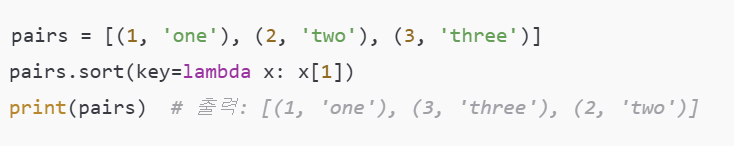
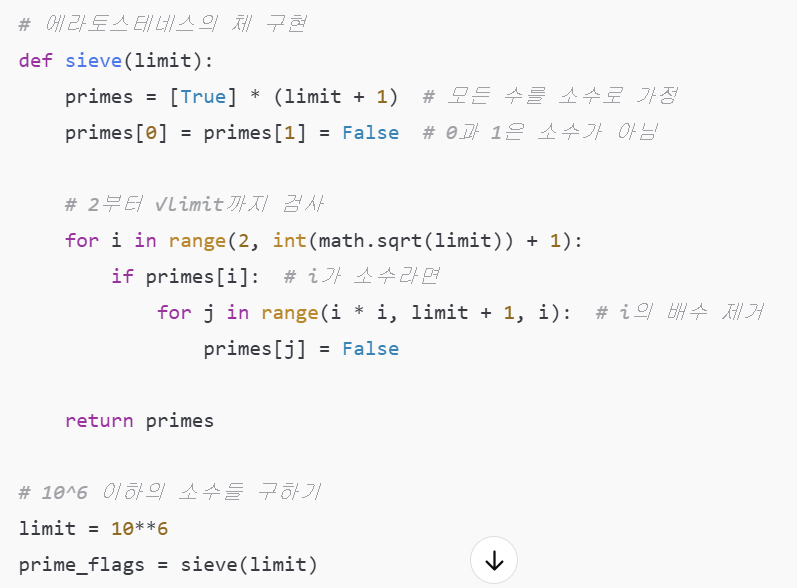
* // 表示除完后整数部分
* str1 = input().strip() 문자열(String)의 양쪽 끝에 있는 공백이나 특정 문자들을 제거
* abs() 绝对值函数
* \*\* 平方
* 用//和10，100把各位数提出来
* .append() list안에 추가
* "".join(results) 여기서 results는 list(빈 str에 list에 있는 글자를 추가, 숫자 불 가)나, 튜플 등 iterable한 것들을 추가
* ‘ ’.join() 요소를 띄어쓰기로 나눠서 추가하려면 공백필요, \n는 줄바꿈
* a = float('inf') 양의 무한대
* a = float(‘-inf’) 음의 무한대
* a = math.ceil() 소수점 이하를 무조건 올려서 정수로 반환 4.2 🡪 5
* a = math.floor() 내림
* ord(character) 문자 하나를 정수(유니코드 값)로 변환
* chr(integer) 유니코드 값을 문자로 변환
* if def\_name : 함수 리턴 값을 ==같이 false아니면 true로 정하면 간단
* dict1 = {‘i’ : 1} 사전안의 value를 key를 통해 나오게 함 dict1[‘i’]
* if s.isdigit(): 문자열 s가 숫자로만 구성되어 있는지 확인
* tv\_price.sort() 리스트를 오름차순으로 정렬, 문자면 사전식으로
* tv\_price.sort(reverse=True)리스트를 내림차순으로 정렬
* pairs.sort(key=lambda x: x[1]) 복잡한 기준으로 할 때



* if a in [list name]: 리스트 안에 a가 있는가에 대한 if문구
* remains\_seat.remove(2) 리스트 안의 2를 제거
* math.sqrt(x) x의 제곱근(실수 형태로 나옴)
* math.gcd(a, b) 최대 공약수
* abs(a \* b) // math.gcd(a, b) 최소 공배수
* 범위가 크지 않으면 리스트에 true, false형태로 소수를 미리 구해둘수 있음



* max\_num = sorted(numbers, key=lambda x: x\*10, reverse=True) 최대수 만들기

근데 위에 저거 하려면 numbers = list(map(str, input().split())) 숫자를 문자열로 받아야함

* money -= p\_weapon.pop(0) 리스트에서 첫번째 요소를 출력 및 제거
* f"{avg:.2f}" avg 변수의 값을 소수점 아래 둘째 자리까지
* copy.deepcopy() 독립적인 복사본 생성 (import copy해야함)
* trees = [tuple(map(int, input().split())) for \_ in range(n)] (xi, hi)이 형태로 저장

x, h = tress[i] 하면 xi, hi 가 각각 입력됨

* input = sys.stdin.read **모든 입력 데이터를 한 번에 읽음.**

data = input().split() 읽은 데이터를 공백 단위로 나누어 리스트로 저장

* from collections import deque duque 가져오기
* queue=deque([(x0,y0)]) **양쪽 끝에서 데이터를 추가하거나 제거할 수 있는 큐** **양쪽 끝에서 O(1)** 시간에 데이터 추가(append, appendleft) 및 제거(pop, popleft)가 가능

DP

**상태 정의**:

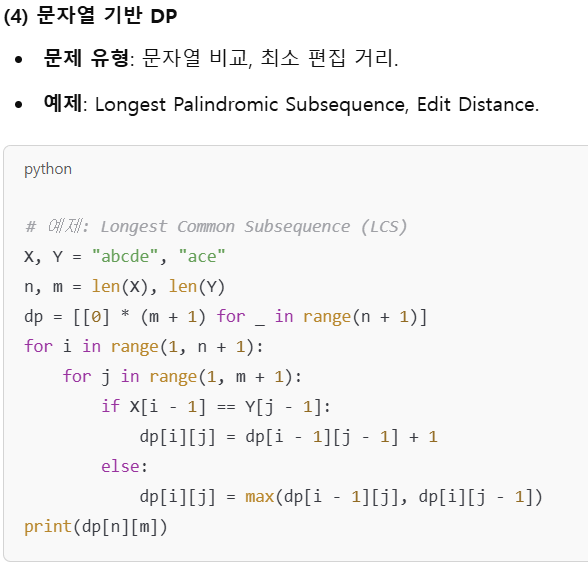
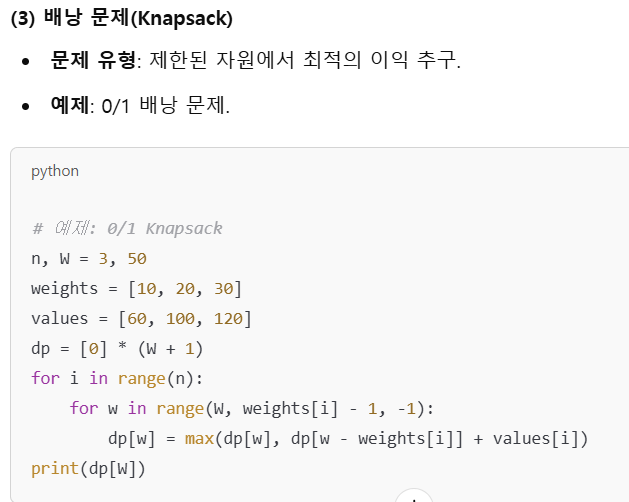
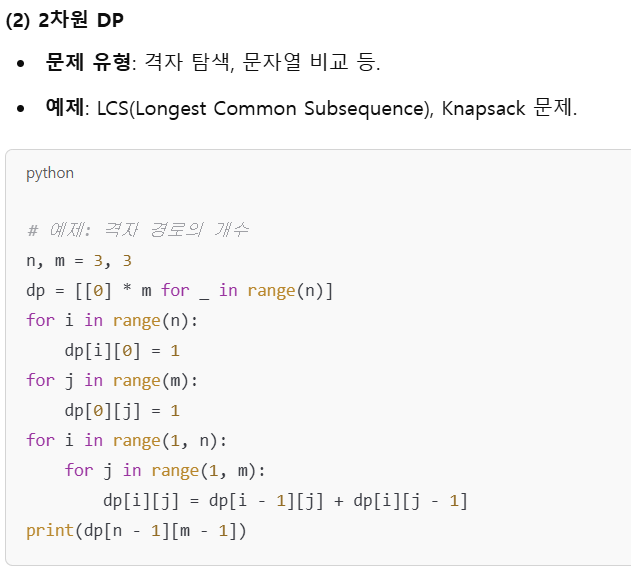
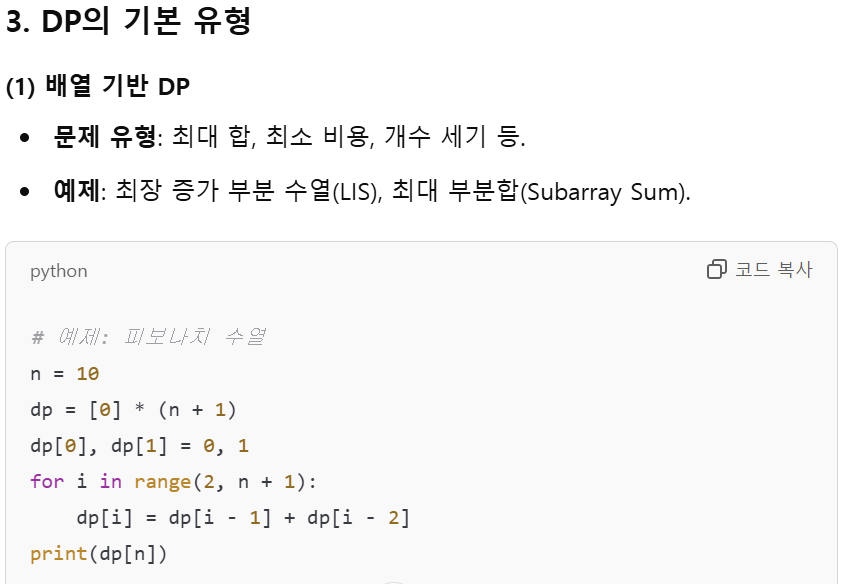
* DP 배열(또는 테이블)을 정의하고, 각 인덱스의 의미를 명확히 이해.
* 예: dp[i]는 \*\*"i까지 고려했을 때의 최적값"\*\*을 의미.

**점화식 세우기**:

* 현재 상태(dp[i])를 이전 상태(dp[i-1], dp[i-2] 등)와의 관계로 표현.
* 예: dp[i] = dp[i-1] + dp[i-2] (피보나치 수열).

**초기값 설정**:

* 테이블 초기화 후, 기저 조건(Base Case)을 정의
* 예: dp[0] = 1, dp[1] = 1 (피보나치 수열의 초기값)



Df 예제:

1.땅값 최대화문제

**def** **max\_value\_with\_removal**(values):

values = **list**(**map**(int, values.**split**(',')))

n = **len**(values)

**if** n == 1:

**return** values[0]

*# DP 배열 초기화*

dp\_no\_removal = [0] \* n

dp\_with\_removal = [0] \* n

*# 초기 조건*

dp\_no\_removal[0] = values[0]

dp\_with\_removal[0] = **float**('-inf') *# 첫 번째는 제외할 수 없으므로 -inf로 초기화*

*# DP 계산*

**for** i **in** **range**(1, n):

*# 아무것도 제외하지 않은 경우*

dp\_no\_removal[i] = **max**(values[i], dp\_no\_removal[i-1] + values[i])

*# 하나를 제외한 경우*

dp\_with\_removal[i] = **max**(dp\_with\_removal[i-1] + values[i], dp\_no\_removal[i-1])

*# 결과: 두 DP 배열의 최댓값 중 하나*

**return** **max**(**max**(dp\_no\_removal), **max**(dp\_with\_removal))

input\_data = **input**() *# "1,-5,0,3"*

**print**(**max\_value\_with\_removal**(input\_data))

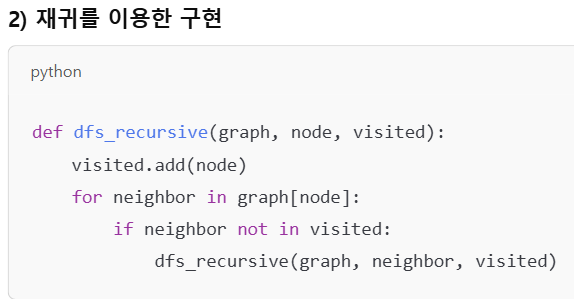
DFS

**그래프 표현**

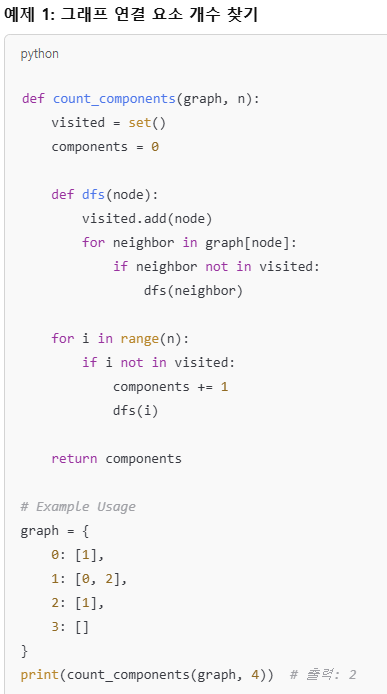
* 그래프를 인접 리스트 또는 인접 행렬로 표현.
* 일반적으로 **인접 리스트**가 더 메모리 효율적.

#### ****방문 여부 관리****

* 방문한 노드를 기록하여 무한 루프 방지.
* 리스트, 집합, 또는 딕셔너리 사용 가능.

dfs예제



**BFS**

#### ****그래프 표현****

* **인접 리스트**를 사용:



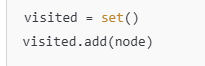
#### ****큐(Queue)를 활용한 구현****

* **Deque** 자료구조를 사용하여 효율적인 삽입과 삭제



#### ****방문 여부 관리****

* **집합(Set)** 또는 **리스트**로 방문 여부를 관리



#### ****최단 거리 계산****

* BFS는 가중치 없는 그래프에서 최단 거리를 자연스럽게 계산



bfs예제