

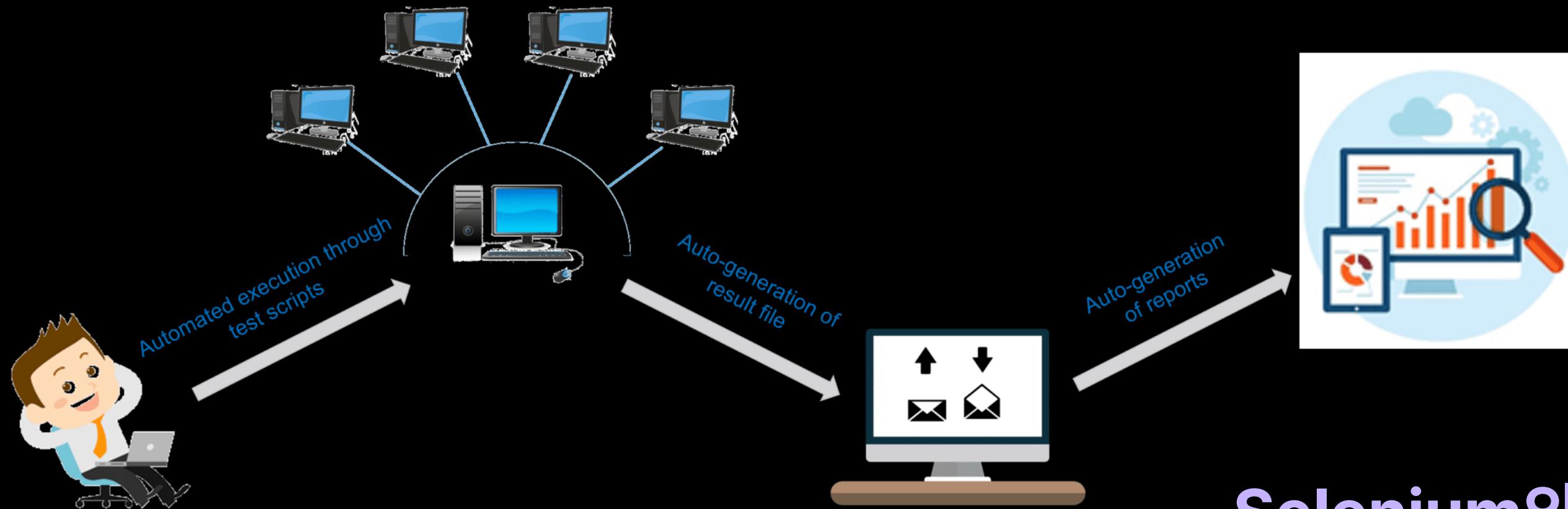
웹페이지 자동 로그인 및 데이터 크롤링



수업내용

- Selenium을 활용하여 로그인 자동화의 기본 개념과 보안 문제를 이해하고, 쿠키와 세션을 이용해 로그인 상태를 유지하는 방법을 학습합니다.
- 문자열 가공 및 데이터 정제 기술을 익혀 크롤링한 데이터를 깔끔하게 정리합니다.
- Selenium과 json 모듈을 활용해 웹사이트에서 수집한 데이터를 JSON 형식으로 저장하고 불러오는 자동화 스크립트를 작성합니다.
- 에러 처리 및 디버깅 기법을 통해 자동화 스크립트의 안정성과 신뢰성을 높입니다.

Selenium으로 어디까지 자동화될까?



Selenium의 확장 기능과 실무 활용

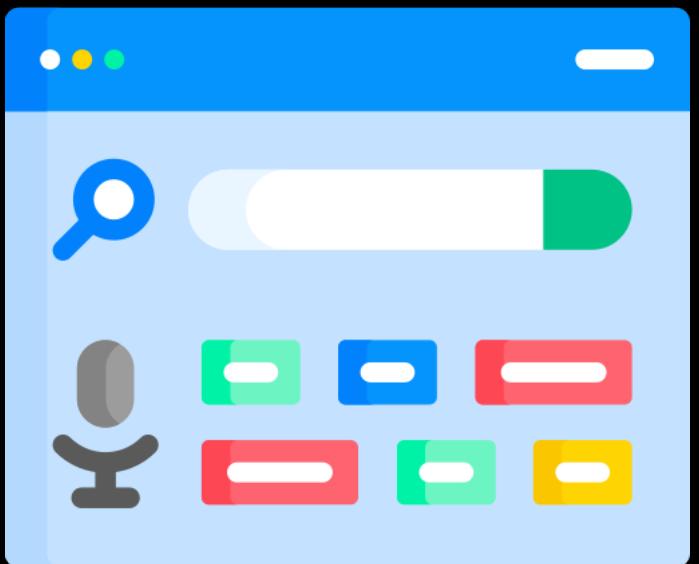
- 반복 작업 자동화 (예: 자동 로그인, 데이터 수집)
- 테스트 자동화 (예: UI 테스트, 기능 검증)

오늘 배울 심화 내용

- 쿠키/세션을 활용한 로그인 유지
- 데이터 크롤링 및 JSON 저장
- 에러 처리 및 디버깅

자동화는 같은 자동화인데, 뭐가 다를까?

- 웹 자동화: 효율성을 높이기 위한 작업 자동화
- 웹 테스트 자동화: 애플리케이션의 품질을 보장하기 위한 검증

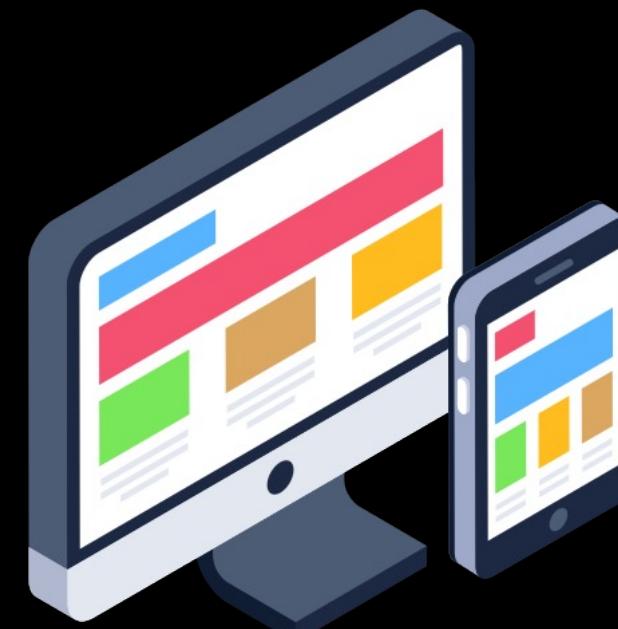


웹 자동화

- 목적: 반복 작업 자동화, 효율성 향상
- 예시: 자동 로그인, 데이터 크롤링, 웹페이지 갱신 확인
- 방식: 사용자의 행동을 흉내 내며 작업 수행

웹 테스트 자동화

- 목적: 웹 애플리케이션의 정상 작동 확인
- 예시: UI 테스트, 기능 검증, Regression Test
- 방식: 테스트 시나리오 작성, Assertion 사용



Selenium의 한계는 어디까지일까?



1. 쿠키와 세션 관리

- 쿠키 저장 및 불러오기
- 세션 유지 및 재인증
- 보안 이슈 처리



2. 문자열 가공 및 데이터 정제

- 문자열 추출 및 정제
- 정규표현식 활용
- HTML 태그 제거 및 텍스트만 추출

3. JSON 데이터 처리 및 저장

- JSON 데이터 파싱
- JSON 파일 저장 및 불러오기
- 다양한 데이터 포맷 지원: CSV, JSON, XML



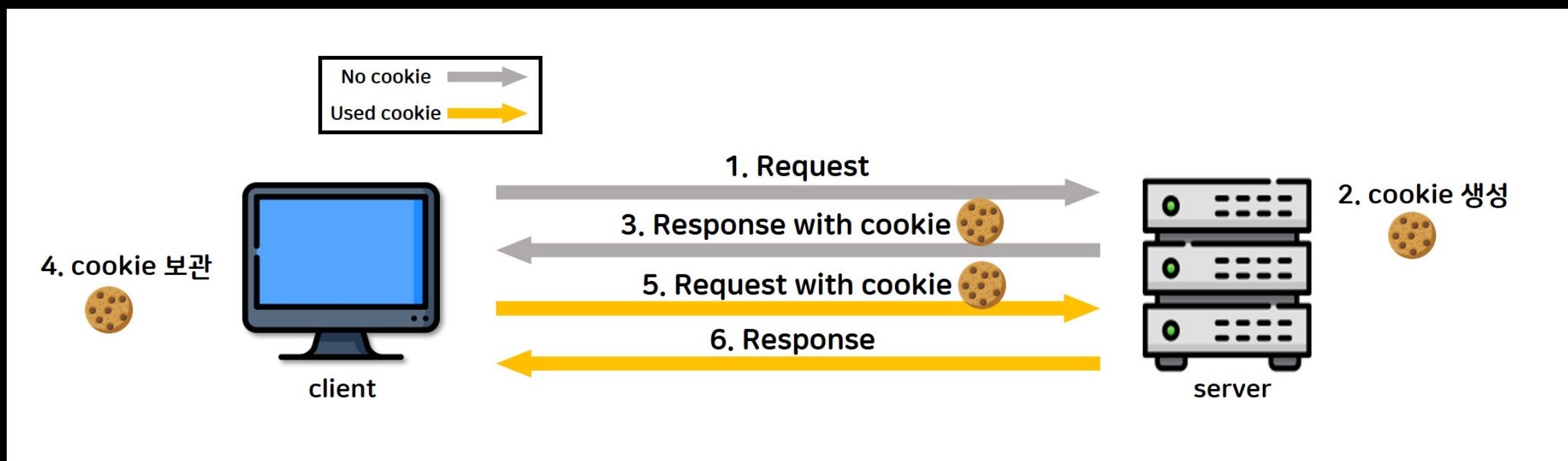
4. 에러 처리 및 디버깅

- 예외 처리
- 디버깅
- 안정성 향상

쿠키가 대체 뭐길래 자동 로그인이 될까?

쿠키(Cookie)

- 정의: 웹사이트가 사용자의 브라우저에 저장하는 작은 데이터 조각
- 역할: 로그인 상태 유지, 사용자 맞춤형 설정 저장, 장바구니 유지
- 종류:
 - 세션 쿠키: 브라우저 종료 시 삭제 (임시 로그인)
 - 영구 쿠키: 만료 기간 설정, 브라우저 종료 후에도 유지 (자동 로그인)



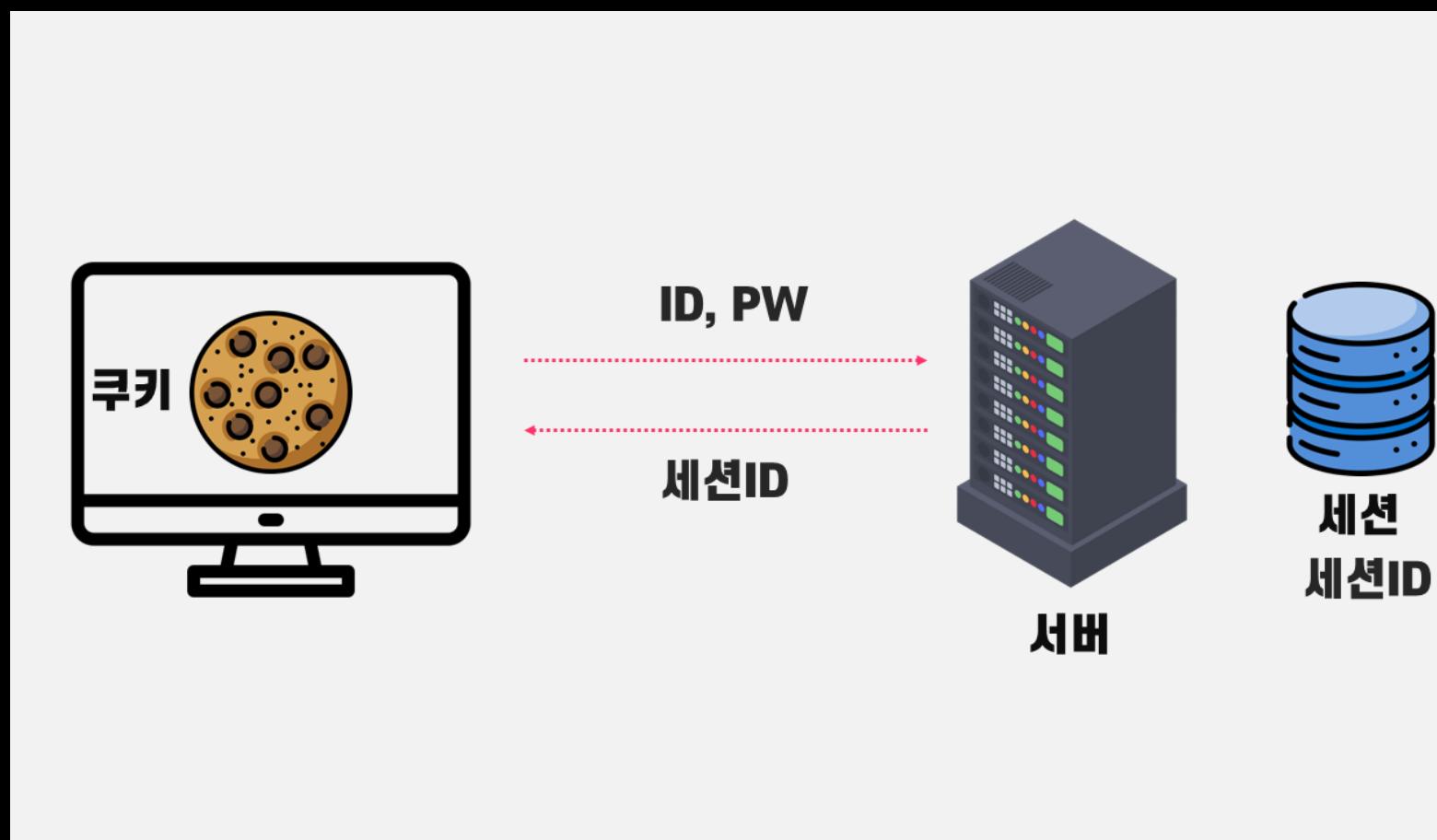
작동 원리

- 로그인 시 쿠키 생성: 서버가 브라우저에 쿠키 저장
- 재방문 시 쿠키 확인: 브라우저가 쿠키를 서버에 전달 → 로그인 상태 유지

세션은 서버가 기억하는 나만의 공간

세션(Session)

- 정의: 서버가 사용자의 상태를 저장하는 임시 공간
- 역할: 로그인 상태 유지, 장바구니 저장, 사용자 정보 관리
- 특징:
 - 서버에 저장됨 (보안성 높음)
 - 세션 ID를 통해 사용자 식별
 - 브라우저 닫거나 일정 시간 후 만료



작동 원리

- 로그인 시 세션 생성: 서버가 세션 ID 생성 후 저장
- 세션 ID 발급: 브라우저에 세션 ID 쿠키 저장
- 요청 시 세션 확인: 서버가 세션 ID를 확인해 로그인 상태 유지



쿠키와 세션, 무엇이 다를까?

[쿠키(Cookie)와 세션(Session)의 차이]

구분	쿠키 (Cookie)	세션 (Session)
저장 위치	사용자의 브라우저	서버 (세션 ID만 브라우저에 저장)
보안 수준	낮음 (클라이언트 저장)	높음 (서버 저장)
유지 기간	설정된 만료 시간까지 유지 (영구 쿠키 가능)	브라우저 종료 시 또는 일정 시간 후 자동 만료
용도	자동 로그인, 사용자 맞춤 설정	로그인 상태 유지, 장바구니 정보 저장
데이터 저장 가능 여부	브라우저에서 직접 읽고 수정 가능	서버에서만 관리하며 브라우저에서 직접 접근 불가
속도	빠름 (클라이언트에서 바로 처리)	비교적 느림 (서버 요청 필요)
취약점	XSS(Cross-Site Scripting) 공격에 취약	세션 고정(Session Fixation) 공격 가능
보안 강화 방법	HTTPOnly, Secure, SameSite 속성 설정	세션 타임아웃, Regenerate Session ID 활용

Selenium으로 쿠키를 훔쳐보자(?)

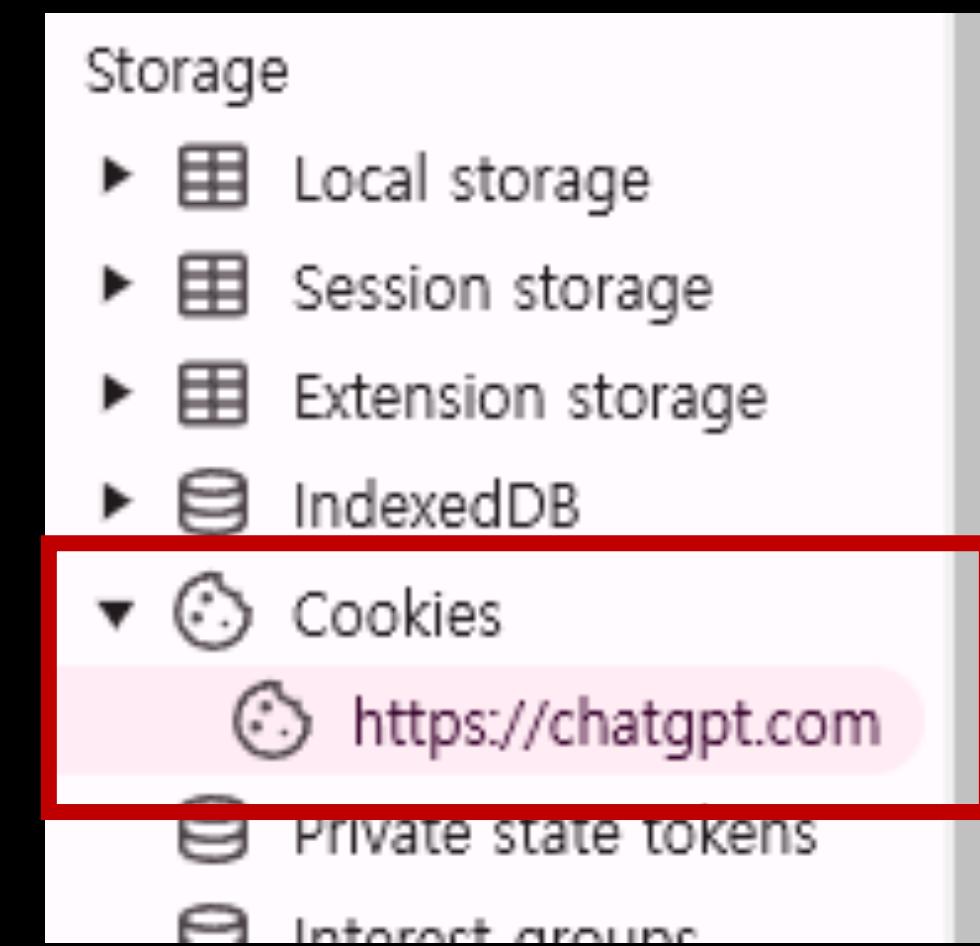
Selenium에서 쿠키 가져오는 방법

- `get_cookies()`: 현재 페이지의 모든 쿠키를 리스트 형태로 가져오기
- `get_cookie(name)`: 특정 이름의 쿠키만 가져오기

사용 예시

- 자동 로그인 상태 확인: 로그인 후 저장된 쿠키 확인
- 세션 유지 확인: 세션 쿠키가 만료되었는지 확인

```
driver.get("https://example.com")
cookies = driver.get_cookies()
for cookie in cookies:
    print(cookie)
```





쿠키를 직접 만들어볼 수 있다고?

Selenium에서 쿠키 추가하기

- `add_cookie(cookie_dict)`: 브라우저에 새로운 쿠키 추가
- 사용 예시: 로그인 없이 자동 로그인 구현

```
driver.get("https://example.com")
driver.add_cookie({"name": "user_session", "value": "abc123", "domain": "example.com"})
```

Selenium에서 쿠키 삭제하기

- `delete_cookie(name)`: 특정 쿠키 삭제
- `delete_all_cookies()`: 모든 쿠키 삭제

```
driver.delete_cookie("user_session") # 특정 쿠키 삭제
driver.delete_all_cookies() # 모든 쿠키 삭제
```

로그인 버튼 없이 로그인하는 마법!?

쿠키를 활용한 자동 로그인 개념

- 웹사이트는 로그인 후 쿠키를 저장하여 로그인 상태 유지
- Selenium에서 쿠키를 저장 → 재사용하면 로그인 버튼 없이 로그인 가능

자동 로그인 구현 흐름

- 사용자가 로그인 → 브라우저에서 로그인 쿠키 생성
- 쿠키를 가져와 저장 (`get_cookies()`)
- 브라우저를 다시 열고 쿠키 추가 (`add_cookie()`)
- 로그인 없이 로그인된 상태 유지

```
from selenium import webdriver  
  
driver = webdriver.Chrome()  
driver.get("https://example.com/login")  
  
# 1. 로그인 후 쿠키 가져오기  
cookies = driver.get_cookies()  
driver.quit() # 브라우저 종료  
  
# 2. 새 브라우저에서 쿠키 추가하여 자동 로그인  
driver = webdriver.Chrome()  
driver.get("https://example.com")  
for cookie in cookies:  
    driver.add_cookie(cookie)  
  
driver.refresh() # 페이지 새로고침 → 자동 로그인 완료!
```



실습1 : 로그인하고 쿠키 받아가세요~

실습 목표

- Selenium을 활용하여 웹사이트 로그인 후 쿠키 정보를 저장하는 과정 학습
- 쿠키를 활용하여 브라우저를 닫아도 로그인 상태를 유지하는 방법 이해
- Selenium의 get_cookies()와 json.dump()를 활용하여 쿠키를 JSON 파일로 저장하는 자동화 코드 구현
- 개발자 도구(F12)를 활용하여 쿠키가 정상적으로 저장되었는지 검증하는 방법 학습

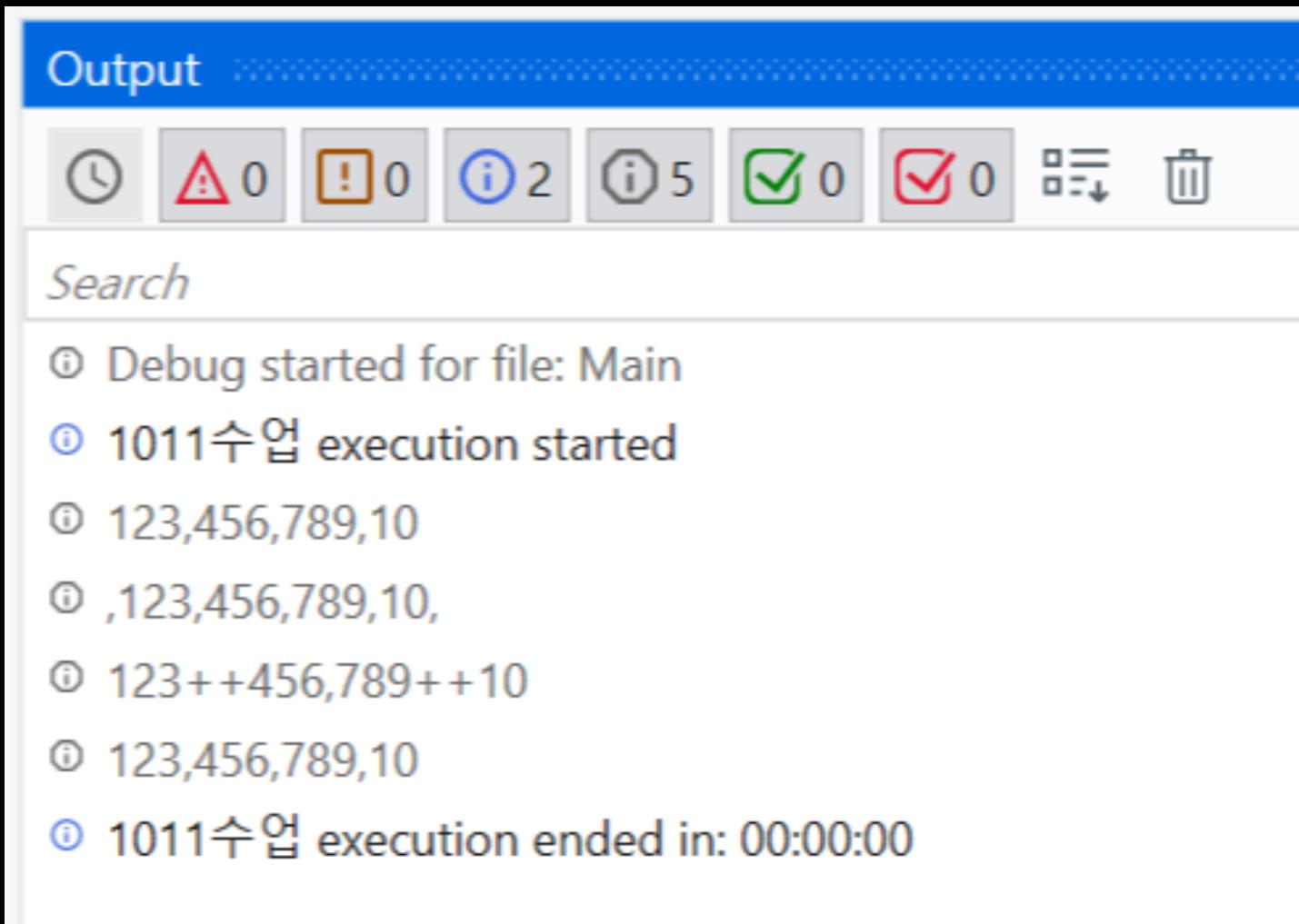
실습 내용

1. 애완동물 커뮤니티에 로그인 후 쿠키 저장 (get_cookies())
- 아이디(user123), 비밀번호(password)를 입력 후 로그인 버튼 클릭
2. 저장된 쿠키를 JSON 파일로 저장 (json.dump())
- 가져온 쿠키 데이터를 json.dump()를 활용해 cookies.json 파일로 저장 후
콘솔창으로 저장 완료 시 문구 확인

QA에게도 문자열 가공이 필요한 이유?

QA에게 문자열 가공이 필요한 이유

- 테스트 데이터 검증: 크롤링한 데이터나 테스트 결과의 공백, 특수문자 제거
- 예상 결과와의 비교: 통화 기호, 날짜 형식 등 포맷 차이 해결
- 에러 메시지 분석: 에러 로그에서 핵심 원인만 추출하여 문제 해결



예시

- 가격 검증: "\$1,299" vs. "1,299원"
→ \$, ,, 원 제거 후 숫자로 비교
- 문자열 비교: Hello vs. Hello
→ strip()으로 공백 제거 후 비교
- 특수문자 처리: New&Sale!
→ New Sale으로 정제

☰ 문자열 다듬기! 공백, 특수문자 제거하기

문자열 추출 및 정제

: 크롤링한 데이터에서 필요한 정보만 추출하고 불필요한 공백, 특수문자 제거

주요 메서드 및 사용법

- `strip()`: 문자열 양쪽 공백 제거

```
# ✅ strip() 예제: 문자열 양쪽 공백 제거
text = "Hello World!"
print(text.strip()) # 출력: "Hello World!"
```

- `replace(old, new)`: 특정 문자열 바꾸기

```
# ✅ replace() 예제: 특정 문자열 바꾸기
message = "Hello, World!"
print(message.replace("Hello", "Hi")) # 출력: "Hi, World!"
```

- `split(delimiter)`: 구분자를 기준으로 문자열 나누기

```
# ✅ split() 예제: 구분자를 기준으로 문자열 나누기
fruits = "apple, banana, cherry"
print(fruits.split(", ")) # 출력: ['apple', 'banana', 'cherry']
```



필요한 데이터만 정확히 골라내는 정규표현식

정규표현식(Regex)

: 특정 패턴을 가진 문자열을 찾아내는 강력한 도구

QA에서 자주 활용되는 이유

- 이메일, 전화번호, 날짜, 금액 등 패턴이 있는 데이터 검증
- 에러 메시지에서 특정 코드나 번호 추출
- HTML에서 특정 태그만 필터링

자주 사용되는 정규표현식 패턴

- 숫자만 찾기: \d+
- 이메일 찾기: \b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b
- 전화번호 찾기: \d{2,3}-\d{3,4}-\d{4}
- HTML 태그 제거: <.*?>

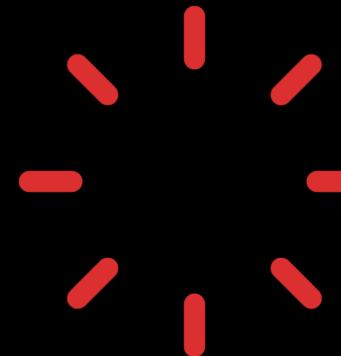
Character classes	
.	any character except newline
\w \d \s	word, digit, whitespace
\W \D \S	not word, digit, whitespace
[abc]	any of a, b, or c
[^abc]	not a, b, or c
[a-g]	character between a & g
Anchors	
^abc\$	start / end of the string
\b \B	word, not-word boundary
Escaped characters	
\. * \\	escaped special characters
\t \n \r	tab, linefeed, carriage return
Groups & Lookaround	
(abc)	capture group
\1	backreference to group #1
(?:abc)	non-capturing group
(?=abc)	positive lookahead
(?!abc)	negative lookahead
Quantifiers & Alternation	
a*	0 or more, 1 or more, 0 or 1
a{5}	exactly five, two or more
a{1,3}	between one & three
a+? a{2,}?	match as few as possible
ab cd	match ab or cd



공백 하나 때문에 테스트 실패?

공백 제거가 중요한 이유

- 예상 결과와의 불일치 문제:
"Hello " vs. "Hello" (공백 때문에 다르게 인식)
- HTML 소스 코드의 불필요한 공백: 크롤링한 데이터에 포함된 불필요한 공백 제거 필요

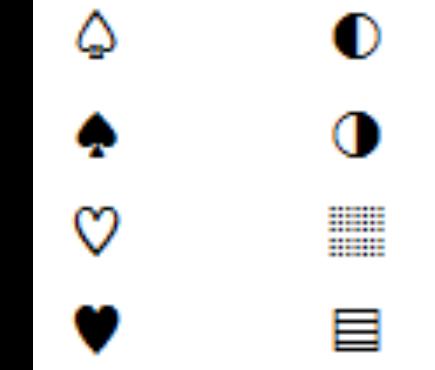


주요 공백 제거 메서드

- strip(): 문자열 양쪽 공백 제거
- lstrip(): 문자열 왼쪽 공백 제거
- rstrip(): 문자열 오른쪽 공백 제거

특수문자 처리의 필요성

- 문자열 비교 시 불일치 문제:
"가격: \$1,299" vs. "1299"
- 숫자로 변환하기 위해 특수문자 제거 필요



주요 특수문자 제거 메서드

- replace(old, new): 특정 문자열 또는 특수문자를 새 문자열로 대체
- 정규표현식 활용 (re.sub())
 - 숫자만 남기기: re.sub(r"\D", "", text)



가격 비교, 통계 분석? 숫자 데이터 정제하자

숫자 및 통화 데이터 정제가 필요한 이유

- 숫자 비교 및 계산: 크롤링한 데이터에 포함된 통화 기호(\$, 원), 쉼표(,) 제거
- 일관된 포맷 유지: 날짜, 가격, 통계 데이터를 일관된 형식으로 변환
- QA 테스트에서 가격 및 수량 검증

주요 메서드 및 사용법

- `re.sub()`: 특수문자 제거 및 숫자만 남기기

```
import re
text = "$1,299원"
clean_text = re.sub(r"[^0-9]", "", text)
print(clean_text) # 1299
```

- `re.findall()`: 문자열에서 숫자만 추출

```
text = "총 금액: 1,299원, 할인: 200원"
numbers = re.findall(r"\d+", text)
print(numbers) # ['1299', '200']
```

- `int()` 및 `float()` 변환

- 문자열을 정수(`int()`) 또는 실수(`float()`)로 변환하여 계산
- 예: "1299" → `int("1299")` → 1299

Selenium으로 수집한 데이터의 특징

- HTML 태그 포함: <div>, 등 불필요한 HTML 태그 포함
- 공백 및 특수문자 포함: 줄 바꿈, 공백, 특수문자가 포함된 경우가 많음
- 형식 불일치: 날짜, 가격, 통화 기호 등 일관되지 않은 형식

Selenium에서 수집한 데이터 정제 방법

- HTML 태그 제거: `re.sub(r"<.*?>", "", text)`
- 공백 및 특수문자 제거: `strip()`, `replace()` 사용
- 숫자만 추출: `re.sub(r"[^0-9]", "", text)`
- JSON 포맷으로 저장: `json.dump()` 사용



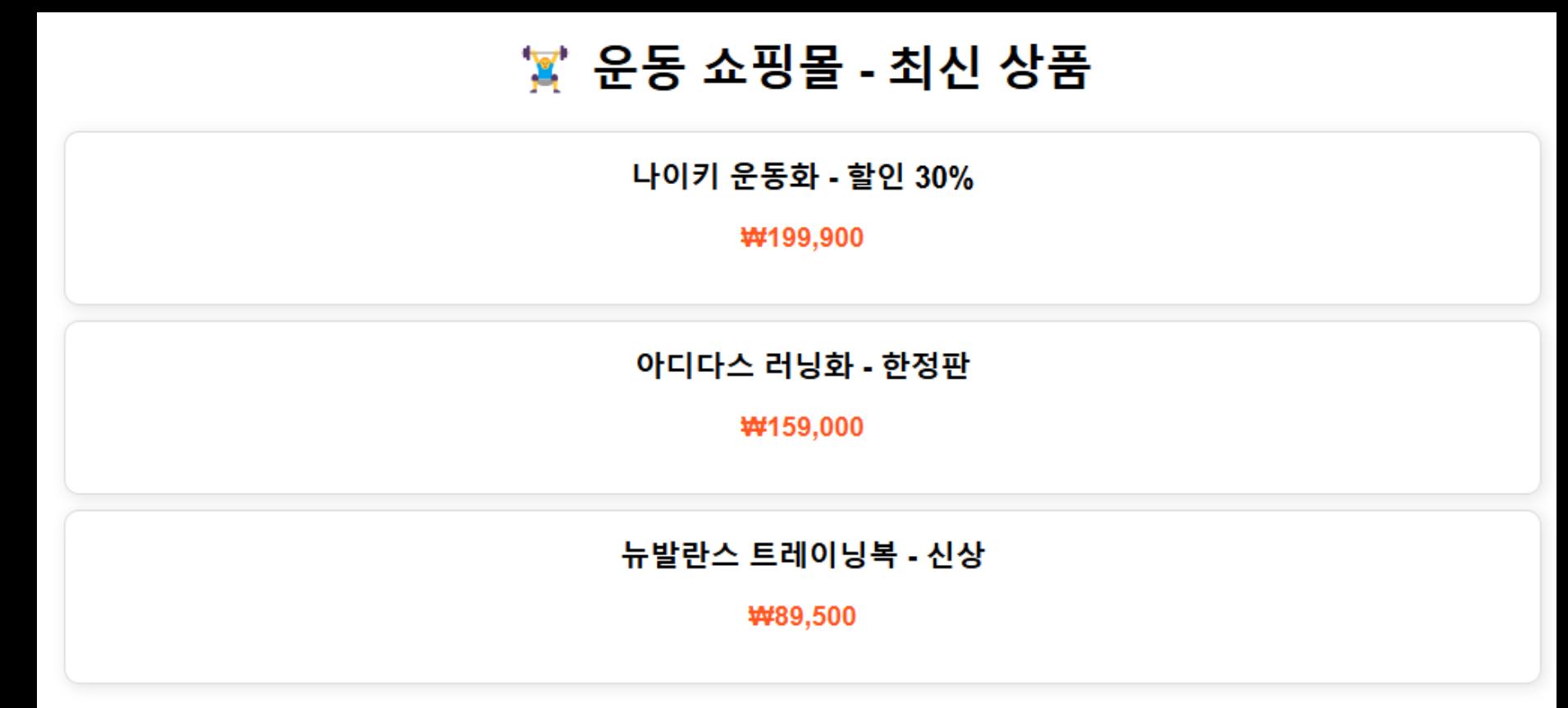
실습2 : 운동 쇼핑몰 문자열 정제 실습

실습 목표

- Selenium을 활용하여 운동 쇼핑몰의 상품명과 가격 데이터를 크롤링
- 문자열 가공 함수 (strip(), replace(), split())를 사용하여 데이터 정제
- 정제된 데이터를 JSON 파일로 저장하여 활용하는 방법 학습

실습 과정

1. Selenium으로 HTML 페이지에서 상품명과 가격 크롤링
2. 문자열 가공 함수(strip(), replace(), split())를 활용해 데이터 정리
3. 가격 데이터를 정수형으로 변환 (int())
4. json.dump()를 활용해 **sports_products.json** 파일로 저장 후
콘솔창으로 저장 완료 시 문구 확인





퀴즈 타임



QUIZ 1번부터 8번을 풀어봐요!



JSON, 데이터를 저장하는 가장 쉬운 방법

JSON

- JavaScript Object Notation (JSON): 데이터를 가볍고 쉽게 저장하고 전송하는 형식
- Key-Value 구조를 가진 텍스트 기반 데이터 형식
- 프로그래밍 언어와 독립적이지만, Python, JavaScript, Java 등에서 쉽게 사용 가능

JSON의 특징

- 가볍고 구조화된 데이터 저장 방식
- 사람도 읽기 쉬운 텍스트 형식
- 키(Key)와 값(Value)으로 이루어진 데이터 구조
- 웹 API, 데이터 전송, 파일 저장 등에 사용

```
{  
    "DocumentType": 1,  
    "No.": "S-ORD101001",  
    "SellToCustNo": "10000",  
    "PostingDate": "2023-04-02",  
    "Lines": [  
        {  
            "LineNo": 10000,  
            "Type": 2,  
            "No": "1996-S",  
            "Quantity": 12,  
            "UnitPrice": 1397.3  
        },  
        {  
            "LineNo": 20000,  
            "Type": 2,  
            "No": "1900-S",  
            "Quantity": 4,  
            "UnitPrice": 192.8  
        }  
    ]  
}
```

JSON과 Python, 닮았지만 다르다?

JSON과 Python 딕셔너리는 비슷하지만 다르다!

- 공통점: 둘 다 Key-Value 구조를 가진다.
- 차이점:
 - JSON은 문자열 형식, 딕셔너리는 Python 객체
 - JSON의 키(Key)는 문자열(str)만 가능, 딕셔너리는 숫자, 튜플도 가능
 - JSON에서는 true, false, null을 사용하지만, Python에서는 True, False, None

JSON 데이터 (문자열 형식)

```
{  
    "name": "홍길동",  
    "age": 25,  
    "city": "서울"  
}
```

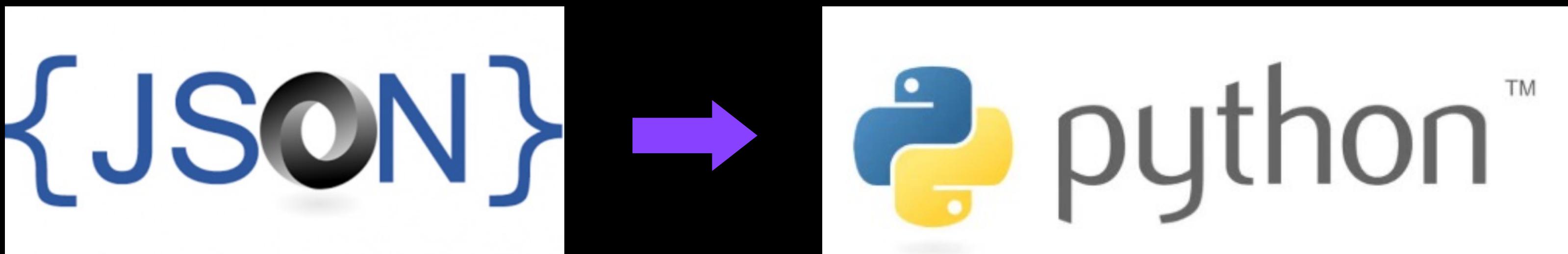
Python 딕셔너리

```
data = {  
    "name": "홍길동",  
    "age": 25,  
    "city": "서울"  
}
```

JSON을 Python에서 사용하려면?

JSON 데이터 파싱

- JSON 형식의 문자열을 Python 딕셔너리로 변환하는 과정
- JSON 데이터를 Python에서 다룰 수 있도록 변환



json.loads() 사용법

```
import json

json_data = '{"name": "홍길동", "age": 25, "city": "서울"}'
python_dict = json.loads(json_data)

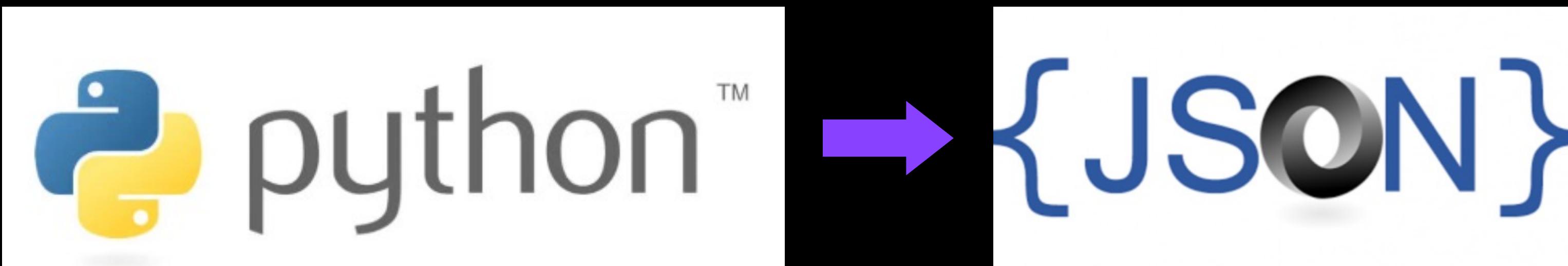
print(python_dict["name"]) # 홍길동
```



Python 딕셔너리를 JSON으로 변환하려면?

JSON으로 변환

- JSON은 웹 API, 데이터 저장, 전송 등에 사용되기 때문에 Python 데이터를 JSON 문자열로 변환해야 함
- Python에서 JSON 데이터로 변환하면 파일 저장, 전송, 공유가 편리해짐



json.dumps() 사용법

```
import json

python_dict = {"name": "홍길동", "age": 25, "city": "서울"}
json_data = json.dumps(python_dict)

print(json_data)
# 출력: '{"name": "홍길동", "age": 25, "city": "서울"}'
```

JSON 데이터를 파일로 저장하고 불러오기

JSON 데이터를 파일로 저장하는 이유

- JSON은 파일로 저장하여 재사용할 수 있음
- API 응답 데이터를 JSON 파일로 저장하면 테스트 및 분석이 가능
- 크롤링한 데이터를 JSON 형식으로 저장하여 다른 프로그램과 공유 가능

JSON 파일 저장 (json.dump())

```
import json

data = {"name": "홍길동", "age": 25, "city": "서울"}

with open("data.json", "w", encoding="utf-8") as f:
    json.dump(data, f, ensure_ascii=False, indent=4)
```

JSON 파일 불러오기 (json.load())

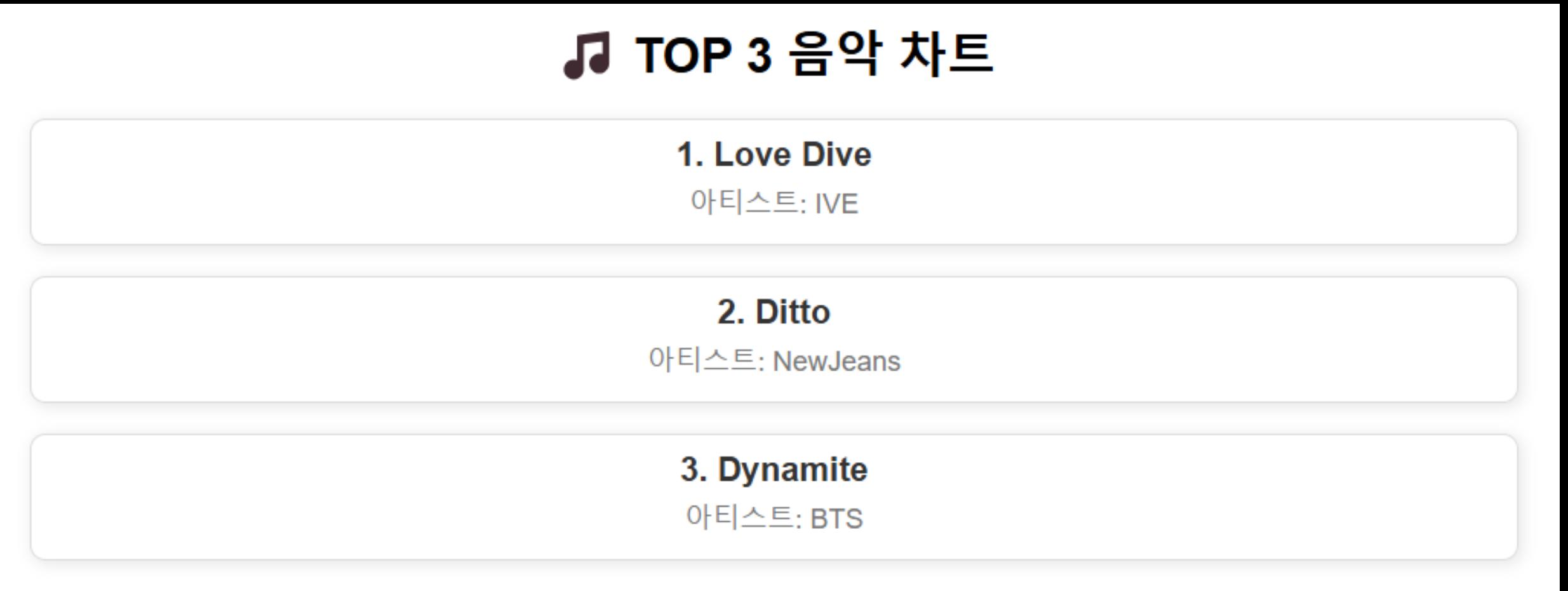
```
with open("data.json", "r", encoding="utf-8") as f:
    loaded_data = json.load(f)

print(loaded_data["name"]) # 홍길동
```

실습3 : 음악 차트 JSON으로 저장하기

실습 목표

- Selenium을 활용하여 음악 차트의 곡명과 아티스트 정보를 크롤링
- 크롤링한 데이터를 Python 딕셔너리로 변환하여 JSON 파일로 저장
- 문자열 가공 (`replace()`, `strip()`)을 활용해 불필요한 문자 제거



실습 과정

1. Selenium을 사용해 HTML 파일에서 음악 차트 정보를 크롤링
2. 곡명과 아티스트 정보를 가져오고, 불필요한 문자열을 제거 (`replace()`, `strip()`)
3. 크롤링한 데이터를 Python 딕셔너리 형태로 변환
4. `json.dump()`를 활용해 `chart_data.json` 파일로 저장 후 콘솔창으로 저장 완료 시 문구 확인

여러 개의 데이터를 한 번에 크롤링하기

단일 요소 vs 다수 요소 선택

- `find_element()` → 하나의 요소 선택
- `find_elements()` → 여러 개의 요소 선택

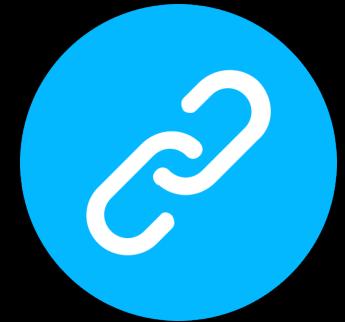
`find_elements()`의 역할

- 같은 태그를 가진 여러 개의 요소를 리스트로 반환
- 페이지 내 여러 상품, 뉴스 목록, 댓글 데이터 수집 가능

```
elements = driver.find_elements("css selector", ".product-name")
for element in elements:
    print(element.text)
```



Selenium으로 링크 수집: get_attribute



페이지 탐색과 링크 추출이 필요한 이유

- Selenium은 버튼 클릭, 링크 이동 등 페이지 탐색 기능 제공
- 자동화된 웹 크롤링을 위해 링크를 가져와 활용 가능

get_attribute("href") 사용법

```
# ✓ 모든 <a> 태그(링크 요소) 찾기
links = driver.find_elements("tag name", "a")

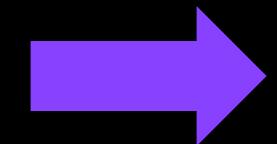
# ✓ 찾은 모든 링크의 href 속성 가져와 출력
for link in links:
    print(link.get_attribute("href"))
```

웹에서 텍스트 & 이미지 URL 가져오기

Selenium을 활용한 텍스트 및 이미지 수집

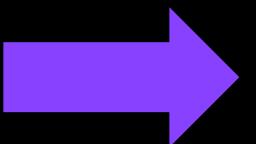
- 웹페이지에서 텍스트 데이터 추출 (.text)

```
text = driver.find_element("tag name", "h1").text  
print(text)
```



- 웹페이지에서 이미지 URL 추출 (get_attribute("src"))

```
image = driver.find_element("tag name", "img").get_attribute("src")  
print(image)
```





크롤링한 데이터를 CSV 파일로 저장해보자



CSV

- Comma-Separated Values (쉼표로 구분된 값)
- 엑셀, 데이터 분석, DB 저장 등에 사용되는 대표적인 데이터 형식
- 텍스트 기반이라 가볍고 읽기 쉬움

CSV 데이터 예시 (data.csv)

```
employees.csv - Notepad
File Edit Format View Help
EMPID,LNAME,FNAME,BDATE
1236,Smith,John,05-01-1962
2398,Adams,John,03-22-1976
4534,Johnson,Mary,10-07-1971
7834,Kirby,Frank,05-27-1978
9823,Harris,Kathy,11-18-1982
9875,Jones,Bill,01-25-1953
```



CSV 파일 다루기: 저장, 읽기

1. csv.writer() - 리스트 데이터를 CSV로 저장

- 기능: 2차원 리스트 데이터를 CSV 파일로 저장
- 사용법: writerows()를 사용해 여러 행을 한 번에 저장

```
import csv

data = [["이름", "나이", "도시"], ["홍길동", 25, "서울"], ["김철수", 30, "부산"]]

with open("data.csv", "w", encoding="utf-8", newline="") as f:
    writer = csv.writer(f)
    writer.writerows(data) # 여러 행 저장
```

2. csv.reader() - CSV 파일 불러오기

- 기능: CSV 파일을 읽고 리스트 형태로 변환
- 사용법: for row in reader를 사용해 한 줄씩 처리

```
import csv

with open("data.csv", "r", encoding="utf-8") as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)
```



JSON, CSV, XML 비교

[JSON, CSV, XML]

항목	CSV	JSON	XML
구조	단순한 테이블 형식 (행, 열)	Key-Value 기반의 계층 구조	태그 기반의 계층 구조
데이터 표현	콤마(,)로 구분된 단순한 문자열	{ "key": "value" } 형태의 객체	<key>value</key> 형태의 태그
가독성	사람이 쉽게 읽을 수 있음 (단순)	사람이 읽기 쉬움 (구조적)	사람이 읽기 어려움 (태그 많음)
확장성	확장 어려움 (고정된 테이블 구조)	확장 용이 (중첩 가능)	확장 용이 (계층적)
저장 공간	가장 가볍고 용량이 작음	중간 정도의 크기	가장 무거움 (태그 포함)
사용 사례	엑셀, 데이터 분석	API, 웹 애플리케이션, 데이터 저장	설정 파일, 문서 저장 (RSS, SOAP 등)
호환성	대부분의 데이터 처리 툴에서 지원	웹 및 백엔드에서 널리 사용	일부 시스템에서만 필요
검색/조회	인덱스 없이 빠름 (간단)	빠름 (Key 검색)	느림 (DOM 탐색 필요)
주요 활용처	엑셀, 데이터 분석, 로그 파일	웹 API, 모바일 앱 데이터 저장	설정 파일, 문서 변환 (HTML, RSS 등)



실습4 : 인기 게시글 가공해서 CSV 저장하기

실습 목표

- Selenium을 활용하여 대학생 커뮤니티 인기 게시물 크롤링
- 크롤링한 데이터를 가공하여 CSV 파일(community_posts.csv)로 저장
- CSV 파일을 생성하고 엑셀에서 열어볼 수 있는 형식으로 저장하는 방법 학습

실습 과정

1. Selenium을 사용하여 게시물 제목과 추천 수 크롤링
2. 게시물 제목에서 '[']' 대괄호 제거 (replace())
3. 추천 수에서 "추천 수: " 부분 제거 (replace())
4. CSV 파일을 생성하고, 첫 번째 행에 헤더 추가 (writer.writerow())
5. 가공한 데이터를 CSV 파일에 저장 (writer.writerows())
6. CSV 파일을 생성하고 저장된 데이터를 콘솔창을 통해 확인

The screenshot displays a list of three popular posts from a student community:

- [이벤트] 이번 학기 장학금 신청하세요!**
추천 수: 120
- [시험] 공부 꿀팁 공유!**
추천 수: 98
- [모집] 동아리 신입 함께할 사람~**
추천 수: 85



Selenium 자주 발생하는 오류와 원인

Selenium 오류 발생 원인

- 요소가 존재하지 않거나 찾을 수 없음 (`NoSuchElementException`)
- 페이지 로딩이 끝나지 않음 (`TimeoutException`)
- 웹 요소가 아직 사용 불가능 (`ElementNotInteractableException`)
- 팝업/알림창이 가려서 요소 클릭 불가능 (`ElementClickInterceptedException`)

오류 유형	발생 원인
<code>NoSuchElementException</code>	찾으려는 요소가 존재하지 않음
<code>TimeoutException</code>	웹페이지 로딩이 오래 걸려 요소를 찾지 못함
<code>ElementNotInteractableException</code>	요소가 보이지 않거나 비활성화됨
<code>ElementClickInterceptedException</code>	다른 요소가 가려서 클릭 불가능



Selenium에서 오류를 잡아보자!

예외 처리

: 프로그램 실행 중 오류가 발생해도 멈추지 않고 정상적으로 진행되도록 처리하는 방법

기본 예외 처리 (try-except)

```
try:  
    element = driver.find_element("id", "search")  
    element.click()  
except NoSuchElementException:  
    print("요소를 찾을 수 없습니다! 다시 확인하세요.")
```

로그 기록 (logging 모듈 활용)

```
import logging  
  
logging.basicConfig(filename="selenium.log", level=logging.INFO)  
  
try:  
    element = driver.find_element("id", "search")  
    element.click()  
except NoSuchElementException as e:  
    logging.error(f"오류 발생: {e}")
```

오류 예방 방법

Selenium 오류 예방 방법



1. 요소가 존재하는지 확인 후 실행

- `find_element()`를 사용하면 요소가 없을 때 오류 발생
- `find_elements()`를 사용하면 오류 없이 빈 리스트 반환

2. 페이지 로딩을 기다린 후 실행 (WebDriverWait)

- 페이지가 완전히 로딩되지 않으면 요소를 찾을 수 없어 오류 발생
- `WebDriverWait`을 사용하여 요소가 나타날 때까지 기다림

3. 오류 발생 시 예외 처리 (try-except)

- 예상 가능한 오류를 `try-except`로 감싸서 프로그램이 중단되지 않도록 방지

4. 로그 기록을 남겨 디버깅을 쉽게 만들기

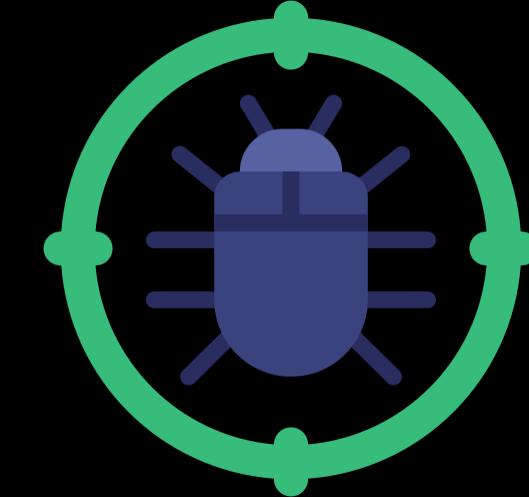
- 오류 발생 시 원인을 빠르게 찾기 위해 `print()`나 `logging`을 활용



Selenium 오류를 잡아보자!

디버깅

- 프로그램의 오류를 찾아 수정하는 과정
- Selenium 테스트 실행 중 어느 부분에서 문제가 발생하는지 확인하는 것이 중요



1. print()를 활용한 디버깅

```
element = driver.find_element("id", "search")
print("찾은 요소:", element.text) # 요소가 올바르게 선택되었는지 확인
```

2. breakpoint()를 활용한 중단점 설정

```
driver.get("https://example.com")
breakpoint() # 여기서 실행이 멈추고 디버깅 가능
```



실행 중 오류가 발생하면? try-except

try-except 문

: 오류(예외)가 발생해도 프로그램이 중단되지 않고 정상적으로 실행되도록 처리하는 구조

try-except 기본 구조

```
try:  
    # 실행할 코드 (오류가 발생할 가능성이 있는 부분)  
    실행할_코드  
except 예외_타입:  
    # 오류 발생 시 실행할 코드  
    오류_처리_코드
```

```
try:  
    element = driver.find_element(By.ID, "login")  
    element.click()  
except NoSuchElementException:  
    print("X 요소를 찾을 수 없습니다! 확인해보세요.")
```

☰ 오류가 발생하면 기록! (logging 활용)

로그(Log)

- 프로그램 실행 중 발생한 이벤트(정보, 오류)를 기록하는 것
- Selenium 실행 중 어떤 오류가 발생했는지 기록하면, 문제 해결이 쉬워짐

Python logging 모듈 활용법

```
import logging

logging.basicConfig(filename="selenium.log", level=logging.INFO)

try:
    element = driver.find_element(By.ID, "search")
    element.click()
except NoSuchElementException as e:
    logging.error(f"오류 발생: {e}")
```

- 실행 중 오류가 발생했을 때 나중에 확인 가능
- print() 대신 로그를 사용하면 디버깅이 더 편리함



실습5 : 취업 사이트에서 발생하는 예외 처리

실습 목표

- Selenium을 사용하여 취업 공고 데이터를 크롤링
- try-except 문을 활용하여 오류를 처리하는 방법 학습

실습 과정

1. Selenium을 사용해 취업 공고에서 회사명과 직무 크롤링
2. try-except 문을 사용하여 크롤링 중 발생하는 오류 해결
3. 데이터를 가공한 후 CSV 파일(job_listings.csv)로 저장
4. Selenium이 특정 요소를 찾지 못할 경우 예외를 처리하여 크롤링이 멈추지 않도록 구현

The screenshot shows a web page titled "인기 취업 공고" (Popular Job Ads) with a fire emoji. It displays three job listings:

- 네이버**
직무: 백엔드 개발자
- 카카오**
직무: 프론트엔드 개발자
- 삼성전자**
직무: 데이터 분석가



웹 크롤링, 합법일까?

웹 크롤링을 할 때 고려해야 할 법적 문제

- robots.txt 규칙 준수
- 개인정보 보호법 및 서비스 약관 확인
- 과도한 요청으로 서버에 부담을 주지 않기

robots.txt

- 사이트 소유자가 크롤링 허용 여부를 명시하는 파일
- 크롤링하기 전에 반드시 robots.txt 확인!
- Disallow된 페이지는 크롤링하면 안 됨!

예제 (example.com/robots.txt)

```
User-agent: *
Disallow: /private/
Allow: /public/
```

테스트 자동화, 어디까지 가능할까?

QA(품질 보증)에서 테스트 자동화 시 고려해야 할 윤리적·법적 문제

- 개인정보 보호 (개인정보가 포함된 테스트 데이터 사용 금지)
- 웹사이트 서비스 약관 준수 (무단 테스트 및 크롤링 방지)
- 과도한 테스트 요청으로 시스템 장애 유발 금지

1. 개인정보 보호법 준수

- 테스트 데이터에 실제 사용자 정보(이메일, 주소, 주민등록번호 등) 포함 금지
- 가짜 테스트 계정 및 샘플 데이터를 사용해야 함
- 개인정보를 로그로 저장하면 법적 문제 발생 가능

2. 서비스 약관 및 법적 규제 준수

- 테스트 자동화가 사이트 약관을 위반하는지 확인
- 특정 기능 테스트 중 과부하 발생 가능성을 고려해야 함
- API 테스트 시, 요청 빈도를 조절하여 서버 부하를 최소화

3. 테스트 자동화 윤리 가이드라인

- 테스트 자동화는 서비스 품질 개선을 위한 목적으로 사용해야 함
- 사용자 경험을 해치지 않는 방식으로 진행해야 함
- 시스템 과부하 또는 데이터 오염을 방지하는 설계를 고려해야 함





퀴즈 타임



QUIZ 9번부터 15번을 풀어봐요!