

1. 은행계좌 만들기

은행계좌 객체를 만드는 BankAccount 클래스를 작성해보자.

은행계좌 객체는 소유자이름과 예금잔고 정보를 갖고 있어야 한다.

- 속성변수

- name: 소유자 이름 (문자열)
- balance: 은행 잔고, 단위: 원 (정수), 반드시 0이상






은행계좌 객체를 생성하면서 이름과 초기 예금금액을 인수로 받아 설정하도록 __init__ 메소드를 추가하자. 그런데 balance 인수는 생략할 수 있고, 생략한 경우에 0으로 설정한다. balance 인수는 절대 음수가 되지 않아야 한다. 음수 인수가 들어오는 경우 모두 0으로 설정하도록 해야 한다.

그 후, __str__ 메소드를 추가해 소유자 이름과 잔고를 출력하는 메소드를 만들도록 하라. 뿐만 아니라, BankAccount 클래스에 입출금 기능을 하는 메소드를 다음의 요구사항에 맞추어 작성하여 추가하자.

- 메소드

- show_balance(): 자신의 balance를 실행창에 프린트한다.
- deposit(amount): amount가 0 이상의 정수이면, 자신의 balance를 amount 만큼 증가시키고 실행창에 프린트한다. amount가 음수이면, 입금이 불가하다는 메시지를 실행창에 프린트한다.
- withdraw(amount): amount가 음수가 아니고 balance를 초과하지 않으면, 자신의 balance를 amount 만큼 감소시키고 실행창에 프린트한다. 그렇지 않으면, 입금이 불가하다는 메시지를 실행창에 프린트한다

BankAccount의 보안을 확보하려면 self.name과 self.balance 속성변수를 외부에서 값을 고칠 수 없도록 해야한다. 이 두 속성변수를 비공개 속성변수로 고쳐서 외부에서 수정불가능하게 만들자. 그러면 객체를 생성하면서 만들어진 self.name 속성변수는 영원히 수정불가능해진다. 그리고 self.balance 속성변수 값은 deposit과 withdraw 메소드 호출을 통해서만 수정이 가능하다. 다른 방법은 없다. 이 두 속성변수를 비공개로 만들어 코드를 수정한 후, 다음의 실행사례와 동일한 결과가 나오는지 확인하자.

```
1 class BankAccount:
2     def __init__(self, name, balance):
3         
4
5
6
7     def __str__(self):
8         
9
10
11     def show_balance(self):
12         
13
14
15     def deposit(self, amount):
16         
17
18
19
20
21
22     def withdraw(self, amount):
23         
24
25
26
27
28
```