1. Grade 클래스 완성하기

학생의 성적을 입력해 평균을 출력하는 Grade 클래스를 생성하십시오.

클래스명	class Grade
입력	Name:str, grade_list: List(int) = int
출력	과목 평균 점수: float
조건	- def __init__(name: str, : grade_list: List(int) = int) - def __len__( ) -> len(grade_list) : 점수 리스트 길이 리턴 - def get_average() -> float :평균 점수 리턴

예시 결과	`p = Grade("박융합",[45,66,93])`
	`print("{0}의 {1}과목 평균 점수: {2}".format(p.name, len(p),p.get_average()))`
	박융합의 3과목 평균 점수: 68.0

2. Account 클래스 완성하기

성균은행에 개설된 계좌의 정보를 확인하고, 입출금을 하려고 합니다. 예금주, 생년월일, 계좌번호, 잔고가 저장된 Account 클래스를 설정하고, 예시 결과와 같이 출력될 수 있도록 메소드를 설정하십시오. 생성자에는 예금주와 생년월일만 입력 받습니다. 계좌번호는 3 자리-2 자리-6 자리 형태로 랜덤하게 생성되고, 잔고도 0~1000 원 안에서 랜덤하게 생성됩니다.

클래스명	Account
입력	Name:str, birthday:str
출력	계좌 개설 정보 (예시 결과 참고)
조건	- 보안문제로 외부에서 내부의 정보를 수정할 수 없도록 변수를 설정해야함 - 계좌번호와 잔고의 경우 표시된 결과와 다를 수 있음 - class Account : 계좌 개설 클래스 - def __init__(name: str, birthday: str) - def __str__( ) -> 계좌정보(예금주, 생년월일, 계좌번호, 잔고) - def show_balance( ) -> 잔고:int : 잔고 확인 함수 - def deposit(amount:int) :입금 함수 - def withdraw(amount:int) 출금 함수 - 필요시 random() 함수 사용 가능 - 안전코딩 : 입금은 0 원 미만일 수 없고, 출금은 잔고 이상일 수 없음

예시 결과	`person = Account("김성균", "990218")`
	`print(person)`
	`print("#####")`
	`person.deposit(100)`
	`person.deposit(-1)`
	`person.show_balance()`
	`person.withdraw(500)`
	`person.withdraw(0)`
	`person.show_balance()`
	>>>
	이름: 김성균
	생년월일: 990218
	계좌번호: 789-21-591520
	잔고: 2504
	#####
	100원 입금
	입금불가
	잔고: 2604
	500원 출금
	출금불가
	잔고: 2104