계산기의 목적

재료의 **극한 강도, 허용 응력, 단면적**을 입력받아 다음의 세 가지 계산을 수행하는 것을 목적으로 한다.

① 안전율 계산: 사용자로부터 입력받은 재료의 극한 강도와 허용 응력을 이용하여 안전율을 계산한다.

② 허용 응력 계산: 안전율을 이용하여 재료의 극한 강도를 허용 응력으로 변환한다.

③ 허용 하중 계산: 허용 응력과 단면적을 이용하여 재료가 버틸 수 있는 최대 하중을 계산한다.

이 프로그램은 사용자에게 직관적인 인터페이스를 제공하고, 안전공학에서 중요한 계산을 빠르고 정확하게 수행하여 안전성을 보장한다.

사용자가 원하는 작업을 선택하고 필요한 값을 입력함으로써 안전 관련 계산을 편리하게 수행할 수 있다.

계산기 개발 계획

1. 입력 변수:
   - 재료의 극한 강도 (ultimate_strength)
   - 허용 응력 (allowable_stress)
   - 단면적 (cross_section)

2. 연산 과정 설계:

   사용자로부터 입력 받은 값을 기반으로 **안전율**, **허용 응력**, **허용 하중**을 계산한다.

- 안전율 = 재료의 극한 강도 / 허용 응력
- 허용 응력 = 재료의 극한 강도 / 안전율
- 허용 하중 = 허용 응력 × 단면적

3. 코드 구성:
    - 안전율 계산 함수 (print_safety_factor(ultimate_strength, allowable_stress))
    - 허용 응력 계산 함수 (print_allowable_stress(ultimate_strength, safety_factor))
    - 허용 하중 계산 함수 (print_allowable_load(allowable_stress, cross_section))
    - 사용자 입력 함수 (get_user_input())
    - 메인 함수 (main())

      사용자 입력 함수를 호출하여 입력을 받고, 반복문과 조건문을 사용하여 사용자에게 메뉴를 표시하고 작업을 수행한다.

      프로그램 종료 옵션을 선택하면 반복문을 종료한다.

4. 조건문 사용 이유:
    - **사용자 입력 값의 유효성 검사**: 사용자로부터 입력받은 값이 음수이거나 0인 경우, 잘못된 값으로 인식하여 프로그램의 정확성을 유지하기 위해 조건문을 사용한다.
    - **사용자 선택에 따른 기능 수행**: 사용자가 원하는 작업에 따라 알맞은 함수를 호출하기 위해 조건문을 사용한다.

      "1"일 때는 안전율 계산 함수 호출, "2"일 때는 허용 응력 계산 함수 호출, "3"일 때는 허용 하중 계산 함수 호출, "4"일 때는 프로그램 종료 기능을 수행한다.

5. 반복문 사용 이유:
    - **사용자가 원할 때까지 프로그램 유지**: 반복문을 사용하여 사용자가 "4"를 선택하기 전까지 프로그램이 계속해서 동작하도록 한다.

      이렇게 함으로써 사용자가 여러 계산을 수행할 수 있다.
    - **잘못된 입력 시 재시도 유도**: 사용자가 잘못된 입력을 할 경우 프로그램이 종료되지 않고 계속해서 재시도할 수 있도록 하기 위해 반복문을 사용한다.

6. 함수 사용 이유:

    - **모듈화 및 코드 재사용**: 각 기능을 별도의 함수로 구현하여 프로그램을 모듈화하고, 동일한 기능이 필요한 경우 함수를 호출하여 코드의 재사용성을 높인다.

      이는 유지보수 및 확장에 유리하다.

    - **가독성 및 유지보수 향상**: 각 함수는 특정 기능을 수행하므로 코드의 가독성을 높이고, 수정이 필요한 경우 해당 함수만 수정하여 전체 코드의 유지보수를 향상시킨다.

계산기 개발 과정 및 후기

1. 에러 발생 지점 확인:

    - **음수 값 또는 0 입력**: 사용자로부터 입력 받은 값 중에 음수 값 또는 0이 입력되는 경우 에러가 발생할 수 있다.

      허용 응력은 양수이어야 하며, 단면적도 0보다 커야 한다.

    - **입력 값이 숫자가 아닌 경우**: 사용자가 숫자가 아닌 문자열을 입력하는 경우 float(input(...))에서 ValueError가 발생한다.

2. 에러 해결 방법 제시:

    - **음수 값 또는 0 입력 방지**: 사용자로부터 입력 받은 값에 대해 조건문을 사용하여 음수 값 또는 0인 경우 에러 메시지를 출력하고, 적절한 값 입력을 요청한다.

    - **입력 값이 숫자가 아닌 경우 처리**: 'try-except' 구문을 사용하여 사용자 입력을 시도하고, ValueError가 발생한 경우 에러 메시지를 출력하고 다시 입력을 요청한다.

3. 해결 방법 적용 시 변화한 내용:

    - **음수 값 또는 0 입력 방지**: 사용자가 음수 값 또는 0을 입력할 경우 에러 메시지를 출력하고, 올바른 입력을 받을 때까지 반복해서 입력을 요청한다.

    - **입력 값이 숫자가 아닌 경우 처리**: 사용자가 숫자가 아닌 값을 입력하면 에러 메시지를 출력하고, 올바른 숫자를 입력할 때까지 반복해서 입력을 요청한다.

4. 개발 후 느낀점:

    이 프로젝트를 통해 프로그래밍의 기본 개념을 익히고 Python 언어에 익숙해졌다.

    함수, 조건문, 반복문을 사용하는 방법을 배우고, 사용자로부터 입력을 받고 이를 처리하는 방법을 익혔다.

또한, 프로그램의 구조를 설계하고 에러를 처리하는 방법을 습득했다.

이 프로젝트를 통해 가장 중요한 것은 사용자 친화적인 프로그램을 만드는 것이라고 느꼈다.

적절한 안내와 에러 처리를 통해 사용자가 프로그램을 쉽게 이해하고 사용할 수 있도록 하는 것이 중요하다고 생각한다.

또한, 프로그래밍에서는 문제 해결 능력이 매우 중요하다는 것을 깨달았다.

사용자로부터 받은 문제를 정확하게 파악하고, 그에 맞게 코드를 작성하는 과정에서 문제 해결 능력이 길러진 것 같다.

앞으로는 더 복잡한 프로젝트에 도전하여 프로그래밍 스킬을 향상시키고, 실용적인 소프트웨어를 개발해보고 싶다.

계산기의 효과

- 안전공학 분야에서 기본적인 계산을 자동화하여 사용자의 시간과 노력을 절약하고, 안전한 결정을 내릴 수 있도록 지원한다.
- 사용자가 입력한 값을 기반으로 정확한 계산 결과를 반환하여 안전한 설계 및 결정에 도움을 준다.
- 사용자가 입력한 값에 대한 예외 처리를 통해 프로그램의 안정성을 높였다.

  또한, 사용자가 필요한 안전공학 계산을 쉽게 수행할 수 있는 신뢰성 있는 도구로서의 역할을 한다.
- 공학 학습자들에게 중요한 개념을 이해하고 적용하는 방법을 가르치는데 도움을 줄 수 있다.
- 실무에서 안전성을 평가하고 설계할 때 이 프로그램을 활용할 수 있다.