Smart Contract: Decorator in Python

Background
In banking, variation of a financial product, says deposit account, is commonly being offered to a depending on market segment of a customer whom he/she is belong to. Though specific features vary from variation to the next, most of the functionalities are still shared and tied to a product, says our deposit account.

In SW context, we would like smart contracts, our financial products, to be in the form of template. A specific variation can be easily configured while certain parameters are shared between the variations within the product group. This will help to accelerate development and enable adaptation of best practices.

Tasks
Take a financial product, says deposit account:

- Create a class/template for a simplified deposit account with attributes like account number, currency, interest rate, etc.
- Use *Decorator* design pattern, create an enabler to allow interests to be accrued differently depending on the product variation.
    - For example, at the end of day, month or year.
    - We can see the same philosophy will certainly apply if we are to do lending products.
- Create test cases
- Create a unit test, make use of a Python Unit Test framework, and run through it with test cases that you created.

Submission
Please submit your codes and test results through GitHub and share the URL.

Miscellaneous

- Python is to be used and its coding convention to be followed.
- Use Design Pattern when applicable and try not to hard code the variables.