```
(1)
-1,2:import random
-3: len(quest_list)
-4: {quest}
-5: !=
-6: idx
-7: idx+1
-8: from (Distress import *)
-9: print

(2)
-2번 # key값에 따옴표 빠져있다
-1,2번 # 기본적으로 인스턴스 메서드, 순서의 문제가 있지만 혼용자체는 가능
-3번
-2,4번 # 추상화, 자동 전달되는 인자(self)가 없을 뿐, 호출할 때에는 정의한 인자를 써야 한다
-1,3번 # 비교하는 경우, other를 써줘야 한다, __del__

(3)
-10
-True
-False
-True
-True
-111
-11
-135
-'10101111111'
- 1407 # print(int('10101111111', 2))

(4)
장엽님 사람들한테 장엽님 좋아하는 사람 손 들라 했더니 지구가 성게가 된 거 아세요?
한님 정말 멋지네요.
서준님 나 서른 살인데 동년배들 다 서준님 좋아한다.
칭찬 끝!
# --
누구세요?
# --
칭찬 끝!

# 입력받은 인자가 sparta리스트 안에 있으면, 칭찬문구를 출력하는 함수이다
# 또한, 재귀적 요소가 들어가 있는데, 결과적으로 처음 입력받은 모든 인자를 순회하게 된다
# 인자가 리스트에 없으면 : '누구세요?'를 리턴한다
# 처음에 any(names)가 값이 0이 나오면(False), 즉 비어있으면 '칭찬 끝'을 리턴한다

(5)
-3번: 두 수의 차이가 1e-20 이하면 같다고 볼 수 있어
# 1e-10이다...쥬륵..
-2번: dictionary는 다른 container로 형변환할 수 없어
# ★list로 형변환 가능하다. 단, key값만 추출 가능하다
# ★추가 : str / list, tuple, set는 key만 가능 / 나머지는 형변환 x
-1번: for - else 문에서 else는 반복문이 break 문으로 종료될 때 실행돼
# break에 안 걸리고, 끝까지 반복이 실행된다면 else문이 출력된다.
```

```
-1번: dictionary는 불변형이고 비시퀀스형이야.
# 딕셔너리는 순서가 없는 비시퀀스형이지만, 요소를 수정할 수 있다. 즉, 가변형이다.
# 예로 들어서, 키를 이용해서 값을 바꿀 수 있다.
-4번: 반복 제어로 break를 쓰면 이후의 코드를 수행하지 않고 다음 요소로 넘어가.
# ★ㄴㄴ. continue임. break는 반복문 자체를 종료하는 역할이다.
# ★continue는 해당 경우만 넘기는 것이지, 반복문 자체를 종료하지 않는다

(6)
-1: self.name = name
-2: {self.name} # 아 {} 빼먹었다. f스트링이니까..
-3: @classmethod
-4: Practice, Eat
-5: super()
-6: self.age = age
-7: __str__(self)
-8: 메서드 오버라이딩
-9: 인스턴스
-10: Play.coding # + ssafy.coding도 가능! -> 인스턴스도 클래스 변수에 접근 가능!
-11: 80 # 이 아니네.. 에러
# ssafy.age는 80이 출력되지만, age()이기에.. 호출되지 않는다!

(7)
- 임의 정밀도 산술
- 식별자
- 표현식
- 문장
- 모듈
- 추상화로 인해 코드 작성이 용이하다 / 간편하다 / 유지,보수가 용이하다
- 메서드
- @property

(8)
-4번: Private Member는 하위 클래스에 메서드 오버라이딩을 허용하지만, 직접 호출은 불가능해.
# 하위 클래스에 메서드 오버라이딩이 허용되지 않는다!
-2, 3번: # 숫자는 허용x, E도 가능하다
-4번: .pop 메서드는 문자열이랑 list에서 사용 가능해요. 정해진 위치에 있는 값을 삭제하죠.
# str에서는 사용 불가능하다. 문자열은 불변하기에, 재할당은 가능해도, 일부 수정은 불가
-1, 4번:
# 최근이 아니라, 상속된 순서에 따라 우선순위가 결정된다
# @setter가 아니라, @변수.setter이다. (낚임,,)
-1번: # isinstance(인스턴스, 클래스)이다.

(9)
-한번 만들어 두면, 재사용 가능하다 / + 가독성, 유지 보수 용이
-LEGB rule
-filter함수 안의 조건문에서 True인 경우만 반환한다
-zip은 복수의 iterable 객체를 모아 tuple의 모음으로 구성된 zip object를 반환한다
# 또한, list등, 형변환을 해야 결과값을 볼 수 있다
-enumerate: 인덱스와 값을 튜플로 묶어서 반환한다
-endswith함수
-discard함수

(10)
- 전반적으로, 변수명들이 구분이 잘 안된다.
# 변수명, 변수명에 줄임말을 많이 써 가독성이 좋지 않다.
- return을 제외하고, 들여쓰기가 잘못되었다.
- position_list = list(position) # 굳이 바꿔줄 필요 없다.
# x, y형태이므로, 튜플을 할당할 수 있다
```

```
- print가 아닌, return을 써야한다. # (None이 리턴되기 때문)
- 종료조건이 잘못되었다.
# while i <len(name_list)가 되어야한다.
# i=4일때 까지는 반복문이 잘 작동하지만, i=5일때 인덱스에러가 발생한다.
```