환산 재해율 간편 계산기

학번: 2218042

이름: 임경용

Github address:
https://github.com/yong2218042/Conversion-Disaster-Rate-Easy-Calculator.git

1. **계산기의 목적**

   '환산 재해율'이란 재해자수의 경우 사망자에 대한 가중치를 부여하는 것이다.
   조금 더 쉽게 표현하면 근로자 100 명당 발생하는 재해자 수의 비율이다.
   이러한 '환산 재해율'은 안전성적 평가와 재해예방대책자료로 사용되기 위해 필요하다.
   '환산 재해율'은 건설 안전 관련된 용어이다. 그래서 활용 대상은 건설안전기사 시험을 준비하는 사람이 계산을 하는 과정에서 사용하거나 건설업체에서 평가지표를 마련할 때 사용할 수 있다.

2. **계산기의 네이밍의 의미**

   계산기 이름은 중요 키워드인 '환산 재해율'을 강조하고, 또한 계산기의 본질인 편리함을 강조하여 '환산 재해율 간편 계산기'로 하였다.

3. **계산기 개발 계획**

   입력 변수는 'num_incidents'로 사용자로부터 입력받은 환산 재해자수를 의미하고, 'num_employees'는 사용자로부터 입력 받은 상시 근로자수를 의미한다.
   개발한 함수는 'calculate_incident_rate'이다. 환산 재해자수(num_incidents)와 상시 근로자수(num_employees)를 입력 받은 뒤 이를 사용해 환산 재해율을 계산한 뒤 환산 재해율을 반환한다. 환산 재해율 식은 (num_incidents / num_employees) * 100 이다.

   함수는 입력된 환산 재해자수와 상시 근로자수를 사용하여 환산 재해율을 계산하고, 계산 결과를 반환하는 역할을 수행한다. 또한 예외 처리를 통해 사용자가 유효하지 않은 입력을 할 경우 오류 메시지를 반환해 프로그램 안전성을 유지한다.
   조건문은 입력이 올바른 형식인지 여부 확인과 프로그램의 종료 여부, 예외 처리, 반복 등에 사용된다.

4. 계산기 개발 과정

우선 def 를 통해 환산 재해율을 계산할 함수를 정의했고, try 를 활용해 예외를 처리했다. 그리고 입력을 받을 때 정수 형태로 입력을 받아야 하기 때문에 int 를 사용했고, 이후 계산식을 입력했다. Return 으로 값을 반환한 뒤 except 로 예외처리를 했다. Except 는 숫자가 아닌 값을 입력한 경우 처리하기 위함이다. Return 을 사용한 뒤 숫자가 아닌 값을 입력한 경우 반환할 메시지를 작성했다. 메인프로그램 작동 과정은 if _ _name_ _ == "_ _ main_ _": 코드를 입력해준다. 이 코드는 현재 파이썬 파일이 직접 실행될 때만 내부 코드가 실행되도록 해주는 코드이다. 이후 print 로 프로그램 시작 메시지를 출력해주고, while True: 로 무한 루프를 해준다. 기능을 계속 실행하기 위해 무한루프 반복문을 활용한 것이다. 이후 함수를 호출한 뒤 result 변수에 저장한다. 그리고 isinstance 을 사용하여 데이터 타입을 확인하고, str 로 result 가 문자열인지 확인한다. Else 문으로 result 가 문자열이 아닌 경우(환산 재해율이 계산된 경우) 실행하도록 한다. 환산 재해율을 소수 둘째 자리까지 출력하도록 하는 코드를 작성하고, 다시 계산할지 묻는 코드를 작성한다. Y 를 입력하지 않은 경우에 루프를 빠져나오도록 하고 break 로 프로그램을 종료하도록 한다.

'calculate incident rate' 함수는 사용자로부터 환산 재해자수와 상시 근로자수를 입력 받는다. 입력 받은 값은 정수로 변환되고, 환산 재해율을 계산하는데 사용되며 계산된 환산 재해율은 반환된다. 입력이나 계산 중 오류가 발생하면 예외 처리를 통해 "유효한 숫자를 입력해야 합니다." 메시지가 반환된다.
'if _ _name_ _ == "_ _ main_ _":' 코드는 프로그램이 직접 실행될 때 실행된다. 프로그램 시작 메시지를 출력한 후, 무한 루프를 시작한다.
'result = calculate incident rate()'를 호출해 calculate incident rate 함수를 실행하고, 결과를 result 변수에 저장한다. 이렇게 환산 재해율 또는 오류 메시지를 얻는다.
'if isinstance(result, str):'는 result 가 문자열인지 확인한다. 만약 환산 재해율이나 오류 메시지가 아닌 다른 형식의 값이 반환된다면, 이 부분은 실행되지 않는다.
'print(result)'는 오류 메시지를 출력하며, 그렇지 않은 경우 환산 재해율이 계산된 것이므로 'print(f"환산 재해율:{result:.2f}%'를 사용해 계산된 환산 재해율을 출력한다.

‘input’은 사용자에게 다시 계산할지 여부를 확인하기 위해 사용하고 사용자가 “y”를 입력하지 않으면 루프를 빠져나와 프로그램을 종료한다.

에러 발생은 사용자가 정수 형태가 아닌 숫자를 입력한 경우에 정수 변환 시 오류를 발생시켰다. 또한 사용자가 상시 근로자수를 0 으로 입력하면 환산 재해율을 계산할 때 0 으로 나누는 오류가 발생했다. 또한 사용자에게 다시 계산할 것인지를 물어봤을 때 “y”, “n” 이 아닌 다른 값 입력을 했을 때 오류가 발생했다.

해결책으로는 정수 변환 오류의 경우 try 및 except 문을 사용하여 사용자의 입력이 예외를 발생시키지 않도록 하고, 변환이 실패한 경우 사용자에게 유효한 숫자를 입력하라는 메시지를 표시하고 다시 입력을 요청할 수 있다. 다음으로 0 으로 입력 시 발생하는 오류는 상시 근로자 수를 입력하기 전 0 이상의 값을 작성하도록 하고, 0 을 입력한 경우 다시 입력하게 한다. 마지막으로 “y”, “n” 오류는 다른 입력을 받았을 시 정확한 선택을 다시 입력하도록 하는 메시지를 작성하여 해결할 수 있다.

다시 계산 오류 해결 적용 변화 과정

<해결 전>

```
환산 재해율 계산기
재해자수를 입력하세요: >? d
유효한 숫자를 입력해야 합니다. 다시 입력하세요.
재해자수를 입력하세요: >? 12
상시 근로자수를 입력하세요: >? 265
환산 재해율: 4.53%
다시 계산하시겠습니까? (y/n): >? o
In [8]: o
Traceback (most recent call last):
  File "C:\Users\user\anaconda3\lib\site-packages\IPython\core\interactiveshell.py", line 3457, in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)
  File "<ipython-input-8-2791b773ce0c>", line 1, in <module>
    o
```

<해결 후>

```
환산 재해율 계산기
재해자수를 입력하세요: >? o
유효한 숫자를 입력해야 합니다. 다시 입력하세요.
재해자수를 입력하세요: >? 12
상시 근로자수를 입력하세요: >? 100
환산 재해율: 12.00%
다시 계산하시겠습니까? (y/n): >? o
올바른 입력을 하십시오. 'y' 또는 'n' 중 하나를 입력하세요.
다시 계산하시겠습니까? (y/n): >? n
```

<동작 결과>

```
환산 재해율 계산기
재해자수를 입력하세요: >? 985642
상시 근로자수를 입력하세요: >? 321425964
환산 재해율: 0.31%
다시 계산하시겠습니까? (y/n): >? o
올바른 입력을 하십시오. 'y' 또는 'n' 중 하나를 입력하세요.
다시 계산하시겠습니까? (y/n): >? y
재해자수를 입력하세요: >? 10000001
상시 근로자수를 입력하세요: >? 0
상시 근로자수는 0 이상의 값이어야 합니다. 다시 입력하세요.
재해자수를 입력하세요: >? 1
상시 근로자수를 입력하세요: >? 1
환산 재해율: 100.00%
다시 계산하시겠습니까? (y/n): >? y
재해자수를 입력하세요: >? 1000000
상시 근로자수를 입력하세요: >? 12
환산 재해율: 8333333.33%
다시 계산하시겠습니까? (y/n): >? n
```

5. 계산기 개발 후기

솔직히 안전 관련 학과에서 굳이 컴퓨터 프로그래밍을 배울 필요가 있는지에 대해 의문이 남아있었는데 직접 계산기 프로그램을 개발해보면서 컴퓨터 프로그래밍은 분야를 불문하고 접근성도 높고 활용도도 매우 높다는 것을 느꼈다. 그리고 계산기를 개발하면서 계획하는 과정에서 강의 시간에 배운 내용들이 어렴풋이 생각도 났고, 오류를 해결하는 과정에서도 막막하기 보다는 실습을 통해 배운 내용을 토대로 어떻게 하면 해결이 가능할지 얼추 구상도 가능했다.
또한 개발하는 과정이 생각보다 재밌어서 연습을 통해 추후에 다른 프로그램을 직접 개발하는 시도를 해보고 싶다는 생각이 들었고, 컴퓨터 프로그래밍 관련한 구시대적 발상도 전환할 수 있는 좋은 계기가 된 것 같다.