SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

데이터분석 이론 및 실습

<알림>

본 강의는 삼성 청년 SW아카데미의 컨텐츠로
보안서약서에 의거하여
강의 내용을 어떠한 사유로도 임의로 복사,
촬영, 녹음, 복제, 보관, 전송하거나
허가 받지 않은 저장매체를
이용한 보관, 제3자에게 누설, 공개,
또는 사용하는 등의 행위를 금합니다.

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

9월9일 실습 및 과제 알고리즘

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

실습 알고리즘

실습1. 문자열 처리 및 반복문

• 1-1. 모음 제거

1st. 모음을 제거하기 위해 문자열의 각 문자를 확인할 필요가 있습니다. 이를 위해 for 반복문을 사용하여 문자열의 각 문자를 순회합니다.

2nd. 각 문자가 모음에 포함되어 있는지 검사하기 위해 vowels 변수에 모든 모음(대소문자 포함)을 저장합니다.

3rd. 문자가 vowels에 포함되지 않으면 새로운 문자열 result에 추가합니다.

4th. 반복문이 끝난 후, 모음이 제거된 결과 문자열을 반환합니다.

• 1-2. 단어 첫 글자 대문자 변환

1st. 주어진 문자열을 단어 단위로 나누기 위해 split() 메서드를 사용합니다. 이 메서드는 문자열을 공백을 기준으로 분리하여 리스트로 변환합니다.

2nd. 변환된 각 단어의 첫 글자를 대문자로 만들기 위해 capitalize() 메서드를 사용합니다.

3rd. capitalize()를 사용하여 변환된 단어들을 새로운 리스트에 추가합니다.

4th. 변환된 단어 리스트를 join() 메서드를 사용하여 공백을 기준으로 다시 결합하여 최종 문자열을 생성합니다.

5th. 결과 문자열을 반환합니다.

• 1-3. 특정 단어 등장 횟수 세기

1st. 문자열을 단어별로 나누기 위해 split() 메서드를 사용합니다. 이 메서드는 문자열을 공백을 기준으로 나누어 단어 리스트로 변환합니다.

2nd. for 반복문을 사용하여 리스트의 각 단어를 순회하며 주어진 단어와 일치하는지 확인합니다.

3rd. 각 단어가 주어진 단어와 일치할 때마다 카운트를 증가시킵니다.

4th. 반복문이 종료되면 최종적으로 계산된 카운트 값을 반환하여 등장 횟수를 제공합니다.

실습2. 리스트 처리 및 조건문

• 2-1. 짝수 선택

1st. 리스트의 각 요소를 하나씩 확인하기 위해 for 반복문을 사용합니다.

2nd. 각 요소가 짝수인지 확인하기 위해 if num % 2 == 0 조건문을 사용합니다.

3rd. 짝수인 요소는 새로운 리스트에 추가합니다.

4th. 모든 요소를 확인한 후, 짝수만 포함된 새로운 리스트를 반환합니다.

• 2-2. 3의 배수 제거

1st. 주어진 리스트에서 각 요소를 확인하기 위해 for 반복문을 사용합니다.

2nd. 각 요소가 3의 배수인지 확인하기 위해 if num % 3 != 0 조건문을 사용합니다.

3rd. 3의 배수가 아닌 요소는 새로운 리스트에 추가합니다.

4th. 모든 요소를 검사한 후, 3의 배수가 제거된 리스트를 반환합니다.

• 2-3. 요소 제곱

1st. 리스트의 각 요소를 제곱하기 위해 리스트 컴프리헨션을 사용합니다.

2nd. 각 요소에 제곱 연산을 적용합니다 (num ** 2).

3rd. 제곱된 값을 새로운 리스트에 추가하여 제곱된 요소로 구성된 리스트를 생성합니다.

4th. 완성된 리스트를 반환하여 결과를 확인합니다.

실습3. 리스트와 문자열 변환

• 3-1. 문자열을 리스트로 변환

1st. 문자열의 각 문자를 개별적으로 다루기 위해 list() 함수를 사용합니다.

2nd. list() 함수를 호출하여 문자열을 리스트로 변환합니다. 이 함수는 문자열의 각 문자를 리스트의 개별 요소로 변환해줍니다.

3rd. 변환된 리스트를 반환하여 결과를 확인합니다.

• 3-2. 리스트를 문자열로 변환

1st. 리스트의 요소들을 하나의 문자열로 결합하기 위해 join() 메서드를 사용합니다.

2nd. ''.join(lst)를 사용하여 리스트의 모든 요소를 빈 문자열 기준으로 합쳐 하나의 문자열을 생성합니다.

3rd. 생성된 문자열을 반환하여 결과를 확인합니다.

• 3-3. 문자열을 공백 기준으로 분리

1st. 문자열을 단어별로 나누기 위해 split() 메서드를 사용합니다.

2nd. split() 메서드를 호출하면 공백을 기준으로 문자열이 분리되어 각 단어가 리스트의 요소로 저장됩니다.

3rd. 분리된 단어 리스트를 반환하여 결과를 확인합니다.

• 4-1. 딕셔너리에서 키 삭제

1st. 특정 키를 딕셔너리에서 삭제하기 위해 del 키워드를 사용합니다.

2nd. 키가 딕셔너리에 존재하는지 확인하기 위해 if key in d 조건문을 사용합니다.

3rd. 키가 존재하면 del d[key]를 사용하여 해당 키를 삭제합니다.

4th. 삭제한 후의 딕셔너리를 반환하여 결과를 확인합니다.

• 4-2. 딕셔너리에 키-값 쌍 추가

1st. 새로운 키와 값을 딕셔너리에 추가하기 위해 딕셔너리의 키를 사용하여 값을 할당합니다.

2nd. d[key] = value를 사용하여 새로운 키-값 쌍을 추가합니다.

3rd. 추가된 후의 딕셔너리를 반환하여 결과를 확인합니다.

• 4-3. 특정 기준 이상의 값만 선택

1st. 딕셔너리의 요소를 필터링하기 위해 딕셔너리 컴프리헨션을 사용합니다.

2nd. for 반복문을 사용하여 각 키와 값을 순회하며 if value >= threshold 조건을 사용하여 기준 이상인 요소만 선택합니다.

3rd. 선택된 요소들을 새로운 딕셔너리에 추가하여 반환합니다.

실습5. 집합(Set) 조작

• 5-1. 중복 제거

1st. 리스트에서 중복된 요소를 제거하기 위해 set() 함수를 사용합니다.

2nd. set(lst)를 사용하여 리스트를 집합으로 변환합니다. 이 과정에서 중복된 요소들이 자동으로 제거됩니다.

3rd. 중복이 제거된 집합을 반환하여 결과를 확인합니다.

• 5-2. 교집합 계산

1st. 두 집합의 공통 요소를 찾기 위해 & 연산자를 사용합니다.

2nd. set1 & set2를 사용하여 두 집합의 교집합을 계산합니다. 교집합은 두 집합에서 모두 존재하는 요소들로 구성됩니다.

3rd. 계산된 교집합을 반환하여 결과를 확인합니다.

• 5-3. 합집합 계산

1st. 두 집합의 모든 요소를 포함하는 합집합을 계산하기 위해 | 연산자를 사용합니다.

2nd. set1 | set2를 사용하여 두 집합의 합집합을 계산합니다. 합집합은 두 집합의 모든 요소를 포함합니다.

3rd. 계산된 합집합을 반환하여 결과를 확인합니다.

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

과제 알고리즘

과제1. 문자열 리스트 심화 조작

• 1-1. 문자열 포맷팅

1st. 문자열을 단어 단위로 나누기 위해 split() 메서드를 사용합니다. 이 메서드는 문자열을 공백을 기준으로 분리하여 리스트로 변환합니다.

2nd. 각 단어의 첫 글자를 대문자로 변환하기 위해 capitalize() 메서드를 사용합니다. 이 메서드는 첫 글자만 대문자로 변환해줍니다.

3rd. 변환된 단어들을 하나의 문자열로 합치기 위해 join() 메서드를 사용합니다. 공백 없이 모든 단어를 결합하여 최종 문자열을 만듭니다.

4th. 결과 문자열을 반환하여 출력합니다.

• 1-2. 짝수 인덱스 요소 선택

1st. 리스트의 길이를 기준으로 인덱스를 생성하기 위해 range(len(lst))를 사용합니다. 이 메서드는 리스트의 인덱스를 0부터 끝까지 생성해줍니다.

2nd. for 반복문을 사용하여 리스트의 각 인덱스를 순회합니다.

3rd. 인덱스가 짝수인지 확인하기 위해 if i % 2 == 0 조건문을 사용합니다.

4th. 조건에 맞는 짝수 인덱스의 요소를 새로운 리스트에 추가합니다.

5th. 반복이 끝난 후, 완성된 리스트를 반환하여 결과를 출력합니다.

과제1. 문자열 리스트 심화 조작

• 1-3. 중복 제거 및 정렬

1st. set() 함수를 사용하여 리스트에서 중복된 요소를 제거합니다. 이 과정에서 중복된 요소들이 제거되고, 순서가 없는 집합 형태가 됩니다.

2nd. sorted() 함수를 사용하여 중복이 제거된 집합을 오름차순으로 정렬된 리스트로 변환합니다.

3rd. 정렬된 리스트를 반환하여 문제의 요구사항을 충족합니다

• 2-1. 리스트 평균 계산

1st. 리스트의 모든 요소를 더하기 위해 sum() 함수를 사용합니다. 이 함수는 리스트의 모든 숫자를 더해줍니다.

2nd. 리스트의 요소 개수를 구하기 위해 len() 함수를 사용합니다.

3rd. 합계와 개수를 나누어 평균을 계산합니다 (sum(lst) / len(lst)). 4th. 계산된 평균을 반환하여 출력합니다.

• 2-2. 소수 판별

1st. 숫자가 2보다 작으면 소수가 아니므로 False를 반환합니다.

2nd. 2부터 숫자의 제곱근까지 반복문을 사용하여 나누어떨어지는지 검사합니다.

3rd. 나누어떨어지면 소수가 아니므로 False를 반환하고, 나누어떨어지지 않으면 반복문이 종료된 후 True를 반환합니다.

• 2-3. 소수 필터링

1st. 리스트의 각 요소를 검사하기 위해 for 반복문을 사용합니다.

2nd. 각 요소가 소수인지 확인하기 위해 is_prime() 함수를 호출합니다.

3rd. 소수인 경우 새로운 리스트에 추가합니다.

4th. 소수로만 구성된 새로운 리스트를 반환하여 결과를 확인합니다.

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

내일 방송에서 만나요!

삼성 청년 SW 아카데미