<배열1 (Array 1)>

* 알고리즘
* 배열
* 버블 정렬 (Bubble Sort)
* 카운팅 정렬 (Counting Sort)
* 완전검색
* 그리디 (Greedy Algorithm)

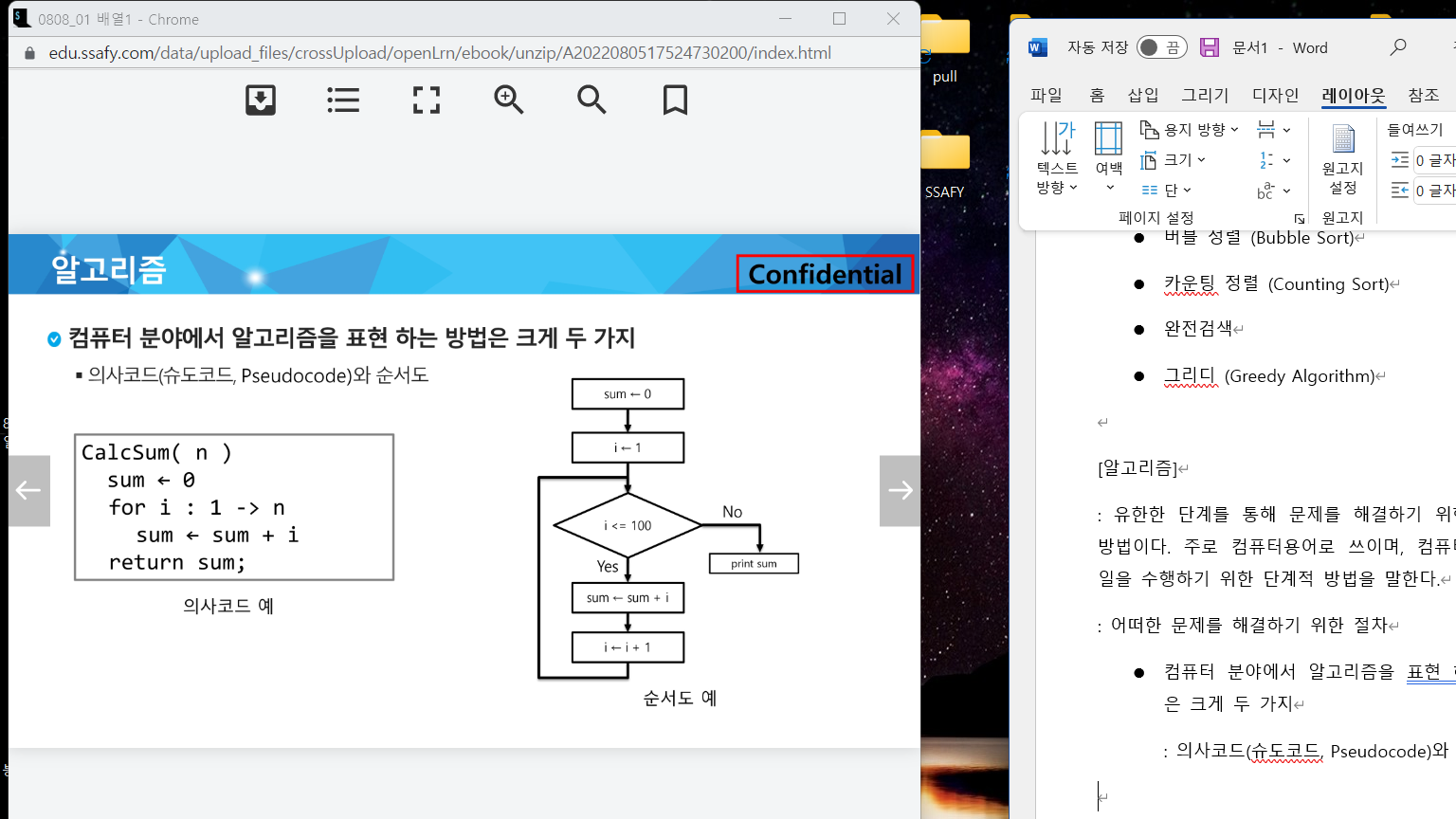
**[알고리즘]**

: **유한한 단계를 통해 문제를 해결하기 위한 절차나 방법**이다. 주로 컴퓨터용어로 쓰이며, 컴퓨터가 어떤 일을 수행하기 위한 단계적 방법을 말한다.

: 어떠한 문제를 해결하기 위한 절차

* 컴퓨터 분야에서 알고리즘을 표현하는 방법은 크게 두 가지

**: 의사코드**(슈도코드, Pseudocode)와 **순서도**



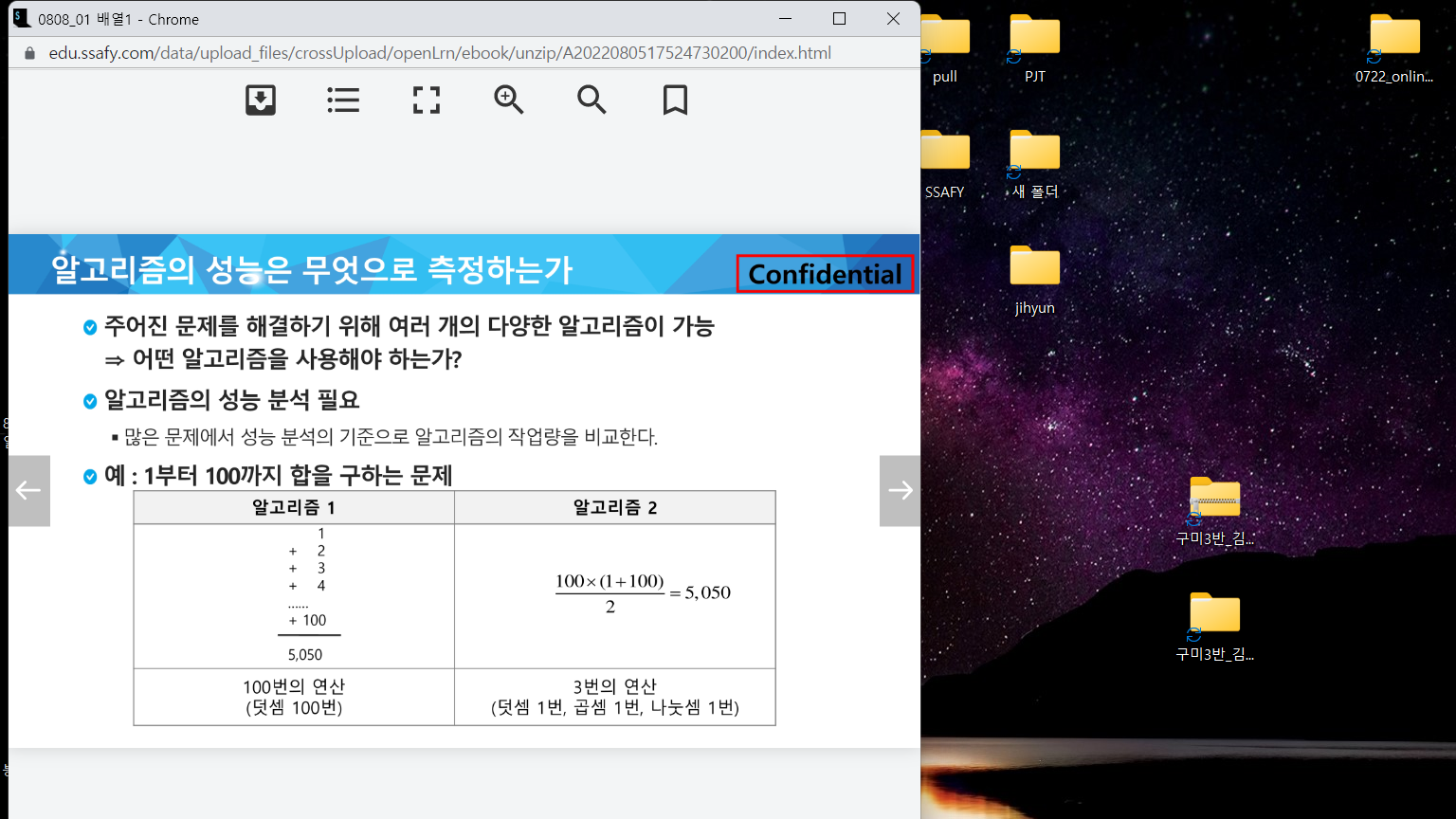
* APS 과정의 목표 중 하나는 보다 좋은 알고리즘을 이해하고 활용하는 것이다.
* 무엇이 좋은 알고리즘인가?

1. 정확성 : 얼마나 정확하게 동작하는가
2. 작업량 : 얼마나 적은 연산으로 원하는 결과를 얻어내는가 (실행시간)
3. 메모리 사용량 : 얼마나 적은 메모리를 사용하는가
4. 단순성 : 얼마나 단순한가
5. 최적정 : 더 이상 개선할 여지없이 최적화되었는가

* 알고리즘의 성능은 무엇으로 측정하는가
* 주어진 문제를 해결하기 위해 여러 개의 다양한 알고리즘이 가능 -> 어던 알고리즘을 사용해야 하는가?

1. 알고리즘의 성능 분석 필요

: 많은 문제에서 성능 분석의 기준으로 알고리즘의 작업량을 비교한다.)



1. 알고리즘의 작업량을 표현할 때 시간복잡도로 표현한다.

**시간 복잡도** (Time Complexity)

: 실제 걸리는 시간을 측정

: 실행되는 명령문의 개수를 계산

테이블이(가) 표시된 사진

자동 생성된 설명

1. 시간 복잡도 

**빅-오 표기법(**Big-Oh Notation)

: 시간 복잡도 함수 중에서 가장 큰 영향력을 주는 n에 대한 항만을 표시

: 계수(Coefficient)는 생략하여 표시

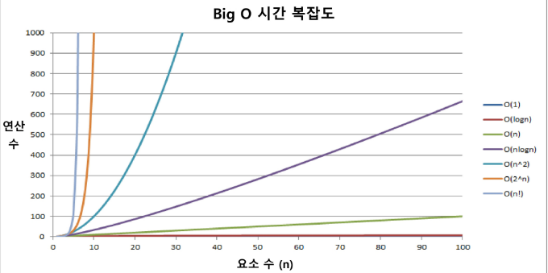
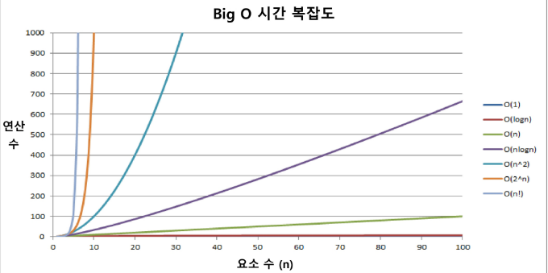
테이블이(가) 표시된 사진

자동 생성된 설명

Ex) n개의 데이터를 입력 받아 저장한 후 각 데이터에 1씩 증가시킨 후 각 데이터를 화면에 출력하는 알고리즘의 시간복잡도는 어덯게 되나? : O(n)

* 다양한 시간 복잡도의 비교

: 요소 수가 증가함에 따라 각기 다른 시간복잡도의 알고리즘은 아래와 같은 연산 수를 보인다.

****

**2진탐색**

**순차탐색**

**Quick,Merge sort**

**Bubble,선택정렬**

**NP문제**

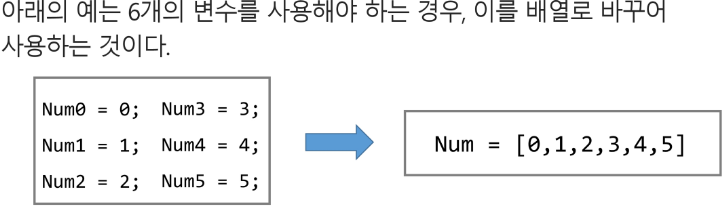
**P문제**

**(다항식)**

**상수시간**

**[배열]**

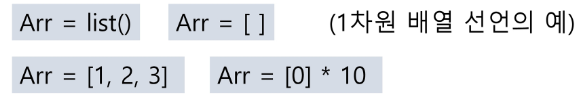
: **일정한 자료형의 변수들을 하나의 이름으로 열거하여 사용하는 자료구조**



* 배열의 필요성
* 프로그램 내에서 여러 개의 변수가 필요할 때, 일일이 다른 변수명을 이용하여 자료에 접근하는 것은 매우 비효율적일 수 있다.
* 배열을 사용하면 하나의 선언을 통해서 둘 이상의 변수를 선언할 수 있다.
* 단순히 다수의 변수 선언을 의미하는 것이 아니라, 다수의 변수로는 하기 힘든 작업을 배열을 활용해 쉽게 할 수 있다.

[1차원 배열]

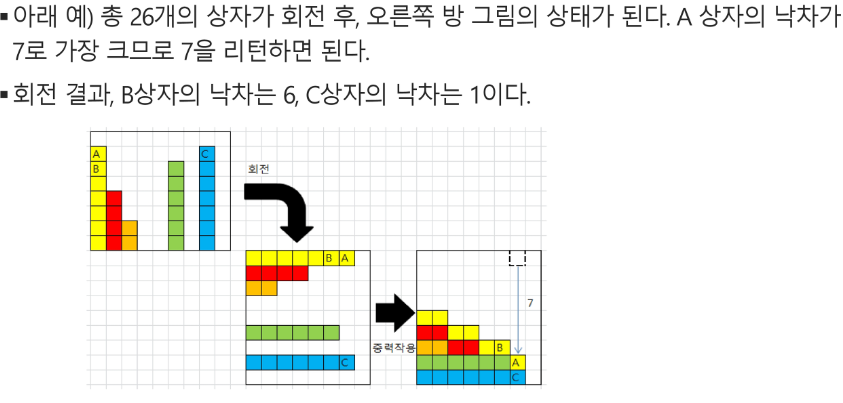
* 1차원 배열의 선언
* 별도의 선언 방법이 없으면 변수에 처음 값을 할당할 때 생성
* 이름 : 프로그램에서 사용할 배열의 이름



* 1차원 배열의 접근
* Arr[0] = 10 # ‘배열 Arr의 0번 원소에 10을 저장하라’
* Arr[idx] = 20 # ‘배열 Arr의 idx번 원소에 20을 저장하라’

텍스트이(가) 표시된 사진

자동 생성된 설명



**[정렬]**

: 2개 이상의 자료를 특정 기준에 의해 작은 값부터 큰 값(오름차순: ascending), 혹은 그 반대의 순서대로(내림차순: descending) 재배열하는 것

* 키

: 자료를 정렬하는 기준이 되는 특정 값

* 정렬 방식의 종류
* 버블 정렬 (Bubble Sort)
* 카운팅 정렬 (Counting Sort)
* 선택 정렬 (Selection Sort)
* 퀵 정렬 (Quick Sort)
* 삽입 정렬 (Insertion Sort)
* 병합 정렬 (Merge Sort)

**[버블 정렬]**

**파괴**

: **인접한 두 개의 원소를 비교하며 자리를 계속 교환**하는 방식

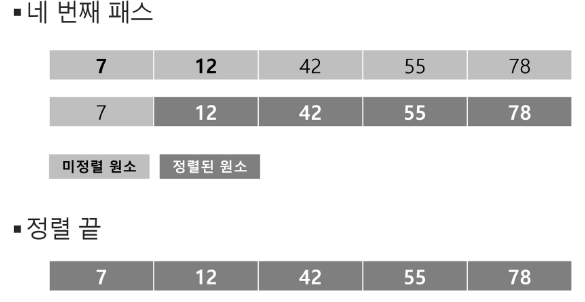
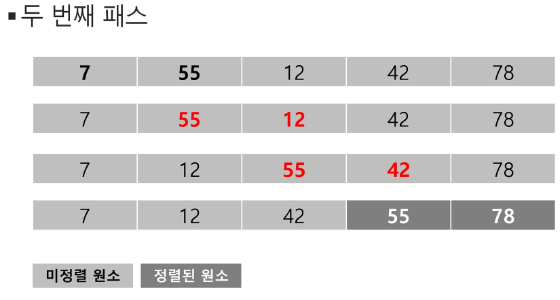
**<정렬 과정>**

* 첫 번째 원소부터 인접한 원소끼리 계속 자리를 교환하면서 맨 마지막 자리까지 이동한다.
* 한 단계가 끝나면 가장 큰 원소가 마지막 자리로 정렬된다.
* 교환하며 자리를 이동하는 모습이 물 위에 올라오는 거품 모양과 같다고 하여 버블 정렬이라고 한다.

**<시간 복잡도>**



테이블이(가) 표시된 사진

자동 생성된 설명

**구간의 시작과 끝을 정해줘야 함**

**끝은 1씩 줄어든다.**

**→가장 큰 수가 맨 오른쪽으로 감**

텍스트이(가) 표시된 사진

자동 생성된 설명

**a[j]와 a[j+1]을 비교하므로**

**N-1 부터 시작**

**[카운팅 정렬]**

**비파괴**

: 항목들의 **순서를 결정하기 위해** 집합에 **각 항목이 몇 개씩 있는지 세는 작업**을 하여, 선형 시간에 정렬하는 효율적인 알고리즘

**<제한 사항>**

* 정수나 정수로 표현할 수 있는 자료에 대해서만 적용 가능 (ex. 소수점 둘째자리 까지 나타난 수는 100을 곱해서 정렬 가능)

: 각 항목의 발생 횟수를 기록하기 위해, 정수 항목으로 인덱스 되는 카운트들의 배열을 사용하기 때문이다.

* 카운트들을 위한 충분한 공간을 할당하려면 집합 내의 가장 큰 정수를 알아야 한다.

**<시간 복잡도>**

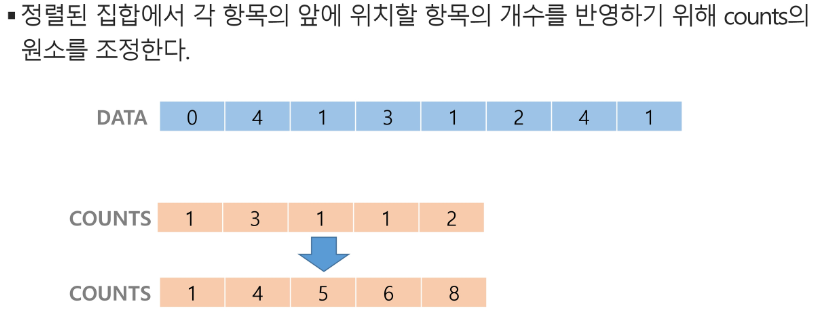




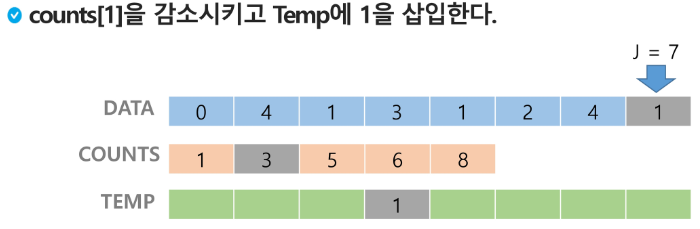
**COUNTS[ DATA[i] ]+=1**

**0 1 2 3 4**

**+ 0이상 4이하의 정수가 주어진다.**

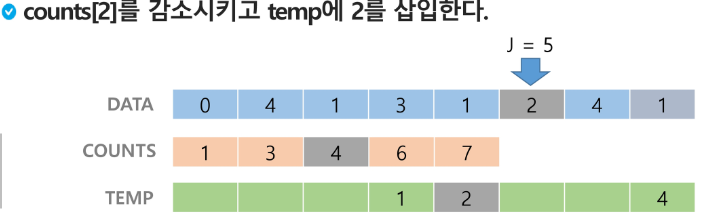
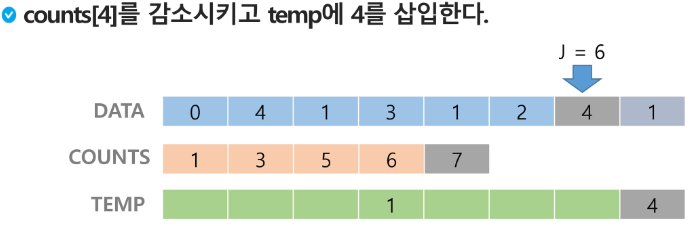


**누적된 개수의 합**



**0 1 2 3 4**

**1 2 3 4 5 6 7 8**



텍스트, 영수증, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**뒤에서부터 진행**

**인덱스로 변환**

**초기화**

**카운팅**

**누적**

테이블이(가) 표시된 사진

자동 생성된 설명

**[완전 검색]**

: 문제의 해법으로 생각할 수 있는 **모든 경우의 수를 나열해보고 확인**하는 기법이다.

: Brute-force 혹은 generate-and-test 기법이라고도 불리 운다.

: 모든 경우의 수를 테스트한 후, 최종 해법을 도출한다.

: 일반적으로 경우의 수가 상대적으로 작을 때 유영하다.

* 완전 검색으로 시작하라
* 모든 경우의 수를 생성하고 테스트하기 때문에 수행 속도는 느리지만, 해답을 찾아내지 못할 확률이 작다.
* 자격검정평가 등에서 주어진 문제를 풀 때, 우선 완전 검색(재귀)으로 접근하여 해답을 도출한 후, 성능 개선(가지치기)을 위해 다른 알고리즘을 사용하고 해답을 확인하는 것이 바람직.

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

순열 (Permutation)

: 서로 다른 것들 중 몇 개를 뽑아서 한 줄로 나열하는 것

* 서로 다른 n개 중 r개를 택하는 순열





텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

[탐욕(Greedy) 알고리즘]

: 최적해를 구하는 데 사용되는 근시안적인 방법

: 여러 경우 중 하나를 결정해야 할 때마다 그 순간에 최적이라고 생각되는 것을 선택해 나가는 방식으로 진행하여 최종적인 해답에 도달한다.

: 각 선택의 시점에서 이루어지는 결정은 지역적으로는 최적이지만, 그 선택들을 계속 수집하여 최종적인 해답을 만들었다고 하여, 그것이 최적이라는 보장은 없다.

: 일반적으로, **머릿속에 떠오르는 생각을 검증 없이 바로 구현하면 Greedy 접근**이 된다.

<알고리즘 동작 과정>

1. 해 선택: 현재 상태에서 부분 문제의 최적 해를 구한 뒤, 이를 부분해 집합(Solution Set)에 추가한다.
2. 실행 가능성 검사: 새로운 부분해 집합이 실행 가능한지를 확인한다. 곧, 문제의 제약 조건을 위반하지 않는지를 검사한다.
3. 해 검사: 새로운 부분해 집합이 문제의 해가 되는지를 확인한다. 아직 전체 문제의 해가 완성되지 않았다면 1)의 해 선택부터 다시 시작한다.

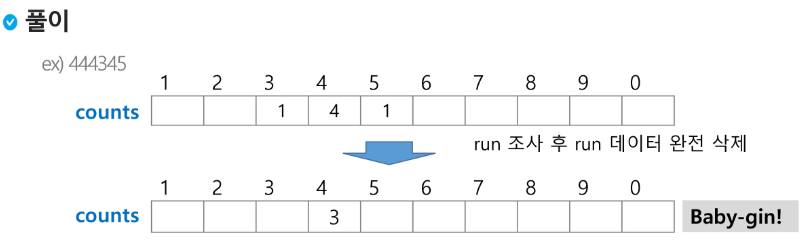
텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

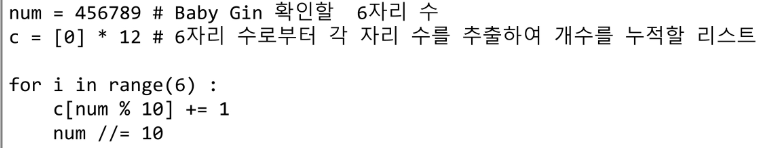
자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명



**9까지인데 12까지 만든 이유**

**: run 조사할 때 9뒤에 값이 없어서 오류 발생하므로 0을 넣은 인덱스 추가**

**: 아니면 if문에 조건문을 추가해서 7까지만 run 검색하도록 한다.**

텍스트이(가) 표시된 사진

자동 생성된 설명

**Ex. 123123 인 경우**

**Ex. 666666 인 경우**

**\*자주 실수하는 오답\***

텍스트, 실내이(가) 표시된 사진

자동 생성된 설명