리뷰 관리 더 편해졌계산기

학번: 2118315

이름: 이준호

Github address: https://github.com/leejunho210/HW.01

1. 계산기의 목적

a. 특정날짜들의 리뷰 개수 숫자 세기 라는 단순 반복잡업을 쉽고 간편하게 몇번의 문자 입력과 클릭으로 단순화 시킬 목적으로 만들었다.

b. 계산기 활용 대상: 통신판매업과 마케팅관리를 하는 개인 및 기업 대상 (리뷰 및 고객을 관리해야 하는 사람들)

2. 계산기의 네이밍의 의미

a. 일일이 다 찾아서 리뷰를 관리해야하는 번거로움이 없어졌겠네?에서 언어유희를 통해 졌계산기로 정했다

3. 계산기 개발 계획

a. 입력 변수 : 리뷰 데이터, 시작 날짜, 종료 날짜.
조건문 : 날짜 범위 설정을 위해 사용.
반복문 : 2 주 단위로 데이터를 분할하고 합산.

b. 1. input_dates 함수
사용자로부터 리뷰 날짜, 시작 날짜, 끝 날짜를 입력 받는 함수입니다.

input() 함수로 입력을 받고, 리뷰 날짜는 쉼표로 구분하여 리스트로 변환합니다.
반환 값: review_dates (리뷰 날짜 리스트), start_date (시작 날짜), end_date (끝 날짜).

2. get_review_counts 함수
2 주 간격으로 리뷰 수를 계산하는 함수입니다.

parse_date(date_str): 날짜 문자열을 datetime 객체로 변환하는 내부 함수입니다.

try-except 블록: 날짜 형식 오류를 처리합니다.

while 반복문: 시작 날짜부터 끝 날짜까지 2 주 단위로 구간을 설정하고 리뷰 수를 계산합니다.

sum()과 조건문: 각 구간에 포함된 리뷰 수를 계산합니다.

c. 연산 과정

1. **날짜 문자열을 datetime 객체로 변환**:
   - parse_date(date_str) 함수가 날짜 문자열을 datetime 객체로 변환합니다. 이를 통해 날짜 계산이 가능해집니다.
   - 예외 처리(try-except)를 통해 잘못된 날짜 형식이 입력되면 오류를 잡아냅니다.
2. **2 주 단위로 리뷰 수 계산**:
   - while 반복문을 사용하여 시작 날짜부터 2 주 단위로 구간을 설정하고, 각 구간에 포함된 리뷰 수를 계산합니다.
   - sum() 함수와 조건문(current_start <= review_date < current_end)을 사용하여 각 구간에 포함된 리뷰 날짜를 세어 리스트에 저장합니다.

조건문

- 각 날짜가 현재 계산 중인 구간에 포함되는지 확인하는 데 사용됩니다. 이 조건문은 리뷰 날짜가 특정 2 주 구간에 포함되는지를 판별합니다.
- if 조건문은 주로 날짜 형식이 올바른지, 리스트가 비어있는지 등을 확인하는 데 사용됩니다.

반복문

- 모든 리뷰 날짜를 순회하며 각 날짜가 속하는 구간에 포함되는지 확인합니다. while 반복문을 사용하여 2 주 단위로 구간을 이동하며, for 반복문을 사용하여 각 구간 내의 리뷰 수를 세어줍니다.

4. 계산기 개발 과정

a. 계획 후 실제 개발 과정을 기록

1. input_dates 함수

사용자로부터 리뷰 날짜, 시작 날짜, 끝 날짜를 입력 받는 함수입니다.

- input() 함수로 입력을 받고, 리뷰 날짜는 쉼표로 구분하여 리스트로 변환합니다.
- 반환 값: review_dates (리뷰 날짜 리스트), start_date (시작 날짜), end_date (끝 날짜).

```
def input_dates():
    review_dates = input("리뷰 날짜를 입력하세요 (YYYY-MM-DD 형식, 쉼표로 구분): ").split(',')
    start_date = input("시작 날짜를 입력하세요 (YYYY-MM-DD 형식): ")
    end_date = input("끝 날짜를 입력하세요 (YYYY-MM-DD 형식): ")
    return review_dates, start_date, end_date
```

2. get_review_counts 함수

2 주 간격으로 리뷰 수를 계산하는 함수입니다.

- parse_date(date_str): 날짜 문자열을 datetime 객체로 변환하는 내부 함수입니다.
- try-except 블록: 날짜 형식 오류를 처리합니다.
- while 반복문: 시작 날짜부터 끝 날짜까지 2 주 단위로 구간을 설정하고 리뷰 수를 계산합니다.
- sum()과 조건문: 각 구간에 포함된 리뷰 수를 계산합니다.

```python
def get_review_counts(review_dates, start_date, end_date):
    # 날짜 문자열을 datetime 객체로 변환하는 함수
    def parse_date(date_str):
        return datetime.strptime(date_str, '%Y-%m-%d')

    # 입력된 날짜들을 datetime 객체로 변환
    try:
        review_dates = [parse_date(date) for date in review_dates]
        start_date = parse_date(start_date)
        end_date = parse_date(end_date)
    except ValueError as e:
        print("잘못된 날짜 형식이 입력되었습니다. 날짜 형식은 YYYY-MM-DD이어야 합니다.")
        return

    # 결과를 저장할 리스트
    review_counts = []

    # 2주 간격으로 리뷰 수를 계산
    current_start = start_date
    while current_start < end_date:
        current_end = current_start + timedelta(days=14)
        count = sum(current_start <= review_date < current_end for review_date in review_dates)
        review_counts.append((current_start.strftime('%Y-%m-%d'), current_end.strftime('%Y-%m-%d'), count))
        current_start = current_end

    return review_counts
```

- parse_date(date_str):
    - 입력받은 날짜 문자열을 datetime 객체로 변환합니다.
    - datetime.strptime(date_str, '%Y-%m-%d')을 사용하여 변환합니다.
- try-except 블록:
    - 변환 과정에서 발생할 수 있는 ValueError 를 처리합니다.
    - 예외가 발생하면 사용자에게 잘못된 날짜 형식을 알립니다.
- while 반복문:
    - current_start 를 start_date 로 초기화합니다.
    - current_start 가 end_date 보다 작을 때까지 반복합니다.
    - current_end 를 current_start 에 14 일을 더한 값으로 설정합니다.
    - count 는 current_start 부터 current_end 까지의 리뷰 수를 계산합니다.
    - 계산된 리뷰 수를 review_counts 리스트에 추가합니다.
    - current_start 를 current_end 로 업데이트하여 다음 구간으로 이동합니다.

3. main 함수 - 전체 프로그램의 메인 함수입니다.

- input_dates()를 호출하여 사용자 입력을 받습니다.
- get_review_counts()를 호출하여 2 주 간격 리뷰 수를 계산합니다.

```
def main():
    review_dates, start_date, end_date = input_dates()
    review_counts = get_review_counts(review_dates, start_date, end_date)
    if review_counts:
        print("2주 간격 리뷰 수:")
        for period_start, period_end, count in review_counts:
            print(f"{period_start} ~ {period_end}: {count}개")

if __name__ == "__main__":
    main()
```

- input_dates()를 호출하여 리뷰 날짜 리스트, 시작 날짜, 끝 날짜를 입력받습니다.
- get_review_counts()를 호출하여 2 주 간격 리뷰 수를 계산합니다.
- 결과가 있으면 각 구간의 리뷰 수를 출력합니다.

b. 에러 발생 지점

1. **잘못된 날짜 형식 입력**
2. **리뷰 날짜 리스트에 유효하지 않은 날짜 값이 포함된 경우**
3. **시작 날짜와 끝 날짜의 순서가 올바르지 않은 경우**
4. **빈 리뷰 날짜 리스트**
5. **날짜 범위에 포함된 리뷰가 없는 경우**

에러 발생 지점 및 디버깅 방법

1. 잘못된 날짜 형식 입력

디버깅 방법: parse_date 함수에서 날짜 변환 시 ValueError 가 발생하는지 확인합니다. 날짜 형식이 잘못되었을 경우, 예외를 발생시키고 문제의 날짜 형식을 출력하여 사용자가 입력한 날짜 형식의 오류를 파악할 수 있도록 합니다.

2. 리뷰 날짜 리스트에 유효하지 않은 날짜 값이 포함된 경우

디버깅 방법: get_review_counts 함수에서 review_dates 리스트의 각 날짜를 변환할 때 ValueError 가 발생하는지 확인합니다. 리스트 내 각 날짜를 변환하는 과정에서 예외를 처리하고, 잘못된 날짜 형식이 포함된 경우 예외를 발생시켜 문제를 추적합니다.

3. 시작 날짜와 끝 날짜의 순서가 올바르지 않은 경우

디버깅 방법: get_review_counts 함수에서 start_date 와 end_date 를 비교하여 시작 날짜가 끝 날짜보다 이후인지 확인합니다. 시작 날짜가 끝 날짜보다 이후일 경우 오류 메시지를 출력하고 종료하여 문제를 파악할 수 있도록 합니다.

4. 빈 리뷰 날짜 리스트

디버깅 방법: get_review_counts 함수에서 리뷰 날짜 리스트가 비어 있는지 확인합니다. 리스트가 비어 있을 경우 오류 메시지를 출력하고 종료하여 사용자가 입력한 데이터의 유효성을 검사합니다.

5. 날짜 범위에 포함된 리뷰가 없는 경우

디버깅 방법: get_review_counts 함수에서 2 주 단위로 리뷰 수를 계산할 때, 각 구간에 포함된 리뷰 수를 확인합니다. 각 구간별 리뷰 수를 출력하여 특정 구간에 리뷰가 없는 경우를 확인할 수 있습니다.

c. 해결책 적용 시 어떻게 변화

1. 잘못된 날짜 형식 입력

디버깅 결과: 사용자가 잘못된 날짜 형식을 입력했음을 확인할 수 있습니다. 예외 발생 시 입력된 날짜를 출력하여 문제를 파악합니다.

2. 리뷰 날짜 리스트에 유효하지 않은 날짜 값이 포함된 경우

디버깅 결과: 리뷰 날짜 리스트에 유효하지 않은 날짜가 포함되어 있음을 확인할 수 있습니다. 예외 발생 시 문제가 되는 날짜를 출력하여 파악합니다.

3. 시작 날짜와 끝 날짜의 순서가 올바르지 않은 경우

디버깅 결과: 시작 날짜가 끝 날짜보다 이후일 경우 오류 메시지를 출력하고 종료합니다. 문제의 원인을 쉽게 파악할 수 있습니다.

4. 빈 리뷰 날짜 리스트

디버깅 결과: 리뷰 날짜 리스트가 비어 있음을 확인할 수 있습니다. 리스트가 비어 있을 경우 오류 메시지를 출력하고 종료합니다.

5. 날짜 범위에 포함된 리뷰가 없는 경우

디버깅 결과: 각 구간별 리뷰 수를 출력하여 특정 구간에 리뷰가 없는 경우를 확인할 수 있습니다. 이를 통해 리뷰가 없는 구간에 대한 처리를 추가할 수 있습니다.

5. 계산기 개발 후기

a. 계산기 개발 후 느낀 점 설명

이번 학기에 수업을 들으면서 프로그래밍의 중요성을 깨닫게 되었고, 이를 실생활에 어떻게 적용할 수 있을지에 대해 많은 고민을 했습니다. 그러던 중 친구가 일하는 회사에서의 상황을 보며 파이썬으로 프로그램을 개발하게 됬습니다.

제 친구가 일하는 회사는 네이버플레이스를 통해 고객을 유입시키고, 상품을 구매한 후 고객들이 리뷰를 작성해주는데, 이 리뷰들을 관리하는 일이 상당히 번거로웠습니다. 여러 직원들이 매일같이 수많은 리뷰를 일일이 찾아보며 관리해야 했고, 이로 인해 다른 중요한 일들을 놓치거나, 심지어 잠을 줄여가며 업무를 처리하는 모습을 옆에서 지켜보았습니다. 이러한 비효율적인 업무 처리 방식을 개선할 수 있는 방법이 필요하다고 느꼈습니다.

그래서 저는 파이썬을 이용해 리뷰 관리를 자동화하는 계산기를 개발하기로 했습니다. 실제로 프로그램을 개발하면서 리뷰를 자동으로 수집하고, 특정 기간 동안의 리뷰 수를 계산해주는 기능을 구현했습니다.

이 계산기를 통해 친구의 회사에서는 고객 관리를 훨씬 더 효율적으로 할 수 있게 되었습니다. 직원들은 이제 리뷰를 일일이 찾아보지 않아도 되어 중요한 업무에 더 많은 시간을 투자할 수 있게 되었습니다.

개인적으로도 이번 경험을 통해 많은 것을 배울 수 있었습니다. 프로그래밍 수업에서 배운 내용을 실제로 적용해보는 과정에서 얻은 실전 경험이 많이 도움이 되었습니다. 또한, 문제를 해결하는 능력과 함께, 나중에 제가 비슷한 상황에 처했을 때 스스로 프로그램을 만들어 문제를 해결할 수 있다는 자신감을 얻었습니다.

앞으로도 파이썬을 주변에 홍보하고, 반복되는 작업을 단순화시키는 작업이 필요한 사람들에게 이 계산기를 추천하고 싶습니다. 파이썬은 정말 알기 전후가 확연하게 다른 도구이며, 이를 통해 많은 사람들이 더 효율적으로 업무를 처리할 수 있을 것입니다. 이번 경험을 통해 프로그래밍이 더 많은 사람들에게 소개되고 사용되면 좋겠습니다. 감사합니다.