상).

RAM: 64GB 이상.

용도: 대규모 학습(수만 장 이미지) 및 최고 정확도 추론.
'''

카메라 캡처 초기화 (0은 기본 웹캠)

cap = cv2.VideoCapture(0)

해상도 설정 (선택 사항)

cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)

cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

객체 카운팅을 위한 딕셔너리

object_counts = defaultdict(int)

while True:

프레임 읽기

ret, frame = cap.read()

if not ret:

print("카메라에서 프레임을 읽을 수 없습니다.")

break

YOLOv8으로 객체 감지

results = model(frame)

결과 처리

object_counts.clear() # 매 프레임마다 카운트 초기화

for result in results:

boxes = result.boxes # 감지된 객체의 바운딩 박스

for box in boxes:

바운딩 박스 좌표

x1, y1, x2, y2 = map(int, box.xyxy[0])

confidence = box.conf[0] # 신뢰도

class_id = int(box.cls[0]) # 클래스 ID

label = model.names[class_id] # 객체 이름

신뢰도가 0.5 이상인 객체만 표시

if confidence > 0.5:

객체 카운팅

object_counts[label] += 1

바운딩 박스와 라벨 그리기

cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

cv2.putText(frame, f"{label} {confidence:.2f}", (x1, y1 - 10),

cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

화면 상단에 객체 카운팅 정보 표시

```

y_offset = 20
for obj_name, count in object_counts.items():
    count_text = f"{obj_name}: {count}"
    cv2.putText(frame, count_text, (10, y_offset),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
    y_offset += 30

# 결과 프레임 표시
cv2.imshow("YOLOv8 Real-Time Object Detection", frame)

# 'q'를 누르면 종료
if cv2.waitKey(1) & 0xFF == ord('q'):
    break

# 자원 해제
cap.release()
cv2.destroyAllWindows()

```

konlpy 사용을 위한 자바세팅 방법

KoNLPy 사용 시 **JVM** 관련 문제가 지속적으로 발생하는 경우

- 다운로드
 - <http://10.0.66.99/openlogic-openjdk-8u442-b06-windows-x64.zip>

1. JAVA_HOME 환경변수 재확인

```
# Windows의 경우 예시 경로
JAVA_HOME = C:\Program Files\Java\jdk1.8.0_xxx
```

```
# 시스템 Path에 추가되어야 할 경로
%JAVA_HOME%\bin
```

2. 자바 버전 호환성 체크

```
# 터미널에서 자바 버전 확인
java -version
```

KoNLPy는 Java 8(1.8)과 가장 호환성이 좋습니다. 다른 버전을 사용 중이라면 Java 8로 다운그레이드하는 것을 추천합니다.

3. KoNLPy 재설치

```
pip uninstall konlpy
pip install --upgrade pip
pip install konlpy
```

4. JPyype 재설치

```
pip uninstall JPyype1
pip install JPyype1
```

konlpy를 이용한 개발환경 구성시 환경설정

1. Python 설치

먼저 Python이 설치되어 있어야 합니다. [Python 공식 웹사이트](#)에서 다운로드하여 설치하세요.

2. 가상 환경 설정 (선택 사항)

가상 환경을 사용하는 것이 좋습니다. 이를 위해 **venv**를 사용할 수 있습니다.

```
python -m venv myenv
source myenv/bin/activate # Linux/Mac
myenv\Scripts\activate # Windows
```

3. 필요한 라이브러리 설치

다음으로 **konlpy**와 그 의존성을 설치합니다.

```
pip install konlpy
pip install JPyype1
```

4. Java 설치

konlpy는 Java 기반의 **JPyype**를 사용합니다. Java Development Kit (JDK)를 설치해야 합니다. [Oracle JDK 다운로드](#)에서 설치할 수 있습니다.

5. 환경 변수 설정

Java를 설치한 후, JAVA_HOME 환경 변수를 설정해야 합니다.

- Windows:
 1. 제어판 -> 시스템 및 보안 -> 시스템 -> 고급 시스템 설정 -> 환경 변수
 2. 시스템 변수에서 새로 만들기를 클릭하고, 변수 이름에 JAVA_HOME, 변수 값에 JDK 설치 경로를 입력합니다.
- Linux/Mac:

```
export JAVA_HOME=/path/to/jdk
export PATH=$JAVA_HOME/bin:$PATH
```

6. Python 코드에서 Okt 사용하기

이제 Python 코드에서 Okt를 사용할 수 있습니다. 아래는 간단한 예제입니다.

```
from konlpy.tag import Okt

okt = Okt()
text = "안녕하세요. 반갑습니다."
print(okt.morphs(text))
```

joblib과 pickle의 비교 자료(머신러닝 모델)

특성	joblib	pickle
목적	대규모 데이터 및 NumPy 배열 저장에 최적화	일반적인 객체 직렬화 및 역직렬화
성능	대규모 배열의 경우 더 빠름	소규모 객체의 경우 충분히 빠름
압축 기능	자동으로 데이터 압축 가능	압축 기능 없음
병렬 처리	지원 (예: Parallel 클래스)	지원하지 않음

사용 용도	머신러닝 모델, 대규모 데이터	다양한 Python 객체 및 데이터 구조
파일 확장자	일반적으로 .joblib 사용	일반적으로 .pkl 사용
호환성	Scikit-learn 과 같은 라이브러리와 잘 통합됨	Python 의 기본 직렬화 도구

- **joblib**: 대규모 데이터와 머신러닝 모델을 다룰 때 적합하며, 성능과 효율성을 중시하는 경우 유리합니다.
- **pickle**: 일반적인 **Python** 객체의 직렬화에 적합하며, 간단한 사용 사례에 적합합니다.

🔗 머신러닝_model_로드방법.ipynb

회귀 모델의 성능 평가 - 4가지 지표

R^2 Score (결정계수)

R^2 는 모델이 데이터 변동성을 얼마나 잘 설명하는지를 나타내는 지표입니다. 값의 범위는 0에서 1 사이이며, 1에 가까울수록 모델이 데이터를 잘 설명하는 것입니다. R^2 가 0이면 모델이 평균값으로 예측하는 것과 같고, 1이면 모든 데이터를 정확히 예측하는 것입니다.

RMSE (Root Mean Square Error, 평균 제곱근 오차)

RMSE는 예측값과 실제값 간의 차이를 제곱한 후 평균을 내고, 다시 제곱근을 취한 값입니다. 값이 작을수록 모델의 예측이 실제값에 가깝다는 것을 의미합니다. RMSE는 단위가 원래 데이터와 동일하므로 해석이 용이합니다.

MAE (Mean Absolute Error, 평균 절대 오차)

MAE는 예측값과 실제값 간의 절대적인 차이의 평균입니다. RMSE와 마찬가지로 값이 작을수록 모델의 성능이 좋음을 나타냅니다. MAE는 이상치의 영향을 덜 받기 때문에, RMSE보다 더 강건한 지표로 여겨질 수 있습니다.


MSE (Mean Squared Error, 평균 제곱 오차)

MSE는 예측값과 실제값 간의 차이를 제곱한 후 평균을 내는 지표입니다. RMSE와 비슷하지만, 제곱근을 취하지 않기 때문에 단위가 원래 데이터의 제곱이 됩니다. 값이 작을수록 모델의 예측이 더 정확하다는 것을 나타냅니다.

자기소개서 자료

- [개발자 자기소개서 완벽 가이드 \[자소서 양식 무료 공유\]](#)
- [IT 합격자소서 만능검색기](#)

최종 프로젝트 주제

 AI 프로젝트(세종).pdf

1. 사물 인식을 활용한 스마트 가로등 서비스 개발

목표: 사물 인식 기술을 활용하여 가로등의 자동 점등/소등 및 밝기 조절 기능을 구현하여 에너지 효율성과 안전성을 향상시키는 것입니다.

프로젝트 계획:

- 사물 인식 모델 개발 및 최적화
- 가로등 제어 시스템 설계 및 구현
- 실시간 환경 모니터링 및 제어 알고리즘 개발
- 사용자 인터페이스 및 관리 시스템 구축
- 실증 테스트 및 성능 평가

2. CNN을 활용한 불량 검출 스마트팩토리 개발

목표: 합성곱 신경망(CNN) 기반의 영상 처리 기술을 이용하여 제조 공정에서 발생하는 불량품을 실시간으로 감지하고 자동 분류하는 것입니다.

프로젝트 계획:

- 제조 공정 현황 분석 및 불량 데이터 수집
- CNN 모델 설계 및 학습 데이터셋 구축
- 실시간 영상 처리 및 불량 감지 알고리즘 개발
- 제조 공정 통합 및 실시간 모니터링 시스템 구축
- 성능 평가 및 최적화

3. 사물 인식을 활용한 스마트 횡단보도 서비스 개발

목표: 사물 인식 기술을 활용하여 횡단보도에 접근하는 보행자와 차량을 실시간으로 감지하고, 안전한 횡단을 위한 신호 제어 및 경고 서비스를 제공하는 것입니다.

프로젝트 계획:

- 사물 인식 모델 개발 및 최적화
- 횡단보도 센서 네트워크 및 신호 제어 시스템 설계
- 보행자 및 차량 감지 알고리즘 개발
- 경고 및 안내 서비스 구현
- 실증 테스트 및 성능 평가

프로젝트 팀 구성

1팀	박선정	안기부	황병천	김혜민	임건우	CNN을 활용한 불량검출 스마트팩토리 개발
----	-----	-----	-----	-----	-----	----------------------------

2팀	정유진	임정은	허지원	주정은	진예찬	사물인식을 활용한 스마트 횡단보도 서비스 개발
3팀	황수용	신준혁	손유빈	이지엽		사물인식을 활용한 스마트 가로등 서비스 개발

프로젝트 팀 구성을 위한 설문

- <https://forms.gle/zhcw4ty1NwcHzAJg9>

ipTIME 공유기의 RAM 크기

ipTIME 공유기의 RAM 크기에 따른 대략적인 동시 접속 규모는 다음과 같습니다:

1. 32MB RAM: 10-20대 정도의 기기 동시 접속 가능
2. 64MB RAM: 30-50대 정도의 기기 동시 접속 가능
3. 128MB RAM: 70-100대 정도의 기기 동시 접속 가능
4. 256MB RAM 이상: 100대 이상의 기기 동시 접속 가능

RAM 용량과 동시 접속 가능한 기기 수는 대체로 비례관계에 있습니다. RAM이 클수록 더 많은 인터넷 연결 상태를 유지할 수 있어, 동시다발적인 인터넷 연결에도 끊김이나 지연 문제를 최소화할 수 있습니다.

선택 시 고려사항

1. 사용 환경: 가정용, 사무실용, 공공장소용 등 용도에 따라 적절한 RAM 용량 선택
2. 연결 기기 수: 예상되는 총 연결 기기 수 고려
3. 사용 패턴: HD 영상 시청, 대용량 파일 전송 등 높은 대역폭을 요구하는 작업의 빈도 고려
4. 확장성: 향후 기기 증가 가능성을 고려하여 여유 있는 RAM 용량 선택

RAM 용량이 큰 공유기를 선택하면 동시에 많은 기기를 연결하더라도 안정적인 성능을 기대할 수 있습니다

머신러닝 분류모델 평가 지표

- 정확도(Accuracy): 전체 샘플 중에서 올바르게 예측한 샘플의 비율입니다.
- 정밀도(Precision): 올바르게 예측한 긍정 샘플의 비율입니다.
- 재현율(Recall): 실제 긍정 샘플 중 올바르게 예측한 비율입니다.

- **F1 스코어**: 정밀도와 재현율의 조화 평균으로, 두 지표의 균형을 평가합니다.

다음은 머신러닝 모델의 성능을 평가하는 주요 지표들에 대한 설명입니다. 각 지표는 모델의 예측 성능을 다양한 관점에서 평가합니다.

1. 정확도 (Accuracy)

- 정의: 전체 샘플 중에서 올바르게 예측한 샘플의 비율입니다.
- 계산식:

$$\text{정확도} = \text{올바른 예측 수} / \text{전체 샘플 수}$$

- 장점: 전체적인 성능을 간단하게 이해할 수 있습니다.
- 단점: 클래스 불균형이 있는 경우, 정확도가 높은 것만으로는 모델의 성능을 평가하기 어려울 수 있습니다.

2. 정밀도 (Precision)

- 정의: 모델이 긍정 클래스로 예측한 것 중에서 실제로 긍정인 샘플의 비율입니다.
- 계산식:

$$\text{정밀도} = \text{TP} / \text{TP} + \text{FP}$$

여기서 TP는 True Positive(올바르게 긍정으로 예측한 수), FP는 False Positive(잘못 긍정으로 예측한 수)입니다.

- 장점: 잘못된 긍정 예측을 줄이는 것이 중요한 경우(예: 스팸 필터링)에 유용합니다.
- 단점: 재현율과의 균형이 필요합니다.

3. 재현율 (Recall)

- 정의: 실제 긍정 샘플 중에서 올바르게 긍정으로 예측한 비율입니다.
- 계산식:

$$\text{재현율} = \text{TP} / \text{TP} + \text{FN}$$

여기서 FN은 False Negative(잘못 부정으로 예측한 수)입니다.

- 장점: 실제 긍정 샘플을 놓치는 것을 줄이는 것이 중요한 경우(예: 질병 진단)에 유용합니다.
- 단점: 정밀도와의 균형이 필요합니다.

4. F1 스코어 (F1 Score)

- 정의: 정밀도와 재현율의 조화 평균으로, 두 지표의 균형을 평가합니다.
- 계산식:


$$\text{F1 스코어} = 2 \times ((\text{정밀도} \times \text{재현율}) / (\text{정밀도} + \text{재현율}))$$

- 장점: 정밀도와 재현율 간의 균형을 고려하여 모델의 성능을 평가합니다.
- 단점: 정밀도나 재현율 중 하나가 과도하게 낮으면 F1 스코어도 낮아질 수 있습니다.

요약

- 정확도: 전체 예측 중 올바른 예측의 비율.
- 정밀도: 긍정으로 예측한 것 중 실제 긍정의 비율.
- 재현율: 실제 긍정 중 올바르게 긍정으로 예측한 비율.
- F1 스코어: 정밀도와 재현율의 조화 평균으로 두 지표 간의 균형 평가.

취업특강 자료

-  2025_세종교육_취업특강.pdf

미니프로젝트 발표자 선정

- <https://forms.gle/p7gVTb9aZJNQPPSw9>

참고할 만한 프로젝트 PPT

- [5조_최종발표.pptx](#) (MyWeatherDiary)
- [\[프로젝트\] 발표 ppt 만들기](#) (분실물 통합 커뮤니티)
- [\[프로젝트\] 발표 ppt 만들기](#) (세모알)

MySQL 심화

☰ Flask에서 SQLAlchemy 사용

☰ 데이터베이스 객체

데이터 모델링

☰ 데이터 모델링의 기초 이론(강의) - 공개용

☰ 게시판 모델링

☰ 프로젝트 관련

☰ “취미활동 동영상 공유 서비스”의 DB 모델링

☰ MySQL의 정규형(Normal Form)

프로토 타입틀

- <https://pixso.net/>
- 어도비 XD
- 피그마
- 제플린

MySQL 무결성(Integrity)

MySQL 무결성(Integrity)은 데이터베이스 내 데이터의 정확성, 일관성 및 신뢰성을 보장하는 개념입니다. 무결성은 여러 가지 형태로 나뉘며, 주로 다음과 같은 유형이 있습니다:

1. 개체 무결성 (Entity Integrity):
 - 각 테이블의 기본 키는 고유해야 하며, NULL 값을 가질 수 없습니다. 이를 통해 각 레코드를 유일하게 식별할 수 있습니다.
2. 참조 무결성 (Referential Integrity):

- 외래 키가 참조하는 기본 키와 일치해야 하며, 외래 키는 **NULL**이 아니거나 해당 기본 키의 값 중 하나여야 합니다. 이를 통해 테이블 간의 관계를 유지합니다.

3. 도메인 무결성 (Domain Integrity):

- 속성이 가질 수 있는 값의 범위를 정의합니다. 예를 들어, 특정 열이 정수형이어야 한다면, 그 열에 문자열이 들어갈 수 없도록 제한합니다.

4. 사용자 정의 무결성 (User-defined Integrity):

- 특정 비즈니스 규칙이나 요구 사항에 따라 정의된 무결성 규칙입니다. 예를 들어, 고객의 나이는 18세 이상이어야 한다는 규칙을 설정할 수 있습니다.

SQL 정리학습

(2025.02.03 숙제)

요걸로 정리해 보세요~ ^^ → 

도커 정리

도커 빌드 및 실행

프로젝트 폴더로 이동후

Dockerfile 파일 생성

```
# Dockerfile
FROM python:3.12-slim

# 환경 변수 설정
ENV DEBIAN_FRONTEND=noninteractive

# 필요한 패키지 설치
RUN apt-get update && \
    apt-get install -y mariadb-client && \
    apt-get clean && rm -rf /var/lib/apt/lists/*
```



```

# 작업 디렉토리 설정
WORKDIR /app

# 필요한 Python 패키지 설치
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# 애플리케이션 코드 복사
#COPY . .

# Gunicorn을 사용하여 Flask 애플리케이션 실행 (8000 포트로 실행하도록 적용)
CMD ["gunicorn", "-w", "2", "-k", "gevent", "app:app", "--bind",
"0.0.0.0:8000", "--reload"]

```

requirements.txt 파일 생성 (설치가 필요한 파이썬 라이브러리 목록)

```

flask
mysql-connector-python
gunicorn
gevent

```

도커 빌드 (이미지 생성) - 이미지명은 소문자로

```

docker build -t 이미지이름 .

```

도커 이미지를 컨테이너로 생성후 실행 (상대경로인 경우 프로젝트 있는 폴더에서 실행해야 함)

```

docker run --name 컨테이너명 -v %cd%:/app -p 8001:8000 -d 이미지명
# %cd% : 현재경로 (윈도우)
# $(pwd) : 현재경로 (리눅스)
# 내PC에서 접근방법: http://127.0.0.1:8001

```

도커 허브로 업로드

사전에 도커 허브에 가입되어 있어야 함

태깅작업

```
docker tag 이미지명 도커허브사용자명/이미지명
```

도커허브로 업로드

```
docker push 도커허브사용자명/이미지명
```

도커 이미지가 없는 **PC**에서 도커 허브로부터 다운로드시

```
docker pull 도커허브사용자명/이미지명
```

우분투 **22.04** 도커설치

- 시스템 업데이트
 - `sudo apt update`
 - `sudo apt upgrade -y`
- 필수 패키지 설치
 - `sudo apt install -y apt-transport-https ca-certificates git curl software-properties-common`
- 도커 GPG키 추가
 - `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker.gpg`
- 도커 저장소 추가
 - `echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null`
- 패키지 목록 업데이트
 - `sudo apt update`
- 도커 설치
 - `sudo apt install -y docker-ce docker-ce-cli containerd.io`
- 도커 서비스 시작 및 자동실행 처리
 - `sudo systemctl start docker`
 - `sudo systemctl enable docker`
- 도커 설치 확인(버전 확인)
 - `sudo docker --version`
- 도커 권한 설정
 - `sudo usermod -aG docker $USER`

```
- newgrp docker
```

도커간 네트워크 공유

```
docker network create my_network
```

```
docker run --name mariadb-container --network my_network -e  
MYSQL_ROOT_PASSWORD=my-secret-pw -d mariadb
```

```
docker run --name board2_container --network my_network -v %cd%:/app  
-p 8001:8000 -d board2
```

```
# 소스에서 DB접속시 host는 컨테이너 이름 지정한다  
host="mariadb-container"
```

[참고용] **docker-compose.yml** 를 이용해도 컨테이너 실행이 가능합니다.

```
version: '3.8'
```

```
services:
```

```
  mariadb:
```

```
    image: mariadb:10.6 # MariaDB 10.6 이미지 사용
```

```
    container_name: mariadb-container
```

```
    environment:
```

```
      MYSQL_ROOT_PASSWORD: my-secret-pw # 루트 비밀번호 설정
```

```
    networks:
```

```
      - my_network
```

```
    ports:
```

```
      - "3306:3306"
```

```
    volumes:
```

```
      - C:\Temp\mariadb-data:/var/lib/mysql
```

```
  board2:
```

```
    image: board2
```

```
    container_name: board2_container
```

```
    networks:
```

```
      - my_network
```

```
    ports:
```

```
      - "8001:8000"
```

```
    volumes:
```

```
      - ./app
```

```
networks:
```

```
my_network:
  driver: bridge
```

docker-compose 실행

```
docker-compose up -d
```

docker-compose 종료 및 정리

```
docker-compose down
```

도커 실행 명령어 종합

[도커 컨테이너 실행방법]

윈도우 프롬프트 창

```
$ docker pull mariadb:10.6
$ docker run --name mariadb-container -e MYSQL_ROOT_PASSWORD=1234 -e
MYSQL_DATABASE=board_db2 -p 3306:3306 -v
C:\Projects\mariadb-data:/var/lib/mysql -d mariadb:10.6

$ docker images # 로컬 저장소에 있는 도커 이미지들
$ docker ps # 실행중인 컨테이너 서비스 목록 (컨테이너ID 확인 가능)
$ docker stop <컨테이너ID> # 특정 컨테이너 서비스 멈춤
$ docker strt <컨테이너ID> # 특정 컨테이너 서비스 시작
$ docker ps -a # 모든 컨테이너 서비스 목록
$ docker rm <컨테이너ID> # 특정 컨테이너 삭제

$ docker exec -it mariadb-container /bin/bash # 컨테이너 쉘에 접속

Dockerfile 파일 작성
requirements.txt 파일 작성

$ docker build -t board2 .
$ docker run --name board2_container -v %cd%:/app -p 8001:8000 -d board2

### $(pwd):리눅스용 == %cd%:윈도우용

$ docker tag board2 s3458600/board2 # 태깅작업: Hub사용자명/이미지명
$ docker push s3458600/board2 # 도커허브로 업로드: Hub사용자명/이미지명
```

▶ AI 풀스택 개발 이란?

인공지능 모델을 구축하고 배포하는 전체 과정에 관련된 다양한 기술과 도구를 포함

AI 풀스택의 주요 구성 요소

1. 데이터 수집

- 소스: 웹 크롤링, API, 데이터베이스 등
- 도구: Beautiful Soup, Scrapy, Pandas, Selenium 등

2. 데이터 전처리

- 작업: 데이터 정제, 결측치 처리, 데이터 변환
- 도구: Pandas, NumPy, Scikit-learn 등

3. 모델 개발

- 알고리즘: 회귀, 분류, 클러스터링 등
- 프레임워크: TensorFlow, Keras 등

4. 모델 평가

- 지표: 정확도, 정밀도, 재현율, F1-score 등
- 기법: 교차 검증 등.

5. 모델 배포

- 환경: 클라우드 서비스 (AWS, GCP, Azure), Docker 등
- 도구: Flask, FastAPI 등

6. 프론트엔드 통합

- 기술: HTML, CSS, JavaScript 프레임워크 (React, Vue.js)
- 목표: 사용자 인터페이스 설계 및 구현

7. 인프라 관리

- 기술: CI/CD, DevOps 도구 (GitHub), 도커 등
- 목표: 자동화된 배포 및 테스트

이와 같은 과정은 AI 프로젝트의 성공적인 수행을 위해 필수적입니다.

▶ 취업상담

- [프로젝트와 포트폴리오는 몇 개가 좋은가요](#)

▶ 프로젝트 설계 순서 참고

<https://velog.io/@dangdang/DB-%EB%B3%91%EC%9B%90-%EA%B4%80%EB%A6%AC-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%A8-1#%EA%B2%B0%EA%B3%BC>

▶ 이력서 작성법

- [개발자 이력서 작성하기 \(feat. 이력서 공개\)](#)
- [주니어 개발자 이력서 쓰는 법](#)
- [신입 개발자는 이력서를 어떻게 작성해야 할까?](#)
- [\[이력서\]표준 포맷.docx](#) (이력서 문서 - 참고용)

☰ GitHub CI/CD 적용방법

☰ AWS 클라우드 기반 Flask 서버 구축

☰ LightSail S3 버킷을 마운트하는 방법

PPT 포함 사항

(*)는 필수

1. 개요

- a. 팀 소개 (*)
- b. 프로젝트 소개 (*)
- c. 시장조사 (*)
- d. 유사 프로그램 분석 (*)
- e. 주요기능 (*)
- f. 기대효과 (*)
- g. 개발환경 (*)

2. 프로젝트 계획 수립

- a. WBS 수립 (*)
- b. 이슈 관리 계획 수립
- c. 커뮤니케이션 계획 수립

3. 업무분석 및 요건 정의

- a. 업무 분석 (*)
- b. 요구사항 명세 (*)
- c. 요구사항 추적 매트릭스
- d. 유스케이스 다이어그램
- e. 프로토타이핑
- f. 전체 업무 흐름도 (*)
- g. 업무 목록 (*)
- h. 단위 업무 정의서
- i. 단위 업무 흐름도

4. 기획 & 설계

- a. 데이터베이스 설계
 - i. 논리 데이터베이스 모델(ERD) (*)
 - ii. 상관 모델링
 - iii. 물리 데이터베이스 모델
 - iv. 테이블 정의서 작성 (*)
- b. 기획 및 화면 설계
 - i. 정책 정의서 (*)
 - ii. Information Architecture(IA) (*)
 - iii. 메뉴 구조도(사이트 맵) (*)
 - iv. 화면목록 및 화면정의서 (스토리보드)
- c. 프로그램 설계
 - i. 프로그램 목록 및 상세 설계 (*)
- d. 테스트 설계
 - i. 단위테스트 시나리오 작성
 - ii. 통합테스트 시나리오 작성 (*)

5. 개발

- a. 개발 진행 사항 보고
- b. 형상관리

6. 테스트

- a. 단위테스트 결과서
- b. 통합테스트 결과서

7. 시연

- a. 구현 기능 설명
- b. 프로젝트 발표
- c. 프로젝트 후기

▶ 프로젝트 역량


1. 사용자의 데이터 요구사항을 분석하여 데이터베이스를 설계하는 역량
2. 사용자의 기능적, 비기능적 요구사항을 분석하여 정책을 수립할 수 있는 역량
3. 사용자의 인터페이스 요구사항을 분석하여 화면을 설계하는 역량
4. 사용자의 기능적 요구사항을 분석하여 프로그램을 설계하는 역량
5. 사용자의 요구사항에 부합하는 서비스를 테스트하기 위해 시나리오를 작성할 수 있는 역량
6. AI 표준 기술과 빅데이터 분석 결과를 출력하는 UI를 구현하며 사용자 입장에서 화면을 구현할 수 있는 구현 능력
7. github, gitlab, bitbucket 등을 이용하여 형상관리를 이용한 팀협업을 할 수 있는 역량
8. 단위테스트와 통합테스트를 진행하며 요구사항에 부합하는 서비스를 검증할 수 있는 역량

9. 폭포수 모형에 따른 프로젝트의 전체 공정에 대한 이해를 기반으로 프로젝트를 수행할 수 있는 역량
10. notion, slack 등을 활용하여 팀협업을 위한 도구를 사용할 수 있는 역량
11. 프로젝트 제안 및 결과보고 시 효율적인 내용을 전달하기 위한 커뮤니케이션 역량

☰ (주)세종교육 AIOT 교육이수내용

☰ Flask 및 Gunicorn 설정

미니프로젝트

- [대본.txt](#)
-  gza ppt_최종.pptx
-  FUNDINO.pptx

Git 관련 자료

Git 정리.pdf

- 윈도우에서 venv 실행
 - "venv/Scripts/activate"
- 리눅스에서 venv 실행
 - source venv/bin/activate
- 같은 프로젝트에 서로다른 별칭으로 다른 원격저장소 추가 가능
 - git remote add origin [원격저장소 URL]
 - git remote add origin2 [원격저장소 URL2]
- 특정 별칭에 매핑된 원격저장소 URL 변경시
 - git remote set-url origin [원격저장소 URL]
- 짧은 명령어 구현방법 - Python venv 대상(우분투용)
 - echo "alias venv='source /home/내계정/flask_app/venv/bin/activate'" >> ~/.bashrc

- source ~/.bashrc

미니프로젝트 준비 자료

☰ 프로젝트 관련

☰ LLM 활용 프로젝트 사례 자료수집

[프로젝트 및 상담자료 설문조사](#)

매출 계산 **SQL** 쿼리

1. 총 매출액 계산:

```
SELECT SUM(quantity * price) AS total_revenue FROM sales
```

2. 제품별 총 매출액 계산:

```
SELECT product_name, SUM(quantity * price) AS total_revenue FROM sales  
GROUP BY product_name
```

3. 제품별 평균 판매 가격 계산:

```
SELECT product_name, AVG(price) AS avg_price  
FROM sales GROUP BY product_name;
```

4. 월별 총 판매량 계산:

```
SELECT DATE_FORMAT(sale_date, '%Y-%m') AS `MONTH`,  
SUM(quantity) AS total_quantity  
FROM sales GROUP BY `MONTH` ORDER BY `MONTH`;
```

```
SELECT LEFT(sale_date, 7) AS `MONTH`,  
SUM(quantity) AS total_quantity  
FROM sales GROUP BY `MONTH` ORDER BY `MONTH`;
```

5. 제품별 최대/최소 판매량 계산:

```
SELECT product_name,  
MAX(quantity) AS max_quantity,  
MIN(quantity) AS min_quantity
```

```
FROM sales GROUP BY product_name;
```

6.제품별 총 판매량 계산:

```
SELECT product_name, SUM(quantity) AS total_quantity  
FROM sales GROUP BY product_name;
```

```
SELECT product_name, SUM(quantity) AS total_quantity  
FROM sales  
WHERE DATE_FORMAT(sale_date, '%Y-%m') = '2023-02'  
GROUP BY product_name ;
```

7.가장 많이 팔린 제품 Top 2 구하기:

```
SELECT product_name, SUM(quantity) AS total_quantity  
FROM sales GROUP BY product_name  
ORDER BY total_quantity DESC LIMIT 2
```

8.제품별 총 수익 대비 판매량 비율 계산:

```
SELECT product_name,  
SUM(quantity * price) / SUM(quantity) AS revenue_per_unit  
FROM sales GROUP BY product_name;
```

9.가장 높은 평균 가격의 제품 구하기:

```
SELECT product_name, AVG(price) AS avg_price  
FROM sales GROUP BY product_name  
ORDER BY avg_price DESC LIMIT 1;
```

매출계산용 테이블

```
CREATE TABLE sales (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  product_name VARCHAR(50),  
  quantity INT,  
  price DECIMAL(10,2),  
  sale_date DATE  
);  
  
INSERT INTO sales (product_name, quantity, price, sale_date)
```

VALUES

```
('Product A', 10, 9.99, '2023-01-01'),
('Product B', 5, 19.99, '2023-01-02'),
('Product A', 8, 9.99, '2023-01-03'),
('Product C', 12, 14.99, '2023-01-04'),
('Product B', 3, 19.99, '2023-01-05'),
('Product A', 6, 9.99, '2023-02-01'),
('Product C', 9, 14.99, '2023-02-02'),
('Product B', 7, 19.99, '2023-02-03'),
('Product A', 4, 9.99, '2023-02-04'),
('Product C', 11, 14.99, '2023-02-05');
```

데이터베이스 종류별 상위목록 가져오는 방법

```
#[MySQL/MariaDB]
SELECT * FROM table_name LIMIT 10;
SELECT * FROM table_name LIMIT 10, 20;

#[PostgreSQL]
SELECT * FROM table_name LIMIT 10;
SELECT * FROM table_name OFFSET 10 LIMIT 20;

#[Oracle]
SELECT * FROM (
    SELECT rownum AS rnum, t.*
    FROM table_name t
) WHERE rnum BETWEEN 1 AND 10;

SELECT * FROM (
    SELECT rownum AS rnum, t.*
    FROM table_name t
) WHERE rnum BETWEEN 11 AND 30;

#[SQL Server]
SELECT TOP 10 * FROM table_name;

SELECT * FROM (
    SELECT ROW_NUMBER() OVER (ORDER BY column1) AS rnum, *
    FROM table_name
) AS subquery
WHERE rnum BETWEEN 11 AND 30;
```

```
#[SQLite]
SELECT * FROM table_name LIMIT 10;
SELECT * FROM table_name LIMIT 20 OFFSET 10;
```

MySQL에서 제공하는 주요 데이터 타입

구분	데이터 타입	설명	(최대) 크기(범위)
문자형	CHAR(n)	고정 길이	n <= 255
	VARCHAR(n)	가변 길이	n <= 65,535
	TINYTEXT		255Byte
	TEXT		64KB
	MEDIUMTEXT		16MB
	LONGTEXT		4GB
	TINYBLOB		255Byte
	BLOB		64KB
	MEDIUMBLOB		16MB
	LOBLOB		4GB
	ENUM	목록에 있는 값만 입력 가능	'Y', 'N'
숫자형	TINYINT	정수	-128~127 Unsigned 0~255
	SMALLINT	정수	-32,768~32,767 Unsigned 0~65,535
	MEDIUMINT	정수	-8,388,608~8,388,607 Unsigned 0~16,777,215
	INT	정수	-2,147,483,648~2,147,483,647 Unsigned 0~4,294,967,295
	BIGINT	정수	-9,223,372,036,854,775,808 ~9,223,372,036,854,775,807

			Unsigned 0~18,446,744,073,709,551,615
	DECIMAL[(M,D)]	실수	소수점 이하 자릿수 포함 최대 65자리 숫자
	DOUBLE[(M,D)]	실수	-1.7976931348623157E+308~ -2.2250738585072014E-308, 0, 2.2250738585072014E-308 ~1.7976931348623157E+308
날짜형	DATE	날짜	1000-01-01~9999-12-31 (1000-01-01 이전 날짜 입력 가능)
	DATETIME	날짜와 시간	1000-01-01 00:00:00.000000 ~9999-12-31 11:59:59.999999 (1000-01-01 이전 날짜 입력 가능)
	TIME	시간	-838:59:59.000000 ~838:59:59.000000
	YEAR	연도	0000, 1901~2155

Jinja2 중첩 블록

부모 템플릿의 블록을 오버라이드하면서 내부의 다른 블록도 함께 오버라이드할 수 있습니다. 복잡한 템플릿 구조를 효율적으로 관리하는데 유용합니다.

사용 예시

부모 템플릿

```

<!-- base.html -->
{% block content %}
    <h1>기본 제목</h1>
    <div class="form-container">
        {% block form %}
            <form method="post">
                {% block form_fields %}
                    <input type="text" name="username" placeholder="사용자
이름">

                {% endblock %}

                {% block form_buttons %}
                    <button type="submit">제출</button>
                {% endblock %}
            </form>
        {% endblock %}
    </div>
{% endblock %}

```

```

        </form>
    {% endblock %}
</div>
{% endblock %}

```

자식 템플릿

```

<!-- child.html -->
{% extends "base.html" %}

{% block form %}
    <form method="post">
        {% block form_fields %}
            <input type="email" name="email" placeholder="이메일">
        {% endblock %}

        {% block form_buttons %}
            <button type="submit">로그인</button>
        {% endblock %}
    </form>
{% endblock %}

```

소스 설명

- 부모 템플릿에서는 **content** 블록 안에 **form** 블록을 정의하고, 그 안에 **form_fields**와 **form_buttons** 블록을 포함하고 있습니다.
- 자식 템플릿에서는 **form** 블록을 오버라이드하면서, 부모 템플릿의 **form_fields**와 **form_buttons** 블록을 각각 다시 정의합니다. 이 경우, 부모의 내용은 모두 사라지고 자식에서 정의한 내용만 사용됩니다.

Jinja2 템플릿 엔진 주요 기능

1. 기본 문법

```

{# 주석 #}
{{ 변수 }} - 출력
{% 문장 %} - 문법

```

2. 변수 출력

```
<!-- 기본 변수 출력 -->
{{ name }}

<!-- 딕셔너리 접근 -->
{{ user.name }}
{{ user['name'] }}

<!-- 리스트/배열 접근 -->
{{ users[0] }}

<!-- 필터 적용 -->
{{ name|capitalize }}
{{ text|truncate(100) }}
```

3. 제어 구조

```
<!-- 변수 선언 -->
{% set items = ['사과', '바나나', '체리'] %}

<!-- if 조건문 -->
{% if user %}
    Hello, {{ user.name }}!
{% elif guest %}
    Hello, Guest!
{% else %}
    Hello, Stranger!
{% endif %}

<!-- for 반복문 -->
{% for item in items %}
    {{ item.name }}
{% else %}
    No items found.
{% endfor %}

<!-- Loop 변수 -->
{% for item in items %}
    {{ loop.index }}      {# 1부터 시작하는 인덱스 #}
    {{ loop.index0 }}     {# 0부터 시작하는 인덱스 #}
```

```

    {{ loop.first }}      {# 첫 번째 반복인지 #}
    {{ loop.last }}       {# 마지막 반복인지 #}
    {{ loop.length }}     {# 전체 항목 수 #}
{% endfor %}

```

[loop변수 예제]

```

{% set items = ['사과', '바나나', '체리'] %}

<ul>
{% for item in items %}
    <li>
        {{ loop.index }}: {{ item }}      {# 1부터 시작하는 인덱스 #}
        (Index0: {{ loop.index0 }})      {# 0부터 시작하는 인덱스 #}
        {% if loop.first %}
            (첫 번째 항목입니다)
        {% endif %}
        {% if loop.last %}
            (마지막 항목입니다)
        {% endif %}
        (전체 항목 수: {{ loop.length }}) {# 전체 항목 수 #}
    </li>
{% endfor %}
</ul>

```

html파일을 Flask 템플릿화 하는 과정

1. app.py 생성
 - a. 기본 구조

```

from flask import Flask, url_for, render_template
app = Flask(__name__)

if __name__ == '__main__':
    app.run(debug=True)

```

2. templates 폴더 생성
 - a. 템플릿 엔진을 사용하기 위해 필수 과정
3. static 폴더 생성

- a. 리소스 파일(css, js, images 등)을 넣는 위치로 사용
 - b. URL에서 직접 접근 가능 (ex) /static/...
4. 원본 html 파일들을 templates 폴더로 복사
 - a. 전체 레이아웃에서 각 메뉴별로 중복된 부분과 바뀌는 부분에 대한 소스분석
5. 전체 레이아웃이 동일한 파일중 하나를 base.html로 복사
6. base.html 파일을 상속받는 자식 html이 수정해도 되는 부분을 block으로 지정
 - a. 재사용 가능한 컴포넌트 영역은 include로 처리
 - b. 컴포넌트 영역은 해당 html소스를 별도 파일로 저장한 다음 include 진행
7. app.py 에서 각 메뉴에 해당하는 URL별 라우팅 작업진행

send_from_directory 설명

지정된 디렉토리에서 안전하게 정적 파일을 제공하는 Flask 함수입니다.

```
from flask import send_from_directory

@app.route('/downloads/<path:filename>')
def download_file(filename):
    return send_from_directory('static/download', filename)
```

주요 특징:

- 첫 번째 인자: 앱 루트 기준 디렉토리 이름
- 두 번째 인자: 제공할 파일 이름
- 자동 보안 검사 수행
- 적절한 HTTP 헤더 반환
- 파일시스템에서 직접 파일을 제공하는 것보다 안전

👉 Flask 템플릿 필터

flask 버전에 따라 지원되지 않는 필터가 있을 수 있습니다.
이 경우 아래와 같은 방법으로 직접 필터를 만들어서 사용해야 합니다.

```
@app.template_filter('currency')
def format_currency(value):
    return "{:, .2f}".format(value)
```

1. **default:**
 - 변수가 정의되지 않았거나 **None** 값을 가질 경우 기본값을 지정할 수 있습니다.
 - 예시: `{{ variable|default('기본값') }}`
2. **capitalize:**
 - 문자열의 첫 글자를 대문자로 변환합니다.
 - 예시: `{{ 'hello world'|capitalize }}`
3. **lower:**
 - 문자열을 모두 소문자로 변환합니다.
 - 예시: `{{ 'HELLO WORLD'|lower }}`
4. **upper:**
 - 문자열을 모두 대문자로 변환합니다.
 - 예시: `{{ 'hello world'|upper }}`
5. **title:**
 - 문자열의 각 단어 첫 글자를 대문자로 변환합니다.
 - 예시: `{{ 'hello world'|title }}`
6. **trim:**
 - 문자열 양 끝의 공백을 제거합니다.
 - 예시: `{{ ' hello world '|trim }}`
7. **length:**
 - 변수의 길이(문자열 길이, 리스트 길이 등)를 반환합니다.
 - 예시: `{{ 'hello world'|length }}`
8. **replace:**
 - 문자열 내의 특정 문자를 다른 문자로 바꿉니다.
 - 예시: `{{ 'hello world'|replace('world', 'flask') }}`
9. **format:**
 - 문자열 내의 형식화된 값을 치환합니다.
 - 예시: `{{ '{ }'|format('hello', 'world') }}`
10. **safe:**
 - **HTML** 태그가 포함된 문자열을 이스케이프 처리하지 않고 그대로 출력합니다.
 - 예시: `{{ 'hello world'|safe }}`
11. **date:**
 - 날짜 데이터를 원하는 형식으로 출력할 수 있습니다.
 - 예시: `{{ post.created_at|date('%Y-%m-%d') }}`
12. **time:**
 - 시간 데이터를 원하는 형식으로 출력할 수 있습니다.
 - 예시: `{{ post.created_at|time('%H:%M:%S') }}`
13. **datetime:**
 - 날짜와 시간 데이터를 원하는 형식으로 출력할 수 있습니다.
 - 예시: `{{ post.created_at|datetime('%Y-%m-%d %H:%M:%S') }}`
14. **int:**
 - 숫자 데이터를 정수형으로 변환합니다.
 - 예시: `{{ '3.14'|int }}`
15. **float:**

- 숫자 데이터를 부동 소수점 형식으로 변환합니다.
- 예시: `{{ '3.14'|float }}`

16. truncate:

- 문자열을 지정된 길이로 자릅니다.
- 예시: `{{ 'This is a long string'|truncate(10) }}`
- 기본적으로 문자열 끝에 ...을 붙입니다.

17. wordwrap:

- 문자열을 지정된 길이로 줄바꿈합니다.
- 예시: `{{ 'This is a long string'|wordwrap(8) }}`
- 기본적으로 공백 문자를 기준으로 줄바꿈합니다.

18. striptags:

- 문자열에서 HTML 태그를 제거합니다.
- 예시: `{{ '<p>This is a test</p>'|striptags }}`

19. escape:

- 문자열에서 HTML 특수문자를 이스케이프 처리합니다.
- 예시: `{{ '<p>This is a test</p>'|escape }}`

20. indent:

- 문자열의 각 줄 앞에 지정된 문자를 삽입합니다.
- 예시: `{{ 'Hello\\n\\nWorld'|indent(4, ' ') }}`
- 기본적으로 4개의 공백 문자를 삽입합니다.

21. dictsort:

- `dictsort` 필터는 딕셔너리 객체를 정렬할 때 사용됩니다. 이 필터를 사용하면 딕셔너리의 키 또는 값을 기준으로 정렬된 리스트를 반환할 수 있습니다.

키를 기준으로 정렬

```
{% for key, value in my_dict|dictsort %}
  <p>{{ key }}: {{ value }}</p>
{% endfor %}
```

키를 기준으로 정렬 (역순)

```
{% for key, value in my_dict|dictsort(reverse=true) %}
  <p>{{ key }}: {{ value }}</p>
{% endfor %}
```

값을 기준으로 정렬

```
{% for key, value in my_dict|dictsort(by='value') %}
  <p>{{ key }}: {{ value }}</p>
{% endfor %}
```

값을 기준으로 정렬 (역순)

```
{% for key, value in my_dict|dictsort(by='value', reverse=true) %}
```

```
<p>{{ key }}: {{ value }}</p>
{% endfor %}
```

윈도우용 파이썬 **3.8** 설치버전

Windows x86-64 executable installer (64비트):

- [다운로드 링크](#)

Windows x86 executable installer (32비트):

- [다운로드 링크](#)

Windows x86-64 web-based installer (64비트):

- [다운로드 링크](#)

Windows x86 web-based installer (32비트):

- [다운로드 링크](#)

윈도우용 파이썬 **3.9** 설치버전

Windows x86-64 executable installer (64비트):

- [다운로드 링크](#)

Windows x86 executable installer (32비트):

- [다운로드 링크](#)


Windows x86-64 web-based installer (64비트):

- [다운로드 링크](#)

Windows x86 web-based installer (32비트):

- [다운로드 링크](#)

미니 프로젝트

 프로젝트 관련

타이타닉 데이터셋 샘플 생성 코드

```
import numpy as np
import pandas as pd
```

```

import random

np.random.seed(42)

# 승객 수
n_passengers = 100

# 기본 데이터 생성
data = {
    '생존': np.random.choice([0, 1], size=n_passengers, p=[0.6, 0.4]), #
    실제 타이타닉 생존율 반영
    '등급': np.random.choice(['1st', '2nd', '3rd'], size=n_passengers,
    p=[0.2, 0.3, 0.5]),
    '성별': np.random.choice(['male', 'female'], size=n_passengers,
    p=[0.65, 0.35]),
    '나이': np.random.normal(30, 14, n_passengers), # 평균 30세, 표준편차
    14, 표준정규분포(난수) - 종모양
    '요금': [],
    '선실코드': [],
    '탑승지코드': np.random.choice(['S', 'C', 'Q', None], size=n_passengers,
    p=[0.7, 0.2, 0.05, 0.05])
}

# 요금 생성 (등급에 따라 다른 분포)
fares = []
for pclass in data['등급']:
    if pclass == '1st':
        fare = np.random.uniform(80, 512) # 균일분포 (난수)
    elif pclass == '2nd':
        fare = np.random.uniform(20, 79)
    else:
        fare = np.random.uniform(7, 19)
    fares.append(round(fare, 4))
data['요금'] = fares

# 선실코드 생성
cabin_prefixes = ['A', 'B', 'C', 'D', 'E', 'F']
cabins = []
for _ in range(n_passengers):
    if random.random() < 0.3: # 70%는 선실코드 없음
        cabins.append(random.choice(cabin_prefixes) + str(random.randint(1,
100)))
    else:

```

```

        cabins.append(None)
data['선실코드'] = cabins

# 파생 변수 생성
data['생존여부'] = ['yes' if x == 1 else 'no' for x in data['생존']]
data['객실등급'] = ['First' if x == '1st' else 'Second' if x == '2nd' else
'Third' for x in data['등급']]
data['탑승지명'] = ['Southampton' if x == 'S' else 'Cherbourg' if x == 'C'
else 'Queenstown' if x == 'Q' else None for x in data['탑승지코드']]

# 나이 데이터에 일부 결측치 추가
age_mask = np.random.random(n_passengers) < 0.2 # 20%의 나이 데이터를
결측치로 (True, False 마스킹 데이터 생성)
data['나이'] = [age if not mask else np.nan for age, mask in
zip(data['나이'], age_mask)]

# 성인여부 컬럼 생성
data['성인여부'] = []
for age, gender in zip(data['나이'], data['성별']):
    if pd.isna(age) or age < 18:
        data['성인여부'].append('child')
    else:
        data['성인여부'].append('man' if gender == 'male' else 'woman')

# 데이터프레임 생성
df = pd.DataFrame(data)

```

데이터 분석 과정 (타이타닉 기준)

1. 기본 데이터 탐색

- 데이터 구조 확인
- 결측치 현황 파악
- 기초 통계량 확인

2. 데이터 전처리

- 결측치 처리
- 새로운 특성 생성 (가족 규모 등)

3. 기본 분석

- 생존자 수와 비율
- 성별에 따른 생존 현황
- 승객 등급별 현황

4. 시각화 분석

- 각 변수의 분포
- 생존율과 관련된 다양한 요인 분석
- 변수간 관계 시각화

5. 심화 분석

- 다변량 분석
- 상관관계 분석
- 세부적인 그룹 분석

6. 인사이트 정리

- 주요 발견사항 정리
- 통계적 수치 요약
- 의미있는 패턴 정리

errors='coerce' 옵션 사용 메소드

1. pd.to_numeric()

```
pd.to_numeric(df['column'], errors='coerce')
```

2. pd.to_datetime()

```
pd.to_datetime(df['column'], errors='coerce')
```

3. pd.to_timedelta()

```
pd.to_timedelta(df['column'], errors='coerce')
```

errors 파라미터의 옵션:

- 'coerce': 변환 불가능한 값을 NaN으로 변환

- 'raise': 오류 발생 (기본값)
- 'ignore': 변환 불가능한 값을 그대로 유지

df.dtypes

1. 숫자형
 - int64: 정수
 - float64: 실수
 - int32, float32: 32비트 버전
2. 문자형
 - object: 문자열 (파이썬의 str 객체)
 - string: 문자열 전용 dtype (pandas 1.0.0 이후)
3. 시간/날짜
 - datetime64: 날짜와 시간
 - timedelta64: 시간 간격
4. 범주형
 - category: 카테고리/범주 데이터
5. 불리언
 - bool: True/False

to_json orient 옵션

1) orient='records': 각 행을 **JSON** 객체로 변환하여 리스트 형태로 저장

```
[
  {"column1": value1, "column2": value2},
  {"column1": value3, "column2": value4}
]
```

2) orient='index': 인덱스를 키로 사용하여 **JSON** 객체를 생성

```
{
  "index1": {"column1": value1, "column2": value2},
  "index2": {"column1": value3, "column2": value4}
}
```

3) orient='columns': 각 열을 키로 사용하여 **JSON** 객체를 생성

```
{
  "column1": [value1, value3, ...],
  "column2": [value2, value4, ...]
}
```



```
"column2": [value2, value4, ...]
}
```

4) orient='values': 데이터를 리스트의 리스트로 저장

```
[
  [value1, value2],
  [value3, value4]
]
```

5) orient='table': Table Schema에 따라 데이터를 저장

```
{
  "schema": {
    "fields": [
      {"name": "column1", "type": "string"},
      {"name": "column2", "type": "number"}
    ],
    "primaryKey": ["column1"],
    "pandas_version": "0.25.0"
  },
  "data": [
    {"column1": value1, "column2": value2},
    {"column1": value3, "column2": value4}
  ]
}
```

판다스 기초

```
# 판다스 복습하기

## 시리즈(Series)
### 1) 딕셔너리로 생성
...

import pandas as pd
city = {'서울': 940, '부산': 330, '울산': 110}
df = pd.Series(city, index = ['부산', '울산'])
df
...

### 2) 리스트로 생성
...
```

```

import pandas as pd
city = ['서울', '부산', '울산']
df = pd.Series(city, index = ['x', 'y', 'z'])
df
...

## 데이터프레임(DataFrame)
### 1) 딕셔너리로 생성 (인덱스 레이블 생성)
...

import pandas as pd
city = {
    '도시': ['서울', '부산', '울산'],
    '인구': [940, 330, 110]
}
df = pd.DataFrame(city, index = ['1등', '2등', '3등'])
df
...

### 2) 리스트로 생성 (컬럼 레이블 생성)
...

import pandas as pd
data = [['김민재', 27, 75, 5428000],
        ['이강인', 22, 57, 3428000],
        ['박찬호', 50, 91, 8428000],
        ['차범근', 70, 80, 4428000],
        ['추신수', 43, 100, 4528000],
        ['손흥민', 31, 72, 7028000],
        ['황희찬', 28, 69, 2528000]]
df = pd.DataFrame(data, columns=['성명', '나이', '몸무게', '급여'])
df
...

* 추후 인덱스 레이블과 컬럼 레이블 추가 또는 수정 가능
* df.columns = ['성명', '성별', '직급', '전화번호', '주소']
* df.index = ['1등', '2등', '3등']

### 데이터 보기
* 시리즈 형태
* df['성명']
* type(df['성명'])          # pandas.core.series.Series
* 데이터프레임 형태
* df[['성명']]
* type(df[['성명']])        # pandas.core.frame.DataFrame
* 성명과 급여 컬럼을 한꺼번에
* df[['성명', '급여']]
* 여러개 컬럼은 반드시 대괄호 포함해야
* 슬라이싱 기능 사용하려면 loc, iloc를 이용해서 컬럼 위치에서 처리

```

```

* loc
* .loc[행] : 인덱스 레이블로 지정 (숫자 또는 문자, index 아님 주의)
* df.loc[0]      # 시리즈 형태
* df.loc[[0]]     # 데이터프레임 형태
* df.loc[[0,1,3]]  # 여러 행을 한꺼번에 보려면 대괄호를 포함
* df.loc[0:3] # 슬라이싱을 이용할 때는 대괄호 제거해야
* df.loc[0:3] # 0 ~ 3번 레이블까지를 의미 --> 4개의 데이터를 보여준다
(0,1,2,3)
* df.iloc[0:3] # 0 ~ 3 index까지를 의미 ---> 3개의 데이터를 보여준다
(0,1,2)
* .loc[행, 열]
* df.loc[0, '성명']      # 시리즈 형태
* df.loc[[0], ['성명']]  # 데이터프레임 형태 (복수개 사용시에는 반드시
대괄호 필요)
* df.loc[0:3, ['성명','급여']]
* df.loc[0:3, '성명':'급여'] # 슬라이싱 사용시 대괄호 반드시 제거
* .loc[조건, 열]
* df.loc[df['급여']>=5000000] # '급여' 컬럼의 데이터에서 5백만원 이상인 행
* df.loc[df['급여']>5000000, ['성명']] # '급여' 컬럼의 데이터에서 5백만원
이상인 행에 대해 '성명' 컬럼만
* 파생 데이터 생성
* df['체질량']=' '
* df.loc[df['몸무게']>80, ['체질량']] = '비만' # '몸무게' 컬럼의 데이터에서
80 이상인 행을 추출한 다음, 해당 행에 대해서만 '체질량' 컬럼에 '비만'을 입력
* iloc
* loc와 형식은 동일한데 loc달리 행과 컬럼의 index 정보로만 접근

### 데이터 삭제
* df.drop(행,열 인덱스, axis) # axis 생략시 기본값은 0
* axis=0: 세로축 (행인덱스 기준)
* axis=1: 가로축 (컬럼인덱스 기준)
* 예시) df.drop(['몸무게', '체질량'], axis=1) # 몸무게, 체질량 컬럼 한꺼번에
삭제시
* 데이터를 실제 삭제하지 않기 때문에 아래와 같이 사용해야
* df = df.drop('몸무게', axis=1)
* df.drop('몸무게', axis=1, inplace=True)
* 특정 인덱스나 컬럼을 삭제하려는 의도를 명확하게 표현
* df.drop(columns = '체질량')
* df.drop(index=[0, 1, 2])
* 조건 삭제
* df.drop(df[df['나이'] > 50].index)
* df.drop(df.index[2:4])
* df.drop(df.iloc[2:4].index)

```

판다스 문자열

```

* 문자열 추출
* .str.len(): 각 문자열의 길이를 반환한다.
* .str.split(): 문자열을 특정 구분자를 기준으로 분할한다.
* .str.get(): 문자열의 특정 위치에 있는 문자를 반환한다.
* 문자열 변환
* .str.lower(): 모든 문자열을 소문자로 변환한다.
* .str.upper(): 모든 문자열을 대문자로 변환한다.
* .str.capitalize(): 각 문자열의 첫 문자를 대문자로 변환한다.
* 문자열 처리
* .str.contains(): 특정 문자열이 포함되어 있는지 확인한다.
* .str.replace(): 특정 문자열을 다른 문자열로 대체다.
* .str.findall(): 정규 표현식과 일치하는 모든 문자열을 리스트로 반환한다.
* .str.strip(): 문자열 양 끝에 포함된 공백을 제거한다.
* .str.lstrip() : 데이터 앞에 포함된 공백을 삭제
* .str.rstrip() : 데이터 뒤에 포함된 공백을 삭제
* 실습
* df = df['직원데이터'].str.split(',', expand=True)
  * expand=True 는 각 행의 문자열을 분할한 다음, 분할된 각각의 데이터를 별도의
  컬럼으로 저장
  * 정규표현식 : 메소드 속성에 regex=True 사용시 정규식 사용 가능
  * .str.replace('^LG', '엘지', regex=True) 'LG'로 시작하는 부분을 '엘지'로
  대체한다.
  * .str.replace('대학$', '대학교', regex=True) '대학'으로 끝나는 부분을
  '대학교'로 대체한다.
  * .str.replace('부산*', '부산시', regex=True) '부산'으로 시작하는 부분을
  '부산시'로 대체한다.
  * .str.replace('부산+', '부산광역시', regex=True) '부산' 뒤에 한 글자가
  존재하는 문자를 '부산광역시'로 대체한다.
  * .str.replace('부산?', '부산광역시', regex=True) '부산' 뒤에 0 글자 또는 한
  글자가 존재하는 문자를 '부산광역시'로 대체한다.

```

NumPy의 브로드캐스팅

형상이 다른 배열 간의 연산을 가능하게 해주는 강력한 기능

```

import numpy as np

# 1. 기본적인 브로드캐스팅
# 스칼라와 배열 연산

```

```

arr = np.array([1, 2, 3, 4])
result = arr * 2 # 각 요소에 2를 곱함
print("스칼라 곱셈:", result) # [2 4 6 8]

# 2. 차원이 다른 배열 간 연산
# 1D 배열과 2D 배열 연산
arr_2d = np.array([[1, 2, 3],
                   [4, 5, 6]])
arr_1d = np.array([10, 20, 30])

# 각 행에 1D 배열 더하기
result_2d = arr_2d + arr_1d
print("\n2D 배열 브로드캐스팅:")
print(result_2d)
# [[11 22 33]
#  [14 25 36]]

# 3. 차원 확장 예제
a = np.array([1, 2, 3]) # 1D 배열
b = np.array([[1], [2], [3]]) # 2D 배열

# 각 요소의 곱셈
result_mul = a * b
print("\n차원 확장 곱셈:")
print(result_mul)
# [[1 2 3]
#  [2 4 6]
#  [3 6 9]]

```

브로드캐스팅의 주요 규칙:

1. 배열의 차원이 다른면 자동으로 차원 확장
2. 크기가 1인 차원은 다른 배열에 맞게 확장
3. 연산 시 배열의 형태가 호환되어야 함

\b 단어 경계

\b는 정규 표현식에서 단어의 시작과 끝을 나타내는 중요한 메타 문자입니다.

이를 통해 특정 단어를 정확하게 찾거나, 단어 경계를 기준으로 패턴을 설정할 수 있습니다.

특징

- 단어의 경계: \b는 알파벳 또는 숫자와 공백, 구두점 등 다른 문자의 경계를 나타냅니다. 즉, 단어의 시작 또는 끝에서 일치합니다.
- 일치하지 않는 문자: \b는 그 자체로는 어떤 문자와도 일치하지 않지만, 특정 위치를 나타냅니다.

예시

1. 단어 찾기

```
import re

text = "Python is a great programming language. I love Python!"
result = re.findall(r'\bPython\b', text)
print(result) # ['Python', 'Python']
```

- 여기서 \b는 "Python"이라는 단어가 단어 경계에서 일치하도록 합니다. 따라서 "Python"이 다른 문자와 연결되어 있지 않을 때만 일치합니다.

2. 단어 시작 찾기

```
import re

text = "Python, PHP, and Perl are programming languages."
result = re.findall(r'\bP\w+', text)
print(result) # ['Python', 'PHP']
```

- 이 예시에서는 \bP\w+를 사용하여 "P"로 시작하는 모든 단어를 찾습니다.

정규표현식 플래그

1. re.IGNORECASE 또는 re.I

- 대소문자를 구분하지 않습니다.

```
re.search(r'abc', 'AbC', re.IGNORECASE)
```

2. re.MULTILINE 또는 re.M

- ^와 \$가 각 줄의 시작과 끝을 나타냅니다.

```
re.findall(r'^abc$', 'abc\nabc', re.MULTILINE)
```

3. re.DOTALL 또는 re.S

- .가 개행 문자를 포함한 모든 문자와 일치합니다.

```
re.search(r'abc.def', 'abc\ndef', re.DOTALL)
```

4. re.VERBOSE 또는 re.X

- 주석과 공백을 허용하여 정규 표현식을 더 읽기 쉽게 만듭니다.

```
pattern = re.compile(r'''
    abc    # 알파벳 'abc'
    \d{3}  # 세 자리 숫자
''', re.VERBOSE)
```

5. re.ASCII 또는 re.A

- ASCII 문자만 일치하도록 제한합니다.

```
re.findall(r'\w+', 'eng 한글', re.ASCII)
```

정규표현식

메타 문자

정규식에서는 문자열의 패턴을 만들 때 특별한 의미로 사용하는 문자들이 있습니다. 그 문자들을 메타 문자라고 하며, 여러 메타 문자들을 조합하여 다양한 문자열의 패턴을 표현할 수 있습니다. 주로 많이 사용되는 메타 문자를 살펴보겠습니다.

메타 문자	의미
.	하나의 문자 자리수를 의미 (공백을 제외한 모든 문자 가능)
^	문자열의 시작 패턴
\$	문자열의 끝 패턴
*	바로 앞에 있는 문자가 0번 이상 반복
+	바로 앞에 있는 문자가 1번 이상 반복
?	바로 앞에 있는 문자가 있을 수도 없을 수도 있음 (0회 또는 1회)
{n}	바로 앞에 있는 문자가 n번 반복

{n, m}	바로 앞에 있는 문자가 최소 n번이상 최대 m번 이하로 반복
()	괄호 안의 문자들을 그룹으로 처리
[]	괄호 안의 문자들 중 하나와 일치
[^]	^뒤에 있는 괄호안의 문자들 제외
[-]	-앞과 뒤에 있는 문자들 사이의 문자
	앞의 문자 또는(OR) 뒤의 문자

특수문자 (백슬래시())와 영어 알파벳의 조합)

메타 문자	의미
\d	숫자 한 자리를 의미. [0-9]와 동일
\D	숫자가 아닌 문자 한 자리
\w	문자 한 자리. 알파벳+숫자+언더바(_) 중 하나의 문자. [a-zA-Z0-9_]와 동일 /LETTER 형식(유니코드)도 포함(한글도 포함)
\W	알파벳+숫자+언더바(_)가 아닌 문자 한 자리
\s	공백이나 탭 또는 줄나누기 문자 한 자리
\S	공백이나 탭, 또는 줄나누기 문자가 아닌 문자 한 자리

데이터 분석(Data Analysis)과 데이터 과학(Data Science)

서로 관련된 분야이지만, 그 초점과 접근 방식에서 차이가 있습니다.

주요 차이점

데이터 분석 (Data Analysis)

1. 목적: 데이터 분석은 주로 데이터를 수집하고 정리하여 유의미한 정보를 도출하는 데 중점을 둡니다. 주어진 데이터에서 통찰력을 얻고, 문제를 해결하거나 의사 결정을

지원하는 것이 목표입니다.

2. 기술: 데이터 분석가는 통계적 방법, 데이터 시각화 도구, **SQL**, **Excel** 등의 기술을 사용하여 데이터를 분석합니다.
3. 결과: 주로 보고서, 대시보드, 데이터 시각화 등의 형태로 결과를 제공합니다.
4. 업무 범위: 데이터 분석은 특정 질문에 대한 답을 찾거나 특정 문제를 해결하는 데 초점을 맞춥니다.

데이터 과학 (Data Science)

1. 목적: 데이터 과학은 데이터를 수집, 저장, 처리, 분석하는 것을 포함하여 복잡한 문제를 해결하고 예측 모델을 개발하는 더 넓은 범위를 포함합니다. 데이터 과학자는 데이터에서 패턴을 찾고, 미래를 예측하는 모델을 만드는 데 중점을 둡니다.
 2. 기술: 데이터 과학자는 프로그래밍 언어(**Python**, **R** 등), 기계 학습, 데이터베이스 관리, 빅데이터 기술(**Hadoop**, **Spark** 등) 등을 활용합니다.
 3. 결과: 데이터 과학의 결과는 예측 모델, 머신러닝 알고리즘, 데이터 기반의 전략적 결정 등으로 나타납니다.
 4. 업무 범위: 데이터 과학은 데이터 분석의 영역을 포함하면서도 데이터 수집부터 모델링, 배포까지의 전체 사이클을 다룹니다.
- 데이터 분석은 특정 데이터에 대한 심층적인 분석과 통찰력을 제공하는 데 중점을 두고, 데이터 과학은 더 넓은 스펙트럼에서 데이터의 활용과 예측 모델링을 포함합니다.
 - 데이터 분석가는 주로 과거 데이터를 기반으로 의사 결정을 지원하는 반면, 데이터 과학자는 미래 예측 및 패턴 인식을 통해 전략적 가치를 창출합니다.

데이터 분석 분야 직업군

1. 데이터 분석가 (**Data Analyst**)
 - 데이터를 수집, 정리, 분석하여 통찰력을 제공.
 - 보고서 작성 및 데이터 시각화 도구 사용.
2. 비즈니스 인텔리전스 분석가 (**Business Intelligence Analyst**)
 - 기업의 비즈니스 데이터 분석 및 전략적 의사결정 지원.
 - **BI** 도구를 활용하여 데이터 시각화 및 대시보드 구축.
3. 마케팅 분석가 (**Marketing Analyst**)
 - 마케팅 캠페인의 효과 분석 및 소비자 행동 연구.

- 시장 조사 및 데이터 기반의 마케팅 전략 수립.
4. 재무 분석가 (**Financial Analyst**)

- 기업의 재무 데이터 분석 및 예측.
- 투자 기회 평가 및 재무 보고서 작성.

데이터 과학 분야 직업군

1. 데이터 과학자 (**Data Scientist**)

- 데이터 수집, 분석, 모델링을 통해 비즈니스 문제 해결.
- 기계 학습 알고리즘 개발 및 예측 모델 구축.

2. 머신러닝 엔지니어 (**Machine Learning Engineer**)

- 머신러닝 모델 설계, 개발 및 배포.
- 데이터 파이프라인 구축 및 최적화.

3. 빅데이터 엔지니어 (**Big Data Engineer**)

- 대량의 데이터를 처리할 수 있는 시스템 설계 및 유지 관리.
- Hadoop, Spark 등의 빅데이터 기술 활용.

4. AI 연구원 (**AI Researcher**)

- 인공지능 관련 연구 및 새로운 알고리즘 개발.
 - 최신 AI 기술 및 트렌드 분석.
- 데이터 분석 분야는 주로 데이터의 수집과 분석을 통해 통찰력을 제공하는 역할에 중점을 둡니다.
 - 데이터 과학 분야는 더 복잡한 문제 해결을 위해 데이터 모델링 및 예측 분석을 수행하는 데 중점을 둡니다.

get_attribute 요소의 속성을 가져오는 특수한 방법

일반적인 HTML 속성

```
element.get_attribute("id")
element.get_attribute("class")
element.get_attribute("href")
element.get_attribute("value")
```

특수 속성/값

```
element.get_attribute("textContent") # 모든 텍스트 (숨김 요소 포함)
element.get_attribute("innerText")   # 보이는 텍스트만
element.get_attribute("innerHTML")   # HTML 태그 포함
element.text                          # 보이는 텍스트만 (속성이 아닌 프로퍼티)
```

Selenium의 Keys 클래스 주요 키

```
from selenium.webdriver.common.keys import Keys

# 기본 키
Keys.ENTER      # 엔터
Keys.RETURN     # 리턴
Keys.TAB        # 탭
Keys.SPACE      # 스페이스바
Keys.BACKSPACE  # 백스페이스
Keys.DELETE     # 삭제
Keys.ESCAPE     # ESC

# 화살표 키
Keys.UP         # ↑
Keys.DOWN       # ↓
Keys.LEFT       # ←
Keys.RIGHT      # →

# 기능 키
Keys.F1 ~ F12   # F1에서 F12

# 특수 키
Keys.HOME       # Home
Keys.END        # End
Keys.PAGE_UP    # Page Up
Keys.PAGE_DOWN  # Page Down
Keys.INSERT     # Insert

# 조합 키에 사용
Keys.SHIFT      # Shift
Keys.CONTROL    # Ctrl
```

```
Keys.ALT          # Alt
Keys.COMMAND      # Command (Mac)

# 조합 예시
search_box.send_keys(Keys.CONTROL + 'a') # Ctrl+A (전체선택)
search_box.send_keys(Keys.CONTROL + 'c') # Ctrl+C (복사)
search_box.send_keys(Keys.CONTROL + 'v') # Ctrl+V (붙여넣기)
```

Selenium의 ActionChains 클래스 주요 메소드

1. click(element)

- 설명: 지정된 요소를 클릭합니다.

2. double_click(element)

- 설명: 지정된 요소를 더블 클릭합니다.

3. context_click(element)

- 설명: 지정된 요소에서 마우스 오른쪽 버튼을 클릭합니다.

4. move_to_element(element)

- 설명: 지정된 요소 위로 마우스를 이동합니다.

5. move_by_offset(x_offset, y_offset)

- 설명: 현재 마우스 위치에서 특정 오프셋만큼 이동합니다.

6. drag_and_drop(source, target)

- 설명: 소스 요소를 타겟 요소로 드래그 앤 드롭합니다.

7. send_keys(*keys)

- 설명: 지정한 키를 입력합니다. 여러 키를 동시에 입력할 수 있습니다.

8. click_and_hold(source)

- 소스 요소를 클릭하고 그 상태를 유지합니다. 이 상태에서 마우스를 이동할 수 있습니다.

9. release()

- 클릭한 상태를 해제하여 드래그한 요소를 타겟에 놓습니다.

10. pause(2)

- 다음 동작을 수행하기 전에 2초 동안 대기합니다. 이 메소드는 대기 시간을 설정하여 사용자가 동작을 시각적으로 확인할 수 있도록 합니다.

11. perform()

- 설명: 지금까지 설정한 모든 작업을 실행합니다.

12. reset_actions()

- 설명: 현재 ActionChains 객체의 작업을 초기화합니다.

13. key_down(value, element=None)

- 설명: 특정 키를 눌러서 상태를 유지합니다. 선택적으로 요소를 지정할 수 있습니다.

14. key_up(value, element=None)

- 설명: 특정 키를 눌렀던 상태를 해제합니다. 선택적으로 요소를 지정할 수 있습니다.

Select 객체의 주요 메소드

`select_by_visible_text(text)`: 주어진 텍스트와 일치하는 옵션을 선택합니다.

```
select.select_by_visible_text('옵션 텍스트')
```

`select_by_value(value)`: 주어진 값과 일치하는 옵션을 선택합니다.

```
select.select_by_value('옵션값')
```

`select_by_index(index)`: 주어진 인덱스의 옵션을 선택합니다. 인덱스는 0부터 시작합니다.

```
select.select_by_index(0) # 첫 번째 옵션 선택
```

`deselect_by_visible_text(text)`: 주어진 텍스트와 일치하는 옵션을 선택 해제합니다.
(다중 선택이 가능한 경우에만 사용 가능)

```
select.deselect_by_visible_text('옵션 텍스트')
```

`deselect_by_value(value)`: 주어진 값과 일치하는 옵션을 선택 해제합니다.
(다중 선택이 가능한 경우에만 사용 가능)

```
select.deselect_by_value('옵션값')
```

`deselect_by_index(index)`: 주어진 인덱스의 옵션을 선택 해제합니다.
(다중 선택이 가능한 경우에만 사용 가능)

```
select.deselect_by_index(0) # 첫 번째 옵션 선택 해제
```

`deselect_all()`: 모든 선택된 옵션을 선택 해제합니다.
(다중 선택이 가능한 경우에만 사용 가능)

```
select.deselect_all()
```

`options`: 드롭다운의 모든 옵션을 반환합니다. 각 옵션은 `WebElement` 객체입니다.

```
all_options = select.options
```

`first_selected_option`: 현재 선택된 첫 번째 옵션을 반환합니다.

```
first_option = select.first_selected_option
```

`all_selected_options`: 현재 선택된 모든 옵션을 반환합니다.
(다중 선택이 가능한 경우)

```
selected_options = select.all_selected_options
```

window_handles

```
from selenium import webdriver
import time

# 웹 드라이버 초기화
```

```
driver = webdriver.Chrome()

# A탭에서 첫 번째 페이지 열기
driver.get("https://example.com")
time.sleep(2)

# A탭에서 링크 클릭하여 B탭 생성
driver.execute_script("window.open('https://google.com');") # B탭 생성
time.sleep(2)

# B탭 핸들 저장
dwh = driver.window_handles
print(dwh)
b_tab_handle = dwh[-1] # B탭은 마지막에 생성됨
print("B탭 핸들:", b_tab_handle)

# B탭에서 작업을 수행하기 위한 루프
while True:
    # 현재 열린 모든 창의 핸들 가져오기
    handles = driver.window_handles

    # B탭으로 전환
    if b_tab_handle in handles:
        driver.switch_to.window(b_tab_handle)
        print("B탭으로 전환됨.")

        # B탭에서 필요한 작업 수행
        # 예: 특정 요소 찾기, 클릭 등
        # driver.find_element(By.CSS_SELECTOR, 'selector').click()

    else:
        print("B탭이 닫혔습니다.")
        break

    # 잠시 대기
    time.sleep(2)

# 드라이버 종료
driver.quit()
```

Selenium find_element의 By 클래스 주요 속성

속성	방법	예시
ID	페이지 내 고유한 ID로 요소를 찾습니다.	find_element(By.ID, "login-form")
CLASS_NAME	CSS 클래스 이름으로 요소를 찾습니다.	find_element(By.CLASS_NAME, "navigation-menu")
NAME	요소의 name 속성으로 요소를 찾습니다.	find_element(By.NAME, "email")
TAG_NAME	태그 이름으로 요소를 찾습니다.	find_element(By.TAG_NAME, "a")
LINK_TEXT	a 태그 안의 링크 문자열이 전체 일치하는 요소를 찾습니다.	find_element(By.LINK_TEXT, "Click Here")
PARTIAL_LINK_TEXT	a 태그 안의 링크 문자열이 부분적으로 일치하는 요소를 찾습니다.	find_element(By.PARTIAL_LINK_TEXT, "More Info")
XPATH	XPath 표현을 사용하여 요소를 찾습니다.	find_element(By.XPATH, "//div[@class='container']")
CSS_SELECTOR	CSS 선택자를 사용하여 요소를 찾습니다.	예) find_element(By.CSS_SELECTOR, "button.submit")

셀레니엄 요소를 찾지 못할 경우

```
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException

driver = webdriver.Chrome()

try:
    element = driver.find_element(By.NAME, 'query')
except NoSuchElementException:
    print("요소를 찾을 수 없습니다.")
```


코랩용 셀레니엄 기본 세팅

```
# 필요한 패키지 설치 (코랩만 처음 세션 1회)
!apt-get update
!apt install chromium-chromedriver
!pip install selenium

# 임포트
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By

# 크롬 옵션 설정
options = webdriver.ChromeOptions()
options.add_argument('--headless')
options.add_argument('--no-sandbox')
options.add_argument('--disable-dev-shm-usage')
options.add_argument('--remote-debugging-port=9222')

# 드라이버 생성
driver = webdriver.Chrome(options=options)

# 네이버 접속 테스트
driver.get('https://www.naver.com')
print("페이지 제목:", driver.title)

# 검색창 찾기
search_box = driver.find_element(By.NAME, 'query')
search_box.send_keys('파이썬')
search_box.submit()

# 검색 결과 페이지 제목 출력
print("검색 결과 페이지 제목:", driver.title)

driver.quit()
```

CSS 선택자에서 사용할 수 있는 :pseudo-class

1. :first-child

- 설명: 부모 요소의 첫 번째 자식 요소를 선택합니다.
- 예시: `div:first-child`는 첫 번째 `div` 요소를 선택합니다.

2. :last-child

- 설명: 부모 요소의 마지막 자식 요소를 선택합니다.
- 예시: `p:last-child`는 부모 요소의 마지막 `p` 요소를 선택합니다.

3. :nth-child(n)

- 설명: 부모 요소의 n 번째 자식 요소를 선택합니다. n 은 숫자, 키워드(`even/odd`), 또는 수식($2n+1$ 등)을 사용할 수 있습니다.
- 예시: `li:nth-child(2)`는 두 번째 `li` 요소를 선택합니다.

4. :nth-of-type(n)

- 설명: 같은 타입의 자식 요소 중에서 n 번째 요소를 선택합니다.
- 예시: `div:nth-of-type(2)`는 두 번째 `div` 요소를 선택합니다.

5. :first-of-type

- 설명: 같은 타입의 자식 요소 중 첫 번째 요소를 선택합니다.
- 예시: `span:first-of-type`는 부모 요소의 첫 번째 `span` 요소를 선택합니다.

6. :last-of-type

- 설명: 같은 타입의 자식 요소 중 마지막 요소를 선택합니다.
- 예시: `li:last-of-type`는 부모 요소의 마지막 `li` 요소를 선택합니다.

7. :nth-last-child(n)

- 설명: 부모 요소의 마지막에서 n 번째 자식 요소를 선택합니다.
- 예시: `ul:nth-last-child(1)`은 마지막 자식 요소인 `ul`을 선택합니다.

8. :nth-last-of-type(n)

- 설명: 같은 타입의 자식 요소 중 마지막에서 n 번째 요소를 선택합니다.
- 예시: `p:nth-last-of-type(2)`는 마지막에서 두 번째 `p` 요소를 선택합니다.

9. :only-child

- 설명: 부모 요소의 유일한 자식 요소를 선택합니다.
- 예시: `div:only-child`는 부모 요소에 단 하나의 `div`가 있을 때 선택됩니다.

10. :only-of-type

- 설명: 같은 타입의 자식 요소 중 유일한 요소를 선택합니다.
- 예시: `span:only-of-type`은 부모 요소에 단 하나의 `span`이 있을 때 선택됩니다.

Selenium 소개

1. ChromeDriver

- Selenium은 브라우저를 자동으로 제어하기 위해 각 브라우저에 맞는 드라이버가 필요합니다. Chrome 브라우저의 경우, ChromeDriver를 사용합니다.
- ChromeDriver는 Chrome 브라우저와 Selenium 간의 통신을 담당하며, 사용자가 브라우저에서 수행하는 작업(클릭, 스크롤, 입력 등)을 제어할 수 있게 해줍니다.

2. 브라우저 인스턴스

- Selenium을 사용하여 Chrome 브라우저를 열면, 실제 사용자의 Chrome 브라우저 인스턴스가 열리게 됩니다. 이는 사용자가 브라우저를 수동으로 조작하는 것과 동일한 방식으로 작동합니다.

3. 환경 설정

- Selenium을 사용할 때, ChromeDriver의 경로를 설정하고 브라우저의 옵션(예: 헤드리스 모드, 사용자 데이터 폴더 등)을 지정할 수 있습니다. 이렇게 함으로써 Chrome의 환경을 세밀하게 조정할 수 있습니다.

4. 자바스크립트 지원

- Selenium은 Chrome 브라우저의 JavaScript 엔진을 활용하여 동적 웹 페이지를 로드하고, JavaScript로 생성된 콘텐츠를 처리할 수 있습니다.

Selenium은 사용자의 Chrome 브라우저(크롬 엔진)를 기반으로 작동하며, 이를 통해 웹 페이지를 자동으로 탐색하고 조작하며 스크래핑을 수행하는데 매우 유용한 도구입니다.

합법적인 크롤링과 불법적인 크롤링

합법적인 크롤링

- 규칙 준수:
 - **robots.txt** 파일을 확인하여 허용된 페이지만 크롤링
 - 크롤링이 허용된 사이트에서만 작업
- 과도한 요청 방지:
 - 한 번에 많은 요청을 보내 서버에 과부하를 주지 않도록 속도 제한 (**Rate Limiting**) 적용

- 데이터 활용:
크롤링한 데이터를 상업적 목적으로 사용하려면 사이트의 *이용 약관(Terms of Service)*을 준수
- 예)
데이터 수집을 위한 오픈 API 또는 허용된 공공 데이터의 크롤링

불법적인 크롤링

- 사이트의 허가 없이 대량 데이터 수집:
웹사이트 소유자가 명시적으로 금지한 데이터(로그인 영역, API 비공개 데이터 등)를 추출
- 개인 정보 침해:
사용자 계정, 이메일, 주소 등 민감한 데이터를 수집 및 저장
- 서비스 방해:
짧은 시간에 과도한 요청을 보내 사이트에 서비스 거부(DoS) 문제를 야기
- 법적 결과:
저작권 침해, 약관 위반으로 민/형사상 책임을 질 수 있음

참고 자료

- 크롤링을 시작하기 전에 반드시 robots.txt 확인
- 사용 가능한 API가 제공되는 경우, 이를 활용해 데이터 수집

robots.txt 확인 방법

robots.txt란?

- 정의:
robots.txt는 웹사이트의 루트 디렉토리에 위치한 텍스트 파일로, 크롤러가 접근할 수 있는 경로와 접근을 차단할 경로를 정의

구조 예시:

```
User-agent: *           # 모든 크롤러에게 적용
Disallow: /admin/       # /admin/ 디렉토리에 접근 금지
Allow: /public/         # /public/ 디렉토리에 접근 허용
```

```
# 모든 크롤러에게 사이트의 모든 페이지를 크롤링하도록 허용
User-agent: *
Disallow:
```

```
# 특정 크롤러에게 특정 페이지 크롤링 금지
User-agent: Googlebot
Disallow: /private/

# 특정 크롤러에게 특정 파일 형식 크롤링 금지
User-agent: *
Disallow: /*.pdf$

# 사이트의 특정 디렉토리 크롤링 금지
User-agent: *
Disallow: /tmp/
Disallow: /backup/
```

- 동작 방식:

1. 크롤러는 대상 웹사이트에 접근할 때 **/robots.txt** 파일을 먼저 확인
2. 정의된 규칙에 따라 크롤링 범위를 결정

BeautifulSoup 텍스트출력 메소드 비교 요약

특성	string	get_text()	text
용도	단일 텍스트 노드에 사용	여러 텍스트 노드를 포함한 전체 텍스트에 사용	특정 태그의 텍스트 추출
반환 값	문자열 또는 None	문자열	문자열
자식 요소 처리	자식 요소가 있으면 None 반환	모든 자식 요소의 텍스트 포함	모든 자식 요소의 텍스트 포함
구분자 설정	없음	separator 인수로 설정 가능	없음
사용 예	<p>단일 텍스트</p>	<div><p>텍스트</p><p>텍스트</p></div>	<div><p>텍스트</p><p>텍스트</p></div>

BeautifulSoup의 soup.select 메서드 사용법

선택자 유형	사용법	설명	예제
태그 선택	<code>soup.select('tag')</code>	특정 태그 이름으로 요소 선택	<code>soup.select('p')</code>
클래스 선택	<code>soup.select('.class')</code>	특정 클래스 이름을 가진 요소 선택	<code>soup.select('.content')</code>
ID 선택	<code>soup.select('#id')</code>	특정 ID를 가진 요소 선택	<code>soup.select('#main')</code>
자손 선택	<code>soup.select('parent child')</code>	특정 부모 태그의 자손 태그 선택	<code>soup.select('div p')</code>
자식 선택	<code>soup.select('parent > child')</code>	특정 부모 태그의 직접 자식 태그 선택	<code>soup.select('ul > li')</code>
속성 선택	<code>soup.select('[attr="value"]')</code>	특정 속성과 값을 가진 요소 선택	<code>soup.select('a[href="link"]')</code>
속성 포함 선택	<code>soup.select('[attr]')</code>	특정 속성이 존재하는 요소 선택	<code>soup.select('img[alt]')</code>
다중 클래스 선택	<code>soup.select('.class1.class2')</code>	여러 클래스를 동시에 가진 요소 선택	<code>soup.select('.content.active')</code>
여러 선택자	<code>soup.select('selector1, selector2')</code>	여러 선택자를 동시에 사용하여 요소 선택	<code>soup.select('p, div')</code>
속성 값 포함 선택	<code>soup.select('[attr*="value"]')</code>	속성 값에 특정 문자열이 포함된 요소 선택	<code>soup.select('a[href*="example"]')</code>
시작 문자열 선택	<code>soup.select('[attr^="value"]')</code>	속성 값이 특정 문자열로 시작하는 요소 선택	<code>soup.select('a[href^="http"]')</code>
끝 문자열 선택	<code>soup.select('[attr\$="value"]')</code>	속성 값이 특정 문자열로 끝나는 요소 선택	<code>soup.select('img[src\$=".jpg"]')</code>

BeautifulSoup find_all()의 사용 방법

사용 방법	설명
-------	----

<code>soup.find_all('태그명')</code> <code>soup.findAll('태그명') - 구버전용</code>	태그명에 해당하는 모든 태그 추출
<code>soup.find_all(속성='속성값')</code>	속성='속성값'과 일치하는 모든 태그 추출
<code>soup.find_all(attrs={'속성':'속성값'})</code>	속성='속성값'과 일치하는 모든 태그 추출
<code>soup.find_all('태그명', 속성='속성값')</code>	태그명과 일치하면서 속성='속성값'과도 일치하는 모든 태그 추출
<code>soup.find_all('태그명', attrs={'속성':'속성값'})</code>	태그명과 일치하면서 속성='속성값'과도 일치하는 모든 태그 추출

BeautifulSoup find()의 사용방법

사용 방법	설명
<code>soup.find('태그명')</code>	태그명에 해당하는 첫번째 태그 추출
<code>soup.find(속성='속성값')</code>	속성='속성값'과 일치하는 첫번째 태그 추출
<code>soup.find(attrs={'속성':'속성값'})</code>	속성='속성값'과 일치하는 첫번째 태그 추출
<code>soup.find('태그명', 속성='속성값')</code>	태그명과 일치하면서 속성='속성값'과도 일치하는 첫번째 태그 추출
<code>soup.find('태그명', attrs={'속성':'속성값'})</code>	태그명과 일치하면서 속성='속성값'과도 일치하는 첫번째 태그 추출

데이터 탐색과 데이터 추출

구분	데이터 탐색 (Data Exploration)	데이터 추출 (Data Extraction)
초점	데이터의 구조 및 특징 파악	필요한 데이터만 선택적으로 가져옴
범위	전체 데이터를 대상으로 함	특정 데이터를 대상으로 함
목적	데이터 수집 전 이해	데이터 활용 및 분석
결과물	데이터 구조 또는 개요	추출된 데이터(정제, 저장된 형태 포함)

작업 형태	탐색적, 비가공 형태로 유지	구체적, 정제된 형태로 가공
예시 도구	<code>robots.txt</code> , <code>sitemap.xml</code> 확인, 페이지 구조	HTML 태그에서 특정 클래스/ID 데이터 추출

웹 크롤링과 스크래핑의 차이

웹 크롤링 (Web Crawling)

- 정의:
웹 페이지를 자동화된 방식으로 탐색하여 URL, 페이지 콘텐츠, 메타데이터 등을 수집하는 작업.
크롤링은 구조적으로 웹사이트 전체 또는 지정된 범위의 콘텐츠를 탐색하는 데 중점을 둠.
- 주요 특징:
 - 다수의 페이지를 자동으로 방문 및 데이터 수집.
 - 주로 검색 엔진의 색인 작업에 사용.
 - 데이터의 양과 범위가 크며, 구체적인 데이터 추출은 포함하지 않을 수 있음.
- 활용 예시:
 - 검색 엔진: **Googlebot**(구글봇)은 웹사이트를 방문하고 색인을 생성하여 검색 결과를 제공.
 - 데이터 분석: 사이트 구조와 연결 관계를 분석하기 위한 연구.

웹 스크래핑 (Web Scraping)

- 정의:
웹 페이지에서 특정 데이터를 추출하는 작업.
스크래핑은 크롤링보다 더 구체적으로 데이터 추출에 초점을 맞춤.
- 주요 특징:
 - 개별 웹 페이지에서 필요한 요소 (텍스트, 이미지, 링크 등)를 추출.
 - 특정 목적에 따라 데이터를 정리하거나 가공.
 - 크롤링을 통해 방문한 웹 페이지의 데이터 추출 과정에 포함될 수 있음.
- 활용 예시:
 - 쇼핑몰 크롤링: 특정 상품의 가격, 평점, 리뷰 데이터를 수집하여 비교 분석.
 - 뉴스 사이트 크롤링: 특정 주제의 기사 헤드라인 수집.

DICTIONARY

핵심 메소드

`user[key] = value`

메소드	설명	반환값
<code>get(key[, default])</code>	키의 값을 반환, 없으면 default 반환	value 또는 default
<code>update(dict)</code>	딕셔너리 갱신/추가	None
<code>pop(key[, default])</code>	키의 값을 제거하고 반환	value 또는 default
<code>keys()</code>	모든 키를 반환	dict_keys 객체
<code>values()</code>	모든 값을 반환	dict_values 객체
<code>items()</code>	모든 (키, 값) 쌍을 반환	dict_items 객체
<code>clear()</code>	모든 항목 제거	None

TUPLE

핵심 메소드

메소드	설명	반환값
<code>count(x)</code>	튜플에서 x 의 등장 횟수를 반환	정수
<code>index(x[, start[, end]])</code>	x 의 첫 번째 등장 위치를 반환 (start, end 범위 지정 가능)	정수

튜플 주요 특징

1. 불변성(**Immutability**)
 - 한 번 생성된 후에는 수정 불가
 - 데이터 무결성 보장에 유용
 - 딕셔너리의 키로 사용 가능
2. 성능
 - 리스트보다 메모리 효율적

- 간단한 자료구조로 처리 속도가 빠름
3. 주요 용도
- 좌표, 색상값 등 관련 데이터 그룹화
 - 함수에서 여러 값 반환
 - 불변성이 필요한 데이터 저장

LIST

핵심 메소드

메소드	설명	반환값
<code>append(x)</code>	리스트 끝에 항목 <code>x</code> 를 추가	None
<code>extend(iterable)</code>	리스트 끝에 <code>iterable</code> 의 모든 항목을 추가	None
<code>insert(i, x)</code>	지정된 위치 <code>i</code> 에 항목 <code>x</code> 를 삽입	None
<code>remove(x)</code>	첫 번째로 나오는 <code>x</code> 를 제거	None
<code>pop([i])</code>	<code>i</code> 위치의 항목을 삭제하고 반환. <code>i</code> 가 없으면 마지막 항목	삭제된 항목
<code>index(x)</code>	<code>x</code> 의 첫 번째 위치를 반환 (<code>start, end</code> 범위 지정 가능)	정수
<code>count(x)</code>	<code>x</code> 가 등장하는 횟수를 반환	정수
<code>sort()</code>	리스트를 정렬 (<code>key</code> 와 <code>reverse</code> 매개변수 사용 가능)	None
<code>reverse()</code>	리스트를 역순으로 뒤집음	None

※ `len()` 함수는 문자열, 리스트, 튜플, 딕셔너리, 세트, NumPy 배열 등 다양한 데이터 구조에서 사용 가능하며, 각 데이터 구조의 요소 수를 쉽게 확인할 수 있습니다.

클래스의 핵심 개념

1. 클래스 정의: 클래스는 `class` 키워드를 사용하여 정의합니다.
2. 인스턴스: 클래스의 객체를 인스턴스라고 하며, 클래스를 통해 생성됩니다.
3. 속성: 클래스 내에서 정의된 변수로, 객체의 상태를 저장합니다.
4. 메서드: 클래스 내에서 정의된 함수로, 객체의 동작을 정의합니다.

5. 생성자: `__init__` 메서드를 통해 객체가 생성될 때 초기화 작업을 수행합니다.
6. 상속: 클래스는 다른 클래스로부터 속성과 메서드를 상속받을 수 있습니다.

객체 지향 프로그래밍(OOP)의 특징

1. 캡슐화 (Encapsulation)

- 정의: 객체의 상태(속성)와 행동(메서드)을 하나의 단위로 묶고, 외부에서 직접 접근하지 못하도록 보호하는 원칙입니다.
- 장점: 데이터 보호와 코드의 모듈화가 가능해져, 유지보수가 용이하고 오류를 줄일 수 있습니다.
- 예: 클래스 내부에 속성과 메서드를 정의하고, 외부에서 접근할 수 있는 인터페이스를 제공합니다.

2. 상속 (Inheritance)

- 정의: 기존 클래스(부모 클래스)의 속성과 메서드를 새로운 클래스(자식 클래스)가 물려받는 기능입니다.
- 장점: 코드 재사용이 가능하고, 계층적 관계를 통해 프로그램 구조를 명확히 할 수 있습니다.
- 예: **Animal** 클래스를 부모 클래스로 하고, **Dog**, **Cat** 클래스를 자식 클래스로 정의하여 공통 속성을 상속받을 수 있습니다.

3. 다형성 (Polymorphism)

- 정의: 동일한 인터페이스를 사용하여 다양한 데이터 타입의 객체를 처리할 수 있는 능력입니다.
- 장점: 코드의 유연성과 확장성을 높여줍니다. 같은 메서드 이름이지만, 각 객체에 따라 서로 다른 동작을 수행할 수 있습니다.
- 예: 메서드 오버로딩(같은 이름의 메서드가 다른 매개변수를 가짐)과 메서드 오버라이딩(부모 클래스의 메서드를 자식 클래스에서 재정의함) 등이 있습니다.

4. 추상화 (Abstraction)

- 정의: 복잡한 시스템에서 중요한 부분만을 강조하고 불필요한 세부 사항은 숨기는 과정입니다.
- 장점: 사용자는 객체의 내부 구현을 알 필요 없이 인터페이스를 통해 객체와 상호작용할 수 있습니다. 따라서 코드의 복잡도를 줄여줍니다.
- 예: 추상 클래스와 인터페이스를 사용하여 공통된 기능을 정의하고, 구체적인 구현은 자식 클래스에서 제공합니다.

SQLite timestamp 처리 관련

```

import sqlite3
from datetime import datetime
import pytz

# SQLite 데이터베이스 연결
conn = sqlite3.connect('example.db')
cursor = conn.cursor()

# 테이블 생성
cursor.execute('''
CREATE TABLE IF NOT EXISTS events (
    id INTEGER PRIMARY KEY,
    event_name TEXT NOT NULL,
    event_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)
''')

# 아시아/서울 시간대 객체 생성
seoul_tz = pytz.timezone('Asia/Seoul')

# 현재 시간 가져오기 (서울 시간)
seoul_now = datetime.now(seoul_tz)

# 서울 시간을 UTC로 변환
utc_now = seoul_now.astimezone(pytz.utc)

# 이벤트 추가 (서울시간 추가/ 만약 UTC로 저장하려면 utc_now를 넣으면 됨)
cursor.execute('INSERT INTO events (event_name, event_time) VALUES (?, ?)',
('Sample Event', seoul_now))

# 타임스탬프 가져오기
cursor.execute('SELECT event_time FROM events')
row = cursor.fetchone()

if row:
    event_time = row[0] # 데이터베이스에서 가져온 타임스탬프
    print("저장된 타임스탬프:", event_time)

# 연결 종료
conn.close()

```

SQLite 예제

```
import sqlite3
import os
import time

def clear_screen():
    # Windows
    if os.name == 'nt':
        os.system('cls')
    # Mac/Linux
    else:
        os.system('clear')

def init_db():
    conn = sqlite3.connect('todo.db')
    cursor = conn.cursor()
    cursor.execute('''
CREATE TABLE IF NOT EXISTS tasks (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    title TEXT NOT NULL,
    completed BOOLEAN DEFAULT FALSE,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
)
''')
    conn.commit()
    return conn, cursor

def add_task(cursor, conn, title):
    cursor.execute('''
INSERT INTO tasks (title, completed)
VALUES (?, ?)
''', (title, False))
    conn.commit()
    print(f'할일 "{title}" 추가됨')
    time.sleep(1)

def complete_task(cursor, conn, task_id):
    cursor.execute('''
UPDATE tasks
SET completed = TRUE
WHERE id = ?
''', (task_id,))
```

```

    if cursor.rowcount > 0:
        conn.commit()
        print(f'할일 #{task_id} 완료 처리됨')
    else:
        print(f'할일 #{task_id}를 찾을 수 없음')
    time.sleep(1)

def delete_task(cursor, conn, task_id):
    cursor.execute('''
DELETE FROM tasks
WHERE id = ?
''', (task_id,))

    if cursor.rowcount > 0:
        conn.commit()
        print(f'할일 #{task_id} 삭제됨')
    else:
        print(f'할일 #{task_id}를 찾을 수 없음')
    time.sleep(1)

def list_tasks(cursor):
    cursor.execute('''
SELECT id, title, completed
FROM tasks
ORDER BY created_at
''')
    tasks = cursor.fetchall()

    if not tasks:
        print('할일이 없습니다.')
        return

    print('\n')
    print('*'*70)
    print('현재 할일 목록:')
    for task in tasks:
        status = '✓' if task[2] else ' '
        print(f"[{status}] #{task[0]} {task[1]}")
    print('*'*70)

def display_menu():
    print("=== ToDo 관리 프로그램 ===")
    print("\n1. 할일 추가")
    print("2. 할일 완료")

```

```
print("3. 할일 삭제")
print("4. 할일 목록")
print("5. 종료")

def main():
    conn, cursor = init_db()

    while True:
        clear_screen()
        display_menu()
        list_tasks(cursor)

        try:
            choice = input("\n선택하세요 (1-5): ")

            if choice == '1':
                clear_screen()
                display_menu()
                list_tasks(cursor)
                title = input("할일 제목: ")
                add_task(cursor, conn, title)

            elif choice == '2':
                clear_screen()
                display_menu()
                list_tasks(cursor)
                task_id = int(input("완료할 할일 번호: "))
                complete_task(cursor, conn, task_id)

            elif choice == '3':
                clear_screen()
                display_menu()
                list_tasks(cursor)
                task_id = int(input("삭제할 할일 번호: "))
                delete_task(cursor, conn, task_id)

            elif choice == '4':
                clear_screen()
                display_menu()
                list_tasks(cursor)
                input("\nEnter를 누르면 계속합니다...")

            elif choice == '5':
```

```

        clear_screen()
        print("프로그램을 종료합니다.")
        conn.close()
        break

    else:
        print("잘못된 선택입니다. 다시 선택해주세요.")
        time.sleep(1)

except ValueError:
    print("올바른 숫자를 입력해주세요.")
    time.sleep(1)
except Exception as e:
    print(f"오류가 발생했습니다: {e}")
    time.sleep(1)

main()

```

SQLite 소개

SQLite는 경량의 관계형 데이터베이스 관리 시스템(RDBMS)으로, SQL을 사용하여 데이터베이스 작업을 수행합니다.

SQLite는 파일 기반 데이터베이스로, 별도의 서버 프로세스 없이 응용 프로그램 내에서 직접 데이터베이스 파일을 읽고 쓸 수 있도록 설계되었습니다.

주요 특징

1. 경량성:
 - SQLite는 매우 작은 크기로 설계되어 있으며, 최소한의 리소스만 필요합니다. 이는 모바일 앱, 소형 서버, IoT 장치 등에서 매우 유용합니다.
2. 서버리스:
 - SQLite는 클라이언트-서버 모델이 아닌, 애플리케이션이 직접 데이터베이스 파일에 접근하여 작업을 수행합니다. 이로 인해 설치 및 관리가 간편합니다.
3. 전체 파일 기반:
 - SQLite 데이터베이스는 단일 파일로 저장됩니다. 따라서 데이터베이스를 복사하거나 이동하기가 매우 쉽습니다.
4. ACID 준수:
 - SQLite는 원자성(Atomicity), 일관성(Consistency), 격리성(Isolation), 지속성(Durability)이라는 ACID 트랜잭션 특성을 지원하여 데이터 무결성을 보장합니다.
5. SQL 지원:
 - SQLite는 표준 SQL을 지원하여 데이터 정의(DDL), 데이터 조작(DML), 데이터 쿼리(DQL) 등의 작업을 수행할 수 있습니다.

파이썬 변수 전달 방식

파이썬에서 외부 변수를 함수의 파라미터로 전달하여 그 값을 수정한 경우, 원래의 변수는 변경되지 않습니다. 이는 파이썬의 변수 전달 방식 때문입니다.

파이썬에서는 기본적으로 **값 전달 방식(immutable)**과 **참조 전달 방식(mutable)**이 혼합되어 있으며, 주로 다음과 같은 방식으로 작동합니다:

1. **불변 객체 (Immutable)**: 정수(int), 문자열(str), 튜플(tuple) 등은 불변 객체로, 함수 내에서 이들 값을 수정하려고 시도하면 새로운 객체가 생성되고 원래의 객체는 변경되지 않습니다.

2. **가변 객체 (Mutable)**: 리스트(list), 딕셔너리(dict), 집합(set) 등은 가변 객체로, 참조로 전달되어 함수 내에서 직접 수정이 가능합니다.

따라서 외부 변수를 함수 내에서 수정하려면 두 가지 방법이 있습니다:

1. 함수가 수정된 값을 반환하고 기존 값을 갱신하기

```
def modify_value(x):  
    x += 10 # 값을 수정  
    return x # 수정된 값을 반환  
  
value = 5  
value = modify_value(value) # 반환된 값을 다시 할당  
print(value)  
# 출력: 15
```

2. 가변 객체를 사용하여 직접 수정하기

```
def modify_list(lst):  
    lst.append(4) # 리스트에 값을 추가  
  
my_list = [1, 2, 3]  
modify_list(my_list) # 리스트를 수정  
print(my_list)  
# 출력: [1, 2, 3, 4]
```

필요한 방식에 따라 변수를 수정할 수 있습니다.

immutable 객체는 반환된 값을 이용해 수정하고, mutable 객체는 직접 수정할 수 있습니다.

함수

장점

- 코드의 재사용성 증가
- 유지보수가 쉬워짐
- 코드의 가독성 향상
- 기능 확장이 용이

```
# 함수의 변수처리
def add(a, b):
    print(a + b)
    return a+b

num1, num2 = 5, 3
num3, num4 = 2, 5

a = add(num1, num2)
b = add(num3, num4)
c = add(7, 8)
print(f"결과: {a}, {b}, {c}")
```

datetime 모듈 설명

strftime과 strptime

strftime과 strptime은 Python의 datetime 모듈에서 날짜와 시간을 문자열로 변환하거나 문자열을 날짜 및 시간 객체로 변환하는 데 사용되는 메서드입니다. 각각의 기능과 사용법을 살펴보겠습니다.

1. strftime (String Format Time)

strftime은 **datetime** 객체를 문자열로 변환할 때 사용됩니다. 날짜와 시간을 특정 형식으로 출력할 수 있도록 포맷을 지정할 수 있습니다.

사용법

```
datetime_object.strftime(format)
```

주요 포맷 코드

- %Y: 4자리 연도 (예: 2023)
- %y: 2자리 연도 (예: 23)
- %m: 2자리 월 (01-12)
- %d: 2자리 일 (01-31)
- %H: 24시간제 시 (00-23)
- %I: 12시간제 시 (01-12)

- %M: 분 (00-59)
- %S: 초 (00-59)
- %A: 요일 이름 (예: Monday)
- %B: 월 이름 (예: January)

예제

```
from datetime import datetime

now = datetime.now()
formatted_date = now.strftime("%Y-%m-%d %H:%M:%S")
print("현재 시간:", formatted_date)
# 예: 현재 시간: 2023-10-12 14:30:59
```

2. strptime (String Parse Time)

`strptime`은 문자열을 날짜 및 시간 객체로 변환할 때 사용됩니다. 입력 문자열과 일치하는 포맷을 지정해야 합니다.

사용법

```
datetime.strptime(date_string, format)
```

예제

```
from datetime import datetime

date_string = "2023-10-12 14:30:59"
date_object = datetime.strptime(date_string, "%Y-%m-%d %H:%M:%S")
print("변환된 날짜 객체:", date_object)
# 예: 변환된 날짜 객체: 2023-10-12 14:30:59
```

timedelta 클래스

`timedelta`는 Python의 `datetime` 모듈에서 제공하는 클래스 중 하나로, 두 날짜 또는 시간 간의 차이를 표현하는 데 사용됩니다. 주로 날짜와 시간을 더하거나 빼는 작업을 수행할 때 유용합니다. `timedelta` 객체는 일수, 초, 마이크로초를 기준으로 시간 간격을 나타냅니다.

특징 및 속성

1. 생성:
 - `timedelta` 객체는 다음과 같은 인자를 통해 생성할 수 있습니다:
 - `days`: 일 수 (정수)
 - `seconds`: 초 (정수)

- microseconds: 마이크로초 (정수)
- milliseconds: 밀리초 (정수)
- minutes: 분 (정수)
- hours: 시간 (정수)
- weeks: 주 (정수)

2. 연산:

- `timedelta` 객체는 날짜와 시간에 더하거나 뺄 수 있습니다. 다음은 몇 가지 예입니다.

```
from datetime import datetime, timedelta

# 현재 날짜와 시간
now = datetime.now()
print("현재 시간:", now)

# 10일 후
future_date = now + timedelta(days=10)
print("10일 후:", future_date)

# 5시간 전
past_time = now - timedelta(hours=5)
print("5시간 전:", past_time)
```

3. 속성:

- `timedelta` 객체는 다음과 같은 속성을 제공합니다:
 - `days`: 타임델타 객체의 일 수
 - `seconds`: 타임델타 객체의 초 수 (일에 포함되지 않는)
 - `microseconds`: 타임델타 객체의 마이크로초 수
 - `total_seconds()`: 타임델타 객체를 초 단위로 변환한 값

```
delta = timedelta(days=1, hours=5, minutes=30)
print("일 수:", delta.days)
print("초 수:", delta.seconds)
print("총 초 수:", delta.total_seconds())
```

예제

사용자가 입력한 날짜에서 특정 기간만큼 더하거나 빼는 기능 구현

```
from datetime import datetime, timedelta

# 사용자로부터 날짜 입력 받기
input_date = input("날짜를 입력하세요 (YYYY-MM-DD 형식): ")
```

```

date_obj = datetime.strptime(input_date, '%Y-%m-%d')

# 10일 더하기
new_date = date_obj + timedelta(days=10)
print(f"{input_date}에서 10일 후: {new_date.strftime('%Y-%m-%d')}")

# 5일 빼기
previous_date = date_obj - timedelta(days=5)
print(f"{input_date}에서 5일 전: {previous_date.strftime('%Y-%m-%d')}")

```

time 모듈

time 모듈은 시간 관련 작업을 수행하는 데 유용한 기능을 제공합니다. 이 모듈은 시간 측정, 대기, 타이머, 시간 형식 변환 등을 위한 다양한 함수와 클래스를 포함하고 있습니다. **time** 모듈의 주요 기능을 살펴보겠습니다.

주요 기능

1. 현재 시간 얻기:
 - **time.time()**: 현재 시간을 초 단위로 반환합니다. Unix 시간(1970년 1월 1일 이후의 초) 기준입니다.
 - **time.localtime()**: 현재 시간을 구조적 시간 형식으로 반환합니다(연도, 월, 일, 시, 분, 초 등)
2. 시간 지연:
 - **time.sleep(seconds)**: 프로그램을 지정한 초만큼 일시 중지합니다. 주로 반복문에서 일정 간격으로 작업을 수행할 때 사용됩니다.

개발 기준 프로세스

1. 요구사항 분석
 - 기능목록 정리
2. 설계 문서 작성
 - 기능 목록을 기반으로 각 기능에 필요한 데이터와 함수 개요 정리
 - 이 문서는 나중에 개발의 기준이 됩니다.
3. 아키텍처 다이어그램
 - 시스템간의 관계 및 구성

4. 플로우차트 그리기

- 프로세스 흐름

5. 기능별 모듈화

- 모듈 설계: 관련된 함수와 클래스를 하나의 모듈로 묶어 관리
ex) 사용자 관련 기능은 `user.py`,
데이터 처리 기능은 `data_processor.py` 등으로 분류
- 함수 정의: 각 기능에 대한 함수를 정의
- 입력값(파라미터)과 반환값을 명확하게
- 하나에 하나의 기능만
- 문서화를 위한 주석 추가(사용법 설명)

마크 다운

<https://gist.github.com/ihoneymon/652be052a0727ad59601>

아나콘다 주피터랩 루트 경로 변경

주피터랩 설정 파일 생성:

관리자 권한으로 **Anaconda Prompt**를 열고 다음 명령어를 입력하여 주피터 설정 파일을 생성합니다:

```
jupyter lab --generate-config
```

1. 이 명령어는 기본 설정 파일을 생성합니다. 기본적으로
C:\Users\<사용자이름>\.jupyter\jupyter_lab_config.py 경로에 저장됩니다.
2. 설정 파일 열기:
파일 탐색기를 열고 위의 경로로 이동하여 jupyter_lab_config.py 파일을
찾습니다. 이 파일을 텍스트 편집기로 엽니다.

루트 경로 변경:

설정 파일에서 다음 줄의 주석을 해제하고 앞 공백지우고 원하는 경로로 수정합니다:

```
c.ServerApp.root_dir = 'C:\\path\\to\\your\\directory'
```

3. 여기서 C:\\path\\to\\your\\directory를 원하는 경로로 변경하세요.
경로 구분자는 \\ 또는 /를 사용할 수 있습니다.
4. 주피터랩 재시작:
변경사항을 저장한 후 주피터랩을 재시작합니다.

GET 방식과 POST 방식의 비교

특징	GET 방식	POST 방식
캐시화(cached)	캐시될 수 있음.	캐시되지 않음.
브라우저 히스토리	히스토리에 쿼리 문자열이 기록됨.	히스토리에 기록되지 않음.
데이터 길이	데이터의 길이가 URL 주소의 길이 이내로 제한됨. (익스플로러에서 URL 주소가 가질 수 있는 최대 길이는 2,083자이며, 이 중에서 순수 경로 길이는 2,048자까지만 허용됨)	제한 없음.
데이터 타입	오직 ASCII 문자 타입의 데이터만 전송할 수 있음.	제한 없음.

보안성	데이터가 URL 주소에 포함되어 전송되므로, 아무나 볼 수 있어 보안에 매우 취약함.	브라우저 히스토리에도 기록되지 않고, 데이터가 따로 전송되므로, GET 방식보다 보안성이 높음.
-----	---	---

.each() vs \$.each()

특징	.each()	\$.each()
반복 대상	jQuery 객체 (DOM 요소)	배열, 객체
this	현재 DOM 요소	현재 값
용도	선택된 DOM 요소들을 반복 처리	일반적인 배열이나 객체 반복 처리

.each()는 jQuery 객체에 특화된 반복 메서드이고, \$.each()는 더 일반적인 용도로 사용되는 반복 메서드입니다. 상황에 맞는 메서드를 선택하여 사용하면 됩니다. DOM 요소를 반복하는 경우에는 .each()를, 일반적인 배열이나 객체를 반복하는 경우에는 \$.each()를 사용하는 것이 좋습니다.

클래스 설정에 관한 메소드

메소드	설명
.addClass()	선택한 요소에 인수로 전달받은 클래스를 추가함.
.removeClass()	선택한 요소에서 인수로 전달받은 클래스를 제거함.
.toggleClass()	선택한 요소에 클래스가 없으면 인수로 전달받은 클래스를 추가하고, 전달받은 클래스가 추가되어 있으면 제거함.
.hasClass()	인수로 전달받은 값이 선택한 요소의 클래스 이름과 일치하는지를 확인함.

CSS 스타일 설정에 관한 메소드

메소드	설명
-----	----

<code>.css()</code>	css() 메소드는 선택한 요소 집합의 첫 번째 요소의 스타일 속성값을 반환하거나, 선택한 요소의 스타일 속성을 인수로 전달받은 값으로 설정함.
<code>.attr()</code>	선택한 요소 집합의 첫 번째 요소의 지정된 속성(attribute)값을 반환하거나, 선택한 요소의 지정된 속성을 전달받은 값으로 설정함.
<code>.prop()</code>	선택한 요소 집합의 첫 번째 요소의 지정된 프로퍼티(property)값을 반환하거나, 선택한 요소의 지정된 프로퍼티를 전달받은 값으로 설정함.
<code>.removeAttr()</code>	선택한 요소에서 지정된 속성(attribute)을 제거함.
<code>.removeProp()</code>	선택한 요소에서 지정된 프로퍼티(property)를 제거함.

필터링 메소드

메소드	설명
<code>.first()</code>	선택한 요소 중에서 첫 번째 요소를 선택함.
<code>.last()</code>	선택한 요소 중에서 마지막 요소를 선택함.
<code>.eq()</code>	선택한 요소 중에서 전달받은 인덱스에 해당하는 요소를 선택함.
<code>.filter()</code>	선택한 요소 중에서 전달받은 선택자에 해당하거나, 함수 호출의 결과가 참(true)인 요소를 모두 선택함.
<code>.not()</code>	선택한 요소 중에서 전달받은 선택자에 해당하거나, 함수 호출의 결과가 참(true)인 요소를 제외한 나머지 요소를 모두 선택함.
<code>.has()</code>	선택한 요소 중에서 전달받은 선택자에 해당하는 요소를 자손 요소로 가지고 있는 요소를 모두 선택함.
<code>.is()</code>	선택한 요소 중에서 전달받은 선택자에 해당하는 요소가 하나라도 존재하면 참(true)을 반환함.
<code>.map()</code>	선택한 요소 집합의 각 요소마다 콜백 함수를 실행하고, 그 반환값으로 구성된 제이쿼리 객체를 반환함.
<code>.slice()</code>	선택한 요소 중에서 전달받은 인덱스 범위에 해당하는 요소만을 선택함.

기타 탐색 메소드

메소드	설명
.add()	선택한 요소의 집합에 전달받은 요소를 추가함.
.each()	선택한 요소 집합의 요소마다 전달받은 콜백 함수를 실행함.
.end()	마지막으로 실행한 메소드의 실행 전 상태로 선택한 요소의 집합을 복원함.
.contents()	선택한 요소의 자식(child) 요소를 모두 선택함. (iframe)

형제 요소를 탐색하는 메소드

메소드	설명
.siblings()	선택한 요소의 형제(sibling) 요소 중에서 지정한 선택자에 해당하는 요소를 모두 선택함.
.next()	선택한 요소의 바로 다음에 위치한 형제 요소를 선택함.
.nextAll()	선택한 요소의 다음에 위치한 형제 요소를 모두 선택함.
.nextUntil()	선택한 요소의 형제 요소 중에서 지정한 선택자에 해당하는 요소 바로 이전까지의 요소를 모두 선택함.
.prev()	선택한 요소의 바로 이전에 위치한 형제 요소를 선택함.
.prevAll()	선택한 요소의 이전에 위치한 형제 요소를 모두 선택함.
.prevUntil()	선택한 요소의 형제 요소 중에서 지정한 선택자에 해당하는 요소 바로 다음까지의 요소를 모두 선택함.

자손 요소를 탐색하는 메소드

메소드	설명
.children()	선택한 요소의 자식(child) 요소를 모두 선택함.

<code>.find()</code>	선택한 요소의 자손(descendant) 요소 중에서 전달받은 선택자에 해당하는 요소를 모두 선택함.
----------------------	---

조상 요소를 탐색하는 메소드

메소드	설명
<code>.parent()</code>	선택한 요소의 부모(parent) 요소를 선택함.
<code>.parents()</code>	선택한 요소의 조상(ancestor) 요소를 모두 선택함.
<code>.parentsUntil()</code>	선택한 요소의 조상 요소 중에서 지정한 선택자에 해당하는 요소 바로 이전까지의 요소를 모두 선택함.
<code>.closest()</code>	선택한 요소를 포함한 조상 요소 중에서 지정한 선택자에 해당하는 요소 중 가장 첫 번째 요소를 선택함.

기존 요소의 외부에 새로운 콘텐츠 추가 메소드

메소드	설명
<code>.before()</code>	선택한 요소의 바로 앞에 새로운 요소나 콘텐츠를 추가함.
<code>.after()</code>	선택한 요소의 바로 뒤에 새로운 요소나 콘텐츠를 추가함.

기존 요소의 내부에 새로운 콘텐츠 추가 메소드

메소드	설명
<code>.append()</code>	선택한 요소의 마지막에 새로운 요소나 콘텐츠를 추가함.
<code>.prepend()</code>	선택한 요소의 처음에 새로운 요소나 콘텐츠를 추가함.
<code>.appendTo()</code>	선택한 요소를 해당 요소의 마지막에 삽입함.
<code>.prependTo()</code>	선택한 요소를 해당 요소의 처음에 삽입함.

.html()	해당 요소의 HTML 콘텐츠를 반환하거나 설정함.
.text()	해당 요소의 텍스트 콘텐츠를 반환하거나 설정함.

<input>요소 선택자

선택자	설명
:button	type 속성값이 "button"인 요소를 모두 선택함.
:checkbox	type 속성값이 "checkbox"인 요소를 모두 선택함.
:file	type 속성값이 "file"인 요소를 모두 선택함.
:image	type 속성값이 "image"인 요소를 모두 선택함.
:password	type 속성값이 "password"인 요소를 모두 선택함.
:radio	type 속성값이 "radio"인 요소를 모두 선택함.
:reset	type 속성값이 "reset"인 요소를 모두 선택함.
:submit	type 속성값이 "submit"인 요소를 모두 선택함.
:text	type 속성값이 "text"인 요소를 모두 선택함.
:input	<input>, <textarea>, <select>, <button>요소를 모두 선택함.
:checked	type 속성값이 "checkbox" 또는 "radio"인 요소 중에서 체크되어 있는 요소를 모두 선택함.
:selected	<option>요소 중에서 선택된 요소를 모두 선택함.
:focus	현재 포커스가 가지고 있는 요소를 선택함.
:disabled	비활성화되어있는 요소를 모두 선택함.

:enabled	활성화되어있는 요소를 모두 선택함.
----------	---------------------

필터링 선택자

선택자	설명
:eq(n)	선택한 요소 중에서 인덱스가 n인 요소를 선택함.
:gt(n)	선택한 요소 중에서 인덱스가 n보다 큰 요소를 모두 선택함.
:lt(n)	선택한 요소 중에서 인덱스가 n보다 작은 요소를 모두 선택함.
:even	선택한 요소 중에서 인덱스가 짝수인 요소를 모두 선택함.
:odd	선택한 요소 중에서 인덱스가 홀수인 요소를 모두 선택함.
:first	선택한 요소 중에서 첫 번째 요소를 선택함.
:last	선택한 요소 중에서 마지막 요소를 선택함.
:animated	선택한 요소 중에서 애니메이션 효과가 실행 중인 요소를 모두 선택함.
:header	선택한 요소 중에서 h1부터 h6까지의 요소를 모두 선택함.
:lang(언어)	선택한 요소 중에서 지정한 언어의 요소를 모두 선택함.
:not(선택자)	선택한 요소 중에서 지정한 선택자와 일치하지 않는 요소를 모두 선택함.
:root	선택한 요소 중에서 최상위 루트 요소를 선택함.
:target	선택한 요소 중에서 웹 페이지 URI의 fragment 식별자와 일치하는 요소를 모두 선택함.
:contains(텍스트)	선택한 요소 중에서 지정한 텍스트를 포함하는 요소를 모두 선택함.

:has(선택자)	선택한 요소 중에서 지정한 선택자와 일치하는 자손 요소를 갖는 요소를 모두 선택함.
:empty	선택한 요소 중에서 자식 요소를 가지고 있지 않은 요소를 모두 선택함.
:parent	선택한 요소 중에서 자식 요소를 가지고 있는 요소를 모두 선택함.

제이쿼리(jQuery)란?

제이쿼리(jQuery)는 오픈 소스 기반의 자바스크립트 라이브러리입니다.

제이쿼리는 여러분의 웹 사이트에 자바스크립트를 더욱 손쉽게 활용할 수 있게 해줍니다. 또한, 제이쿼리를 사용하면 짧고 단순한 코드로도 웹 페이지에 다양한 효과나 연출을 적용할 수 있습니다.

이러한 제이쿼리는 오늘날 가장 인기 있는 자바스크립트 라이브러리 중 하나입니다.

제이쿼리의 역사

제이쿼리는 2006년 미국의 존 레식(John Resig)이 뉴욕시 바캠프(Barcamp)에서 처음으로 소개하였습니다.

현재는 jQuery Team이라는 개발자 그룹이 jQuery Foundation을 통해 개발과 유지 보수를 담당하고 있습니다.

제이쿼리의 장점

제이쿼리가 많이 사용되는 이유는 다음과 같습니다.

1. 제이쿼리는 주요 웹 브라우저의 구버전을 포함한 대부분의 브라우저에서 지원됩니다.
2. HTML DOM을 손쉽게 조작할 수 있으며, CSS 스타일도 간단히 적용할 수 있습니다.
3. 애니메이션 효과나 대화형 처리를 간단하게 적용해 줍니다.
4. 같은 동작을 하는 프로그램을 더욱 짧은 코드로 구현할 수 있습니다.
5. 다양한 플러그인과 참고할 수 있는 문서가 많이 존재합니다.
6. 오픈 라이선스를 적용하여 누구나 자유롭게 사용할 수 있습니다.

[Chart.js](#) ⇒ 무료 자바스크립트 차트 라이브러리

HTML 태그 속성 접근

1. 일반 속성:

모든 HTML 요소에는 다양한 속성이 있으며, 이를 JavaScript를 통해 접근할 수 있습니다.

```
<a id="myLink" href="https://example.com" target="_blank">링크</a>
```

```
const link = document.getElementById('myLink');  
console.log(link.href); // 'https://example.com'  
console.log(link.target); // '_blank'
```

2. 입력 요소:

```
<input id="myInput" type="text" value="기본값" placeholder="입력하세요">
```

```
const input = document.getElementById('myInput');  
console.log(input.value); // '기본값'  
console.log(input.placeholder); // '입력하세요'
```

3. 스타일 속성:

CSS 스타일 속성도 JavaScript를 통해 접근할 수 있습니다.

```
<div id="myDiv" style="color: red; width: 200px;">안녕하세요</div>
```

```
const div = document.getElementById('myDiv');  
console.log(div.style.color); // 'red'  
div.style.color = 'blue'; // 색상 변경
```

addEventListener 메서드의 이벤트 타입

마우스 이벤트

- **click**: 요소를 클릭할 때 발생
- **dblclick**: 요소를 더블 클릭할 때 발생
- **mouseover**: 마우스가 요소 위로 올라갈 때 발생
- **mouseout**: 마우스가 요소에서 나갈 때 발생

키보드 이벤트

- **keydown**: 키를 누를 때 발생
- **keyup**: 키를 떼를 때 발생

폼 이벤트

- **submit**: 폼이 제출될 때 발생
- **change**: 폼 요소의 값이 변경될 때 발생
- **input**: 입력 값이 변경될 때 발생

문서/윈도우 이벤트

- **load**: 페이지나 이미지가 로드될 때 발생
- **resize**: 창 크기가 변경될 때 발생
- **scroll**: 사용자가 스크롤할 때 발생

웹 요소 접근

- **innerText**
 - 화면에 표시되는 텍스트만 포함
 - 스타일 적용된 텍스트만 반환
- **textContent**
 - 모든 텍스트를 포함
 - HTML 태그는 제외되며, 모든 텍스트 반환
- **innerHTML**
 - 요소의 HTML 내용을 포함
 - HTML 태그와 내용을 모두 반환하며 HTML 구조 유지

querySelector와 querySelectorAll

DOM에서 요소를 선택하는 데 사용되는 메서드로, CSS 선택자를 사용하여 HTML 요소를 쉽게 찾을 수 있게 해준다.

querySelector

주어진 CSS 선택자와 일치하는 첫 번째 요소를 반환합니다.

```
const element = document.querySelector('.class-name');
```

- 선택된 요소가 없으면 `null`을 반환합니다.
- CSS 선택자를 사용할 수 있어 매우 유연합니다.
- 단일 요소만 반환하므로, 여러 요소를 선택할 필요가 있을 때는 다른 방법을 사용해야 합니다.

querySelectorAll

주어진 CSS 선택자와 일치하는 모든 요소를 반환합니다.

```
const elements = document.querySelectorAll('.class-name');
```

- 반환값은 `NodeList`입니다.
- `NodeList`는 배열과 유사한 객체로, 여러 요소를 포함합니다.
- 선택된 요소가 없으면 빈 `NodeList`를 반환합니다.
- `NodeList`는 `forEach` 메서드를 사용할 수 있어 반복 처리에 용이합니다.

논리 **OR** 연산자를 이용한 표현식

```
let b = 0; // falsy
let c = 5;

let a = b || c; // a는 5가 됨
console.log(a); // 5

b = 10; // truthy
a = b || c; // a는 10이 됨
console.log(a); // 10
```

classList.add와 className 의 차이

```
classList.add('highlight');
```

`highlight` 클래스를 현재 클래스 목록에 추가합니다.
이미 `highlight` 클래스가 존재하는 경우 중복하여 추가하지 않습니다.
여러 클래스를 추가할 수 있습니다

- `myDiv.classList.add('class1', 'class2');`

className = "highlight":

className 속성에 "highlight" 문자열을 직접 할당합니다.

이 경우, 기존의 모든 클래스명이 제거되고 highlight 클래스만 남습니다.

즉, 요소가 가지고 있던 모든 기존 클래스를 잃게 됩니다.

classList 메서드

add(className):

기능: highlight 클래스를 myDiv에 추가합니다.

버튼 클릭 시 해당 클래스가 추가되고 알림이 표시됩니다.

remove(className):

기능: highlight 클래스를 myDiv에서 제거합니다.

버튼 클릭 시 해당 클래스가 제거되고 알림이 표시됩니다.

toggle(className):

기능: hidden 클래스를 myDiv에 추가하거나 제거합니다.

버튼 클릭 시 클래스가 토글되고 그에 대한 알림이 표시됩니다.

contains(className):

기능: myDiv에 highlight 클래스가 있는지 확인합니다.

버튼 클릭 시 클래스가 존재하는지 여부에 따라 알림이 표시됩니다.

localStorage

localStorage는 웹 스토리지 API의 일부로, 웹 애플리케이션이 브라우저에 데이터를 저장할 수 있도록 해줍니다. 이 데이터는 브라우저를 닫거나 새로 고침해도 유지됩니다.

- 데이터 저장
 - localStorage.setItem(key, value)
- 데이터 조회
 - localStorage.getItem(key)
- 데이터 삭제
 - localStorage.removeItem(key)
- 모든 데이터 삭제
 - localStorage.clear()

Location 객체

1. location.href

현재 페이지의 **URL**을 가져오거나 새로운 **URL**로 변경하는 데 사용됩니다.

- 읽기: 현재 페이지의 **URL**을 반환합니다.
- 쓰기: 다른 **URL**로 페이지를 이동합니다.

2. location.reload()

현재 페이지를 새로 고침합니다.

기본적으로 캐시된 페이지를 새로 고침하지만, 매개변수로 **true**를 전달하면 서버에서 새로운 페이지를 요청합니다.

3. location.replace()

현재 페이지를 새로운 **URL**로 변경하지만, 브라우저의 세션 기록에 현재 페이지를 남기지 않습니다. 즉, 사용자가 "뒤로" 버튼을 눌러도 이전 페이지로 돌아갈 수 없습니다.

window 객체의 주요 속성과 메서드

구분		설명
속성	innerWidth	웹 브라우저 화면의 너비를 px(픽셀) 단위로 나타냅니다.
	innerHeight	웹 브라우저 화면의 높이를 px 단위로 나타냅니다.
	outerWidth	웹 브라우저 창의 너비를 px 단위로 나타냅니다.
	outerHeight	웹 브라우저 창의 높이를 px 단위로 나타냅니다.
	screenY	웹 브라우저 위쪽 면과 모니터의 간격을 px 단위로 나타냅니다.
	screenX	웹 브라우저 왼쪽 면과 모니터의 간격을 px 단위로 나타냅니다.
	scrollX	웹 브라우저의 수평 스크롤 위치를 px 단위로 나타냅니다.
	scrollY	웹 브라우저의 수직 스크롤 위치를 px 단위로 나타냅니다.
메서드	alert()	알림창을 표시합니다.
	confirm()	확인창을 표시합니다.

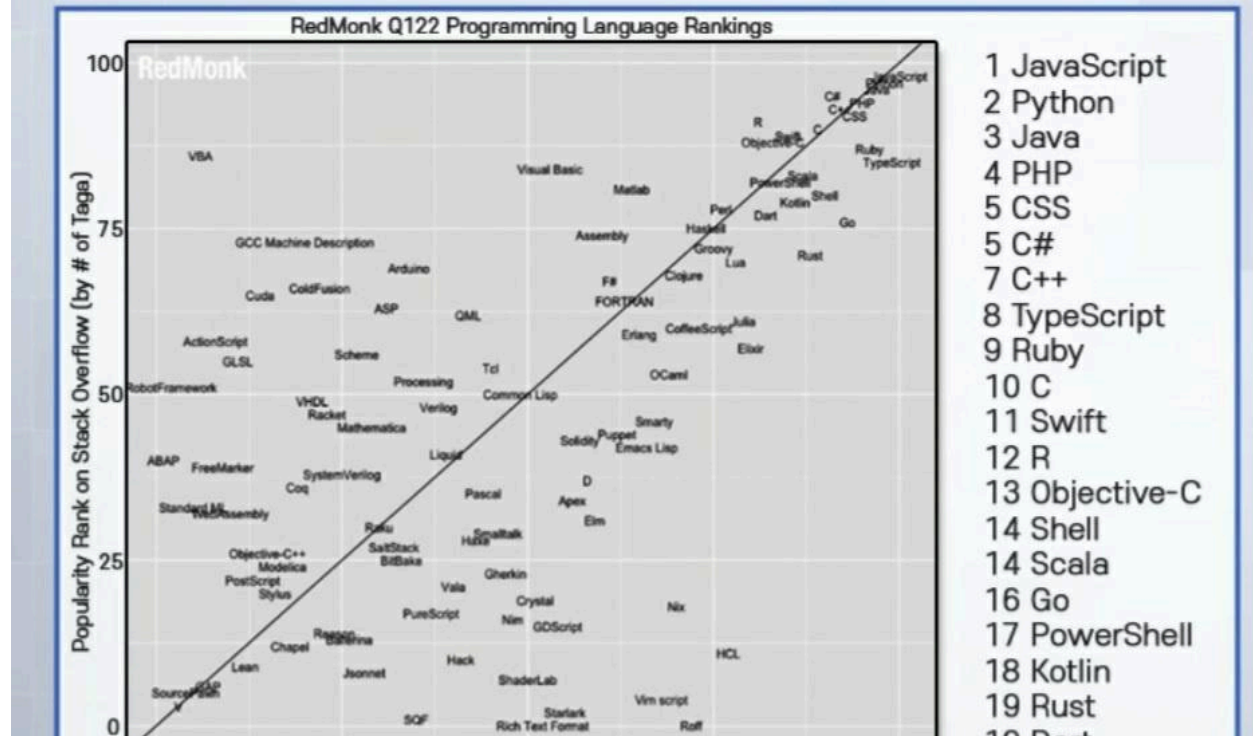
	prompt()	입력창을 표시합니다.
	open()	새로운 웹 브라우저 창을 엽니다.
	close()	웹 브라우저 창을 닫습니다.
	setTimeout()	일정 시간(ms) 뒤에 콜백 함수를 한 번만 실행합니다.
	setInterval()	일정 시간(ms)마다 콜백 함수를 반복적으로 실행합니다.
	clearInterval	setInterval() 메서드로 반복 실행되는 함수를 중지합니다.
	scrollTo()	웹 브라우저의 스크롤을 특정 위치만큼 이동합니다.
	scrollBy()	웹 브라우저의 스크롤을 현재 위치에서 상대적 위치로 이동합니다.

브라우저 객체 모델의 종류

종류	설명
window	웹 브라우저가 열릴 때마다 생성되는 최상위 관리 객체
document	웹 브라우저에 표시되는 HTML 문서 정보가 포함된 객체
location	웹 브라우저에 현재 표시된 페이지에 대한 URL 정보가 포함된 객체
history	웹 브라우저에 저장된 방문 기록이 포함된 객체
navigator	웹 브라우저 정보가 포함된 객체
screen	웹 브라우저의 화면 정보가 포함된 객체

인기언어 순위 (참고)

RedMonk 사가 발표한 2022 상위 20개 언어 인기 순위



참여기업

☰ 프로젝트 참여 기업정보 (2024.10 기준)

Math 객체의 주요 메서드

종류	설명
Math.floor()	주어진 숫자와 같거나 작은 정수 중에서 가장 큰 수를 반환합니다(내림).
Math.ceil()	주어진 숫자와 같거나 큰 정수 중에서 가장 작은 수를 반환합니다(올림).
Math.round()	주어진 숫자를 반올림한 수와 가장 가까운 정수를 반환합니다(반올림).
Math.random()	0 이상 1 미만의 난수를 반환합니다.

Date 객체의 메서드 (new Date())

종류	설명
getFullYear() setFullYear()	연도를 4자리 숫자로 표시합니다.
getMonth() setMonth()	월을 0부터 11까지의 숫자로 표시합니다(1월 → 0, 12월 → 11).
getDate() setDate()	일을 1부터 31까지의 숫자로 표시합니다.
getDay()	요일을 0부터 6까지의 숫자로 표시합니다(일요일 → 0, 토요일 → 6).
getTime()/setTime()	1970년 1월 1일 12:00 이후의 시간을 밀리초(1/1000초) 단위로 표시합니다.
getHours() setHours()	시를 0부터 23까지의 숫자로 표시합니다.
getMinutes() setMinutes()	분을 0부터 59까지의 숫자로 표시합니다.
getSeconds() setSeconds()	초를 0부터 59까지의 숫자로 표시합니다.
getMilliseconds() setMilliseconds()	밀리초를 0부터 999까지의 숫자로 표시합니다.

Array 객체의 주요 속성과 메서드

구분		설명
속성	length	배열의 요소 개수를 반환합니다.
파괴적 메서드	push()	배열의 맨 뒤에 데이터를 추가합니다.
	pop()	배열의 맨 뒤에서 데이터를 추출합니다.
	unshift()	배열의 맨 앞에 데이터를 추가합니다.
	shift()	배열의 맨 앞에서 데이터를 추출합니다.
	sort()	배열의 요소를 정렬합니다.
	reverse()	배열의 요소를 역순으로 정렬합니다.
비파괴적 메서드	forEach()	배열의 요소를 하나씩 순회하면서 요소마다 콜백(callback) 함수를 호출합니다.

	filter()	배열의 요소를 하나씩 순회하면서 요소마다 콜백 함수를 호출해 true 를 반환하는 요소만 추출합니다. 추출한 요소로 새로운 배열을 만들고 만들어진 배열을 반환합니다.
비파괴적 메서드	find()	배열의 요소를 탐색하면서 주어진 판별 함수를 만족하는 첫 번째 값을 반환합니다.
	findIndex()	값 대신 인덱스 숫자를 반환한다는 것만 빼면 find() 메서드와 같습니다.
	includes()	배열에 특정 값이 포함되어 있는지 확인해서 포함됐으면 true , 아니면 false 를 반환합니다.
	join()	배열의 모든 요소를 주어진 구분자로 합칩니다.

string 객체의 주요 속성과 메서드

구분		설명
속성	length	문자열의 길이를 반환합니다.
메서드	includes()	메서드의 매개변수에 인자로 전달되는 문자열이 대상 문자열에 포함되어 있으면 true , 아니면 false 를 반환합니다.
	replace()	대상 문자열에서 메서드의 매개변수에 인자로 전달되는 문자열과 일치하는 한 부분을 찾아서 다른 데이터로 변경한 새로운 문자열을 반환합니다.
	replaceAll()	대상 문자열에서 메서드의 매개변수에 인자로 전달되는 문자열과 일치하는 모든 부분을 찾아서 다른 데이터로 변경한 새로운 문자열을 반환합니다.
	split()	메서드의 매개변수에 인자로 전달되는 구분자를 이용해 대상 문자열을 여러 개의 문자열로 분리하고, 분리한 문자열을 새로운 배열로 반환합니다.
	toUpperCase()	대상 문자열을 대문자로 변경해 반환합니다.
	trim()	대상 문자열의 앞뒤 공백을 제거한 값을 반환합니다.
	indexOf()	대상 문자열과 일치하는 첫 번째 문자의 인덱스를 반환합니다.

자바스크립트 개발 서비스 예제

(개인 포트폴리오용 서비스 주제: 배열 또는 객체 활용)

1. 전자상거래 플랫폼: 상품 목록, 장바구니 관리, 주문 처리.
2. 개인 금융 관리 앱: 수입/지출 기록, 예산 관리, 금융 목표 설정.
3. 프로젝트 관리 도구: 작업 목록, 팀원 관리, 진척도 추적.
4. 소셜 미디어 플랫폼: 사용자 프로필, 게시물 관리, 댓글 기능.
5. 온라인 교육 플랫폼: 강의 목록, 수업 진행 관리, 퀴즈 기능.
6. 뉴스 애그리게이터: 기사 수집, 카테고리 분류, 북마크 기능.
7. 레시피 공유 사이트: 레시피 목록, 재료 검색, 사용자 리뷰.
8. 여행 계획 앱: 여행 일정 관리, 장소 추천, 예산 계산.
9. 건강 관리 앱: 운동 기록, 식단 관리, 건강 목표 설정.
10. 이벤트 관리 플랫폼: 행사 생성, 참석자 관리, 일정 알림.
11. 블로그 플랫폼: 게시물 작성, 태그 관리, 댓글 기능.
12. 채팅 애플리케이션: 사용자 관리, 메시지 전송, 알림 기능.
13. 투표 시스템: 설문 생성, 응답 수집, 결과 분석.
14. 포트폴리오 웹사이트: 프로젝트 목록, 기술 스택, 연락처 양식.
15. 유튜브 클론: 동영상 업로드, 댓글 시스템, 추천 알고리즘.
16. 부동산 관리 플랫폼: 매물 등록, 검색 기능, 문의 관리.
17. 기부 플랫폼: 프로젝트 생성, 기부 관리, 후원자 관리.
18. 가계부 앱: 수입/지출 기록, 카테고리 분석, 보고서 생성.
19. 날씨 정보 앱: 지역별 날씨 조회, 알림 기능, 예보 그래프.
20. 쇼핑 목록 관리 앱: 품목 추가/삭제, 카테고리 분류, 공유 기능.
21. 스케줄 관리 앱: 일정 추가, 알림 기능, 반복 일정 설정.
22. 할 일 목록 앱: 작업 추가, 마감일 설정, 우선순위 지정.
23. 커뮤니티 포럼: 주제별 게시판, 사용자 관리, 신고 기능.
24. 버전 관리 시스템: 파일 버전 추적, 변경 이력 관리, 롤백 기능.
25. **QR** 코드 생성기: URL 입력, **QR** 코드 생성, 다운로드 기능.
26. 채용 플랫폼: 구인 공고 작성, 지원자 관리, 면접 일정 조율.
27. 비디오 회의 플랫폼: 회의 생성, 참가자 초대, 녹화 기능.
28. 디지털 달력: 이벤트 추가, 반복 일정 관리, 공유 기능.
29. 음악 스트리밍 서비스: 플레이리스트 관리, 추천 기능, 사용자 리뷰.
30. 사진 공유 앱: 사진 업로드, 앨범 관리, 댓글 및 좋아요 기능.

수업자료 (공유폴더)

<https://drive.google.com/drive/folders/1LGUxG1upp12jsvtN4DUK5EtbliwCeTJe?usp=sharing>

자바스크립트 배열 메소드

1. push()

배열의 끝에 요소를 추가합니다.


```
let fruits = ['apple', 'banana'];
fruits.push('cherry'); // ['apple', 'banana', 'cherry']
console.log(fruits);
```

2. pop()

배열의 마지막 요소를 제거하고 반환합니다.

```
let numbers = [1, 2, 3];
let lastNumber = numbers.pop(); // lastNumber는 3
console.log(numbers); // [1, 2]
```

3. shift()

배열의 첫 번째 요소를 제거하고 반환합니다.

```
let colors = ['red', 'green', 'blue'];
let firstColor = colors.shift(); // firstColor는 'red'
console.log(colors); // ['green', 'blue']
```

4. unshift()

배열의 시작 부분에 요소를 추가합니다.

```
let animals = ['dog', 'cat'];
animals.unshift('rabbit'); // ['rabbit', 'dog', 'cat']
console.log(animals);
```

5. splice()

배열의 특정 위치에서 요소를 추가하거나 제거합니다.

```
let numbers = [1, 2, 3, 4, 5];
numbers.splice(2, 1, 10); // 인덱스 2에서 1개 요소를 제거하고 10을 추가
console.log(numbers); // [1, 2, 10, 4, 5]
```

6. slice()

배열의 특정 부분을 잘라내어 새로운 배열을 반환합니다.

```
let fruits = ['apple', 'banana', 'cherry', 'date'];
let citrus = fruits.slice(1, 3); // 인덱스 1부터 3까지 잘라냄
console.log(citrus); // ['banana', 'cherry']
```

7. forEach()

배열의 각 요소에 대해 제공된 함수를 실행합니다.

```
let numbers = [1, 2, 3];
numbers.forEach(num => {
  console.log(num * 2); // 2, 4, 6 출력
});
```

```
let numbers = [1, 2, 3];
numbers.forEach(function(num) {
  console.log(num * 2); // 2, 4, 6 출력
});
```

8. map()

각 요소에 대해 주어진 함수를 호출하고, 그 결과로 새로운 배열을 생성합니다.

```
let numbers = [1, 2, 3];
let doubled = numbers.map(num => num * 2); // [2, 4, 6]
console.log(doubled);
```

9. filter()

주어진 조건을 만족하는 요소들로 새로운 배열을 생성합니다.

```
let numbers = [1, 2, 3, 4, 5];
let evenNumbers = numbers.filter(num => num % 2 === 0); // [2, 4]
console.log(evenNumbers);
```

10. reduce()

배열을 단일 값으로 축약하는 데 사용됩니다.

```
let numbers = [1, 2, 3, 4];
let sum = numbers.reduce((acc, num) => acc + num, 0); // 10
console.log(sum);
```

자바스크립트 **Array**

자바스크립트 **Array**는 자바스크립트에서 배열을 생성하고 조작하는 데 사용되는 내장 객체입니다. 배열은 여러 개의 데이터를 순차적으로 저장할 수 있는 데이터 구조로, 각 데이터는 인덱스를 통해 접근할 수 있습니다.

주요 특징

1. 동적 크기: 자바스크립트의 배열은 크기가 고정되어 있지 않으며, 필요에 따라 요소를 추가하거나 삭제할 수 있습니다.
2. 다양한 데이터 유형: 배열은 숫자, 문자열, 객체 등 다양한 데이터 유형을 혼합하여 저장할 수 있습니다.
3. 0부터 시작하는 인덱스: 배열의 인덱스는 0부터 시작합니다. 첫 번째 요소는 인덱스 0에 위치합니다.

배열 생성 방법

리터럴 표기법

```
const fruits = ['apple', 'banana', 'cherry'];
```

Array 생성자:

```
const numbers = new Array(1, 2, 3, 4, 5);
```

주요 메서드

- **push():** 배열의 끝에 요소를 추가합니다.
- **pop():** 배열의 마지막 요소를 제거하고 반환합니다.
- **shift():** 배열의 첫 번째 요소를 제거하고 반환합니다.
- **unshift():** 배열의 시작 부분에 요소를 추가합니다.
- **splice():** 배열의 특정 위치에서 요소를 추가하거나 제거합니다.
- **slice():** 배열의 특정 부분을 잘라내어 새로운 배열을 반환합니다.
- **forEach():** 배열의 각 요소에 대해 제공된 함수를 실행합니다.
- **map():** 각 요소에 대해 주어진 함수를 호출하고, 그 결과로 새로운 배열을 생성합니다.
- **filter():** 주어진 조건을 만족하는 요소들로 새로운 배열을 생성합니다.
- **reduce():** 배열을 단일 값으로 축약하는 데 사용됩니다.

예시

```
const numbers = [1, 2, 3, 4, 5];

// 배열에 요소 추가
numbers.push(6); // [1, 2, 3, 4, 5, 6]

// 배열에서 요소 제거
const last = numbers.pop(); // last는 6, numbers는 [1, 2, 3, 4, 5]
```

```
// 배열을 순회
numbers.forEach(num => {
  console.log(num); // 1, 2, 3, 4, 5 출력
});

// 평균 계산
const average = numbers.reduce((acc, num) => acc + num, 0) /
numbers.length;
console.log(average); // 3 출력
```

Array 객체는 자바스크립트에서 데이터를 다루는 데 매우 유용하며, 다양한 메서드를 통해 효율적으로 데이터를 처리할 수 있게 해줍니다.

CSS 셀렉터

- 태그: `p { margin:0}`
- 클래스: `.classname { margin:0}`
- 아이디: `#uniqueid { margin:0}`
- 그룹화 셀렉터
`h1, h2, h3 { margin: 0; }`
- 자식 셀렉터
`div p {margin:0}`
`div > li {margin}`
`div > .classname {margin}`
- 형제 셀렉터
`h1 + p { margin-top: 20px; }` - 인접한 형제
`h1 ~ p { color: green; }` - 먼형제
- 속성 셀렉터
`input[type="text"] { border: 1px solid black; }`
- 가상클래스 셀렉터
`a:hover { text-decoration: underline; }`
`li:first-child { font-weight: bold; }`

- 가상요소 셀렉터

```
div::before { content: "▶ "; }
```

```
div::after { content: " ♥♥♥"; }
```

```
div::before {
    content: url('path/to/your/image.png');
    display: inline-block;
    margin-right: 5px;
}
```

OT

- 학습자료 추천
 - <https://www.w3schools.com>
- 추천 자격증
 - SQLD
 - 경영정보시각화능력
- 무료 학점은행
 - KMOOC
 - 특허청: 지식재산학과
 - 기상청: 대기과학과
 - 기술교육대학교: 컴퓨터공학과, 기계공학과, 메트로닉스학과
- 무료 및 저렴 웹호스팅
 - 카페24: 500원 부터 시작
 - 닷홈: 무료 → 빌더로만 사용해서 웹호스팅 부적합
 - 아이비로: 무료
- 컴퓨터 프로그램 언어 특성
 - 데이터 유형
 - 배열: 리스트, 튜플, 딕셔너리
 - 제어
 - 반복
 - 입출력
 - 함수
 - 클래스
 - 파일
 - 데이터베이스
- AI풀스택
 - 기초역량: 파이썬, MySQL, 아두이노
 - 웹개발: HTML5, CSS3, JavaScript, jQuery, Flask
 - 빅데이터 분석: 데이터분석/시각화, 머신러닝/딥러닝, 자연어처리, 이미지인식

- 서비스 인프라 구축: Linux, 클라우드, 가상화, Git, Docker
- 디스코드
 - <https://discord.gg/5VKKJxAq>
- AIOT 빅데이터 과정 사전 설문
 - [설문지](#)
- 공유문서 공유를 위한 이메일 정보

강사: 이요섭 (s3458600@gmail.com)

순번	이름	이메일
1	이지엽	olzlduq1@gmail.com
2	김혜민	tulipsfairy@gmail.com
3	허지원	dnjsdl1008@gmail.com
4	안기부	cowasdiy1234@naver.com
5	황수용	ghkdtndyd122@naver.com
6	황병천	hbc1347@gmail.com
7	박선정	psj0691@naver.com psj2790@gmail.com
8	임건우	kw10171@gmail.com
9	임정은	wjddms3416@naver.com
10	임승준	atlasjun@gmail.com
11	손유승	youseung5153@naver.com
12	진예찬	jinstartup@gmail.com
13	김동휘	kimccmpiano@gmail.com
14	신준혁	junhyuk000@naver.com
15	정유진	uzzin822@gmail.com
16	주정은	joojeun1026@naver.com
17	손유빈	dbqls6637@gmail.com