

# 분자량 계산기

학번: 2018086

이름: 주동철

Github address: dongcheolju

## 1. 계산기의 목적

- a. 두산백과에서 정의하는 분자식이란, 분자의 조성 그리고 그 분자를 구성하는 원자의 수를 원소기호를 써서 나타낸 식으로 물질을 이루는 분자의 짜임새를 나타낸 것이다.  
분자식을 이용해서 화학물질의 구성을 간단하게 정리할 수 있고, 분자에 어느 원소가 어느정도 포함되어 있는지 알 수 있다. 이러한 분자를 취급하고 반응하며 계산하는 데 있어서 다양한 화학 개념이 필요한데, 가장 기초가 이 중 원자량은 가장 기초가 되는 개념이라고 말할 수 있다.  
원자량은 특정 원자의 평균 무게를 통일 원자 질량 단위에 의한 상대적인 비율로 나타낸 것이다. 쉽게 말해 원자의 무게를 나타내는 질량 상수라고 생각하면 된다.<sup>1</sup> 이러한 원자량을 이용하여 특정 분자식의 무게인 '분자량'을 구할 수 있는데, 원자량은 각 원소마다 고유의 값을 가지기 때문에 분자량을 구하기 위해서 분자를 이루고 있는 원소의 원자량을 찾아 조합하여 분자량을 구해야 하는 번거로움이 있다. 필자는 이러한 불편함을 줄이고자 파이썬을 이용하여 분자식을 입력했을 때 자동으로 분자량을 계산해 출력해주는 코드를 작성하여 화학을 취급하는 사람이 편하게 분자량을 구할 수 있도록 하고자 한다.
- b. 계산기 활용 대상:  
분자량을 구하고 다루는 화학 교사 및 화학과 관련된 모든 사람

## 2. 계산기의 네이밍의 의미

- a. 각 원소의 원자량을 이용하여 분자식의 분자량을 구하는 계산기

---

<sup>1</sup> 질량수가 12 인 탄소원자의 1/12 의 해당하는 값이 원자질량상수 1 에 해당된다.

### 3. 계산기 개발 계획

#### a. 입력 변수:

종류	모양	설명
라이브러리	pandas	
모듈	re	
변수	Symbol_Atomic_Weight	Symbol_Atomic_Weight.csv 파일의 경로 저장
변수	df	Symbol_Atomic_Weight.csv 파일을 불러와 저장하는 데이터 프레임
매개변수	chemical_formula	사용자가 입력하는 분자식 저장
사용자 함수	calculate_molecular_weight	분자량 계산기 (사용자 지정 함수)
함수	findall	chemical_formula 에서 ([A-Z][a-z]*) (\d*) 형태를 찾아 elements 에 리스트 형태로 저장하는 역할
변수	total_atomic_mass	분자량 값을 저장하는 변수 (초기 값 0.0)
열이름	Symbol	csv 파일의 첫 번째 열이름 (원소명)
열이름	AtomicMass	csv 파일의 두 번째 열이름 (원자량)
변수	count	원소의 갯수 저장
변수	atomic_mass	원자량 저장
변수	element_data	csv 파일에서 찾은 원소 정보 저장

#### b. 함수 연산 과정

```
def calculate_molecular_weight(chemical_formula):
    elements = re.findall(r'([A-Z][a-z]*) (\d*)', chemical_formula)
    total_atomic_mass = 0.0

    for element, count in elements:
        element_data = df[df['Symbol'] == element]
        if not element_data.empty:
            atomic_mass = element_data.iloc[0]['AtomicMass']
            count = int(count) if count else 1
            atomic_mass = float(atomic_mass)
            atomic_mass = count * atomic_mass
            total_atomic_mass = atomic_mass + total_atomic_mass
        else:
            print(f"{element}를 찾을 수 없습니다.")

    return total_atomic_mass
```

함수 형식

```
def calculate_molecular_weight(chemical_formula):
```

: 함수의 정의와 매개변수 지정

```
elements = re.findall(r'([A-Z][a-z]*) (\d*)', chemical_formula)
```

: 계산 전, 입력한 매개변수에서 불러온 re 모듈의 'findall' 함수<sup>2</sup>를 사용하여 'chemical\_formula'의 화학식을 원소와 갯수로 분리하여 'elements'에 리스트 형태로 저장

```
화학식을 입력하세요 : H2O
[('H', '2'), ('O', '1')]
```

H2O 입력시, elements 에 저장되는 리스트

<sup>2</sup> findall 함수: 매치된 부분을 모두 리스트로 취득하는 함수

```
total_atomic_mass = 0.0
```

: 총 분자량을 저장하는 변수 'total\_atomic\_mass' 초기값 0.0 지정

```
for element, count in elements:
```

: 'elements'에 저장된 각 원소에 대한 요소를 반복

```
element_data = df[df['Symbol'] == element]
```

: 저장한 데이터 프레임의 Symbol 열에서 입력한 원소를 찾아 그 데이터를 'element\_data'에 저장

여기서 <df[df['Symbol'] == element]>은 CSV 파일의 Symbol 열에서 해당 원소를 찾는 조건

화학식을 입력하세요 : H2O

Symbol	AtomicMass
7	0 16.0

H2O 입력시, element\_data 에 저장되는 프레임

```
if not element_data.empty:
```

: 직역하면 "element\_data 가 비어있지 않다면"으로, 5 행에서 원소가 올바르게 찾아졌는지 확인하는 과정

```
atomic_mass = element_data.iloc[0]['AtomicMass']
```

: element\_data 에 해당하는 AtomicMass 열의 0 번째 행을 가져와 atomic\_mass 에 저장

```
count = int(count) if count else 1
```

: count 를 정수로 저장하되, 비어있을 경우 1 로 저장(0 일 경우, 값이 분자량 값이 0 이 되기 때문에 1 로 저장)

```
atomic_mass = float(atomic_mass)
```

: 문자형을 부동소수 숫자형으로 전환

```
atomic_mass = count * atomic_mass
```

: 원자량과 원소의 갯수를 곱하는 과정

```
total_atomic_mass = atomic_mass + total_atomic_mass
```

: 구해진 원자량을 total\_atomic\_mass 에 저장

```
else:
```

```
print f'{element}를 찾을 수 없습니다.'
```

: 5 행의 조건문이 참이 아닌 경우 실행되어 분자량을 구할 수 없다는 결과를 나오게 함

c. 연산 과정은 어떻게?, 조건문은 왜 필요하며, 왜 이렇게 설계했는지 등

각 원소의 원자량은 정해져있고, 그러한 원소의 종류는 튜플 자료형으로 만들어 넣기에는 120 여가지로 그 수가 많다.

그렇기 때문에 데이터 프레임을 컴퓨터 저장 경로에서 가져와 데이터 프레임으로 읽고 쓸 수 있는 "pandas" 라이브러리를 import 해서 컴퓨터 특정 경로에 원소열과 원자량열로 구성된 csv 파일 불러오는 것을 시작으로 코드를 작성했다.

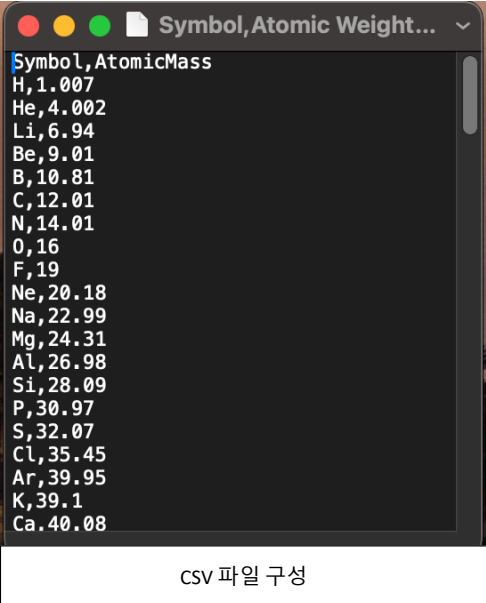
사용자로부터 분자식을 입력받는데, 이 분자식은 문자와 숫자로 이루어져 있고, 각 원소는 "A"처럼 대문자 기호로 이루어질 수 있고, "Aa"처럼 대문자와 소문자로 구성되어 있을 수 있기 때문에, 이러한 경우도 고려하여 사용자에게 입력받은 문자열을 분리하여 원소와 원소의 개수를 구분할 수 있어야 한다. 필자는 인터넷 검색을 통해 "re"모듈을 발견했고, "re" 모듈의 "findall"함수를 이용하여 각 원소와 개수를 분리하여 리스트 형태로 저장하여 반복문에 사용할 수 있게 구성해보았다. 구성형태는 다음과 같다. [('Aa', '1'), ('Bb', '2'), ('Cc', '3')]

반복문의 반복 과정은 간단하다. 분자식으로 만든 리스트의 요소를 하나씩 csv 파일에서 찾아 해당 원소 하나만 담겨있는 새로운 데이터 프레임인 "element\_data"를 만들고 계산을 진행하며 반복한다.

## 4. 계산기 개발 과정

### a. 계획 후 실제 개발 과정

처음 분자량 계산기를 생각했을 때, 1 학기에 배운 pandas 라이브러리를 사용할 계획을 했다. 앞서 언급했던 것처럼 수많은 원소의 원자량은 각자 다르기 때문에, 직접 데이터프레임으로 활용할 수 있는 csv 파일을 만들었다.



Symbol	AtomicMass
H	1.007
He	4.002
Li	6.94
Be	9.01
B	10.81
C	12.01
N	14.01
O	16
F	19
Ne	20.18
Na	22.99
Mg	24.31
Al	26.98
Si	28.09
P	30.97
S	32.07
Cl	35.45
Ar	39.95
K	39.1
Ca	40.08

csv 파일 구성

### b. 각 함수의 역할

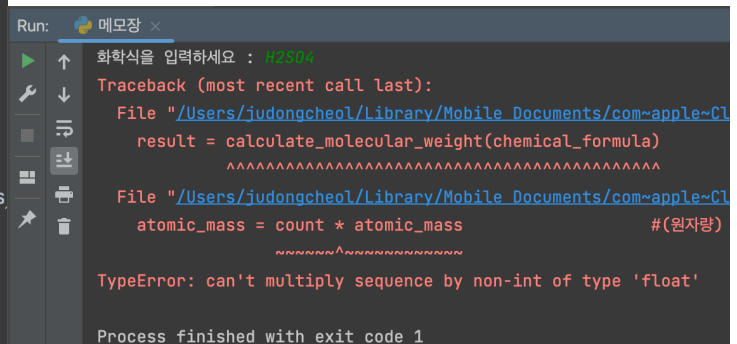
findall : re.findall(패턴, 변수)형태로 원하는 패턴을 변수 안에서 모두 찾아 리스트의 형태로 저장하는 함수로, 분자량 계산기에서는 사용자로 부터 입력받은 분자식에서 원소와 그 원소의 갯수를 찾아 elements 에 리스트로 저장한다.

### c. 에러 발생 지점

처음 함수의 코드를 다음과 같이 작성했다.

```
for element, count in elements:
    #원소를 데이터 프레임에서 찾기
    element_data = df[df['Symbol'] == element]
    if not element_data.empty:
        atomic_mass = element_data.iloc[0]['AtomicMass']
        atomic_mass = float(atomic_mass)
        atomic_mass = count * atomic_mass
        total_atomic_mass = atomic_mass + total_atomic_mass
    else:
        print(f"{element}를 찾을 수 없습니다.")

#return total_atomic_mass
return total_atomic_mass
```



```
Run: 메모장 x
화확식을 입력하세요 : H2O
Traceback (most recent call last):
  File "/Users/judongcheol/Library/Mobile Documents/com~apple~Cl
    result = calculate_molecular_weight(chemical_formula)
    ~~~~~
  File "/Users/judongcheol/Library/Mobile Documents/com~apple~Cl
    atomic_mass = count * atomic_mass          #(원자량)
    ~~~~~
TypeError: can't multiply sequence by non-int of type 'float'

Process finished with exit code 1
```

이 코드로 결과를 추출하면 **<TypeError: can't multiply sequence by non-int of type 'float'>**라는 오류 코드가 발생한다. 해당 오류는 count 변수와, atomic\_mass 의 자료형이 다르기 때문에 곱셈을 진행할 수 없어 발생하는 오류이다. count 는 원소의 개수를 저장하는 변수로, 초기 자료형은 아무런 변환이 없었기 때문에 문자열이다. 문자열은 사칙연산이 불가능하기 때문에 곱셈이 가능하도록 적절한 자료형 변환이 필요하다.

### d. 에러 발생에 대한 해결책

if 문 내부에 count = int(count) 행을 추가하여 자료형 변환이 가능하게 해보았으나, **<ValueError: invalid literal for int() with base 10: ">**의 오류가 발생한다. 이는 공백 즉 개수가 없는 경우에 발생하는 오류로, 개수의 값이 없는 경우 조치할 수 있는 사항을 추가해야 한다.

이에 최종적으로 `count = int(count) if count else 1` 형태로 코드를 구성하여 자료형 변환과 동시에 `count` 가 비어 있을 때(거짓) 1로 값을 부여하는 방법으로 오류를 해결했다.

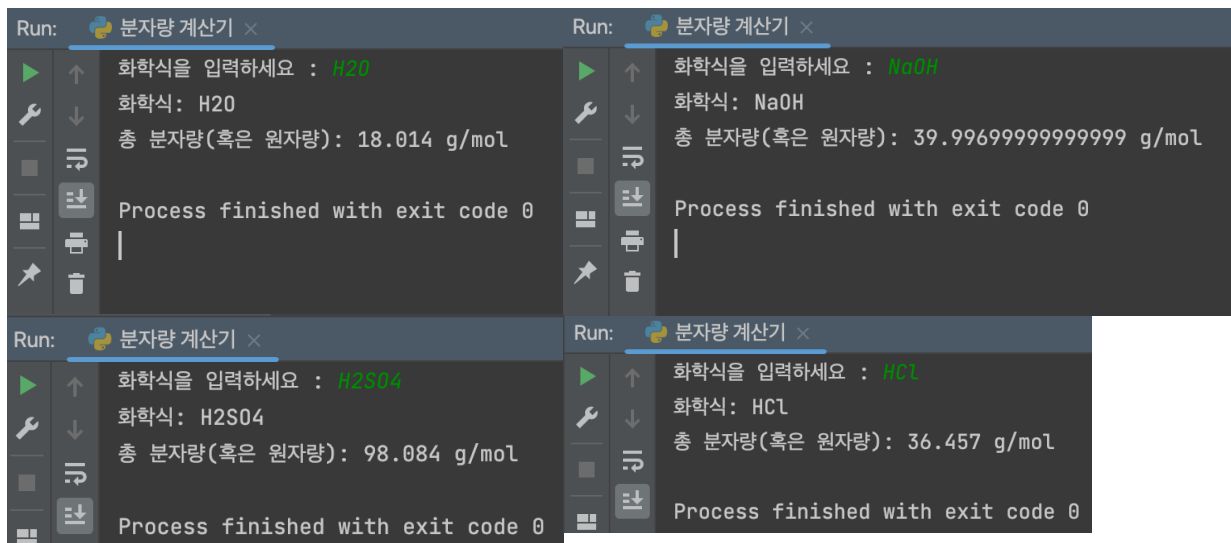
e. 해결책 적용 시 어떻게 변화

반복문의 코드에서 곱하는 요소의 자료형이 달라 발생하는 오류와, 변수에 공백이 있어 변환에 오류가 발생하는 것 모두 수정하여 다음과 같은 형태로 최종적으로 코드를 다듬었다.

```
for element, count in elements:
    #원소를 데이터 프레임에서 찾기
    element_data = df[df['Symbol'] == element]
    if not element_data.empty:
        atomic_mass = element_data.iloc[0]['AtomicMass']
        count = int(count) if count else 1
        atomic_mass = float(atomic_mass)
        atomic_mass = count * atomic_mass
        total_atomic_mass = atomic_mass + total_atomic_mass
    else:
        print(f"{element}를 찾을 수 없습니다.")

#return total_atomic_mass
return total_atomic_mass
```

f. 동작 결과 캡처



## 5. 계산기 개발 후기

### a. 계산기 개발 후 느낀 점 설명

만든 계산기는  $H_2O$ ,  $NaOH$ ,  $NaOH$ ,  $H_2SO_4$  등 간단한 분자의 분자량만 구할 수 있는 코드이다. 하지만 우리가 공부하는 분자식은 이온 분자도 있고, 표기의 편의를 위해 괄호()를 써서 임의로 분자 내에 분자를 묶어 표기하기도 한다. 이러한 부분은 제공 csv 파일을 수정하고 확장하거나, 새로운 csv 파일을 만들고 불러와 그 범위를 넓히며 코드를 보다 견고하게 만들 수 있을 것이다. 마치 함수라는 줄기에 여러 코드를 이용한 가지를 뺀고 변수와 요소라는 나뭇잎으로 풍성하고 우람한 나무를 만드는 것과 같다고 생각한다.

필자는 새로운 코드를 계획하는 것보다, 지금 이 코드에 시간과 노력을 투자하여 더욱 견고하게 만들고 싶은 생각이 있다. 어려서부터 이렇게 직접 나만의 체계를 만들고 남들에게 공유하는 것을 참 좋아했다. 10 년이 흐른 지금 까맣게 잊고 있었던 그 생각과 느낌이 떠올라 초등학생 시절 주동철이 좋아했던 것을 다시금 느낄 수 있었다.<sup>3</sup>

---

<sup>3</sup> 교수님 감사합니다.