Домашнее задание

Дисциплина	Программирование на Python
Тема	Классы и объекты (ООП)
Форма проверки	**Проверяет преподаватель**
Имя преподавателя	Олег Булыгин
Время выполнения	2 часа
Цель задания	Отработать навыки работы с классами в Python
Инструменты для выполнения ДЗ	Jupyter Notebook или Google Colab
Правила приёма работы	Прикрепите ссылку на выполненное задание в Google Colab или GitHub, если вы использовали Jupyter Notebook. Важно: убедитесь в том, что по ссылке есть доступ в Google Colab (иногда в Colab нет доступа для другого логина)
Критерии оценки	**Максимальное количество баллов за задание — 10 баллов** **Задание считается выполненным, если:** - прикреплена ссылка на файл с выполненным заданием; - доступ к файлу открыт; - код даёт правильный ответ к задаче. **Задание не выполнено, если:** - файл с заданием не прикреплён или отсутствует доступ по ссылке; - код выдаёт ошибку или даёт неправильный ответ
Дедлайн	Срок сдачи — 27.12.2025

Описание задания

Перед выполнением задания установите Jupyter Notebook или используйте Google Colab.

Задание

Ваша задача — реализовать класс Account, который моделирует поведение банковского счёта. Этот класс должен не только выполнять базовые операции, но и вести детальный учёт всех действий, а также предоставлять аналитику по истории операций.

Этап 1. Реализация базового класса `Account`

Класс должен быть инициализирован с **параметрами**:

- `account_holder` (str): имя владельца счёта;
- `balance` (float, по умолчанию 0): начальный баланс счёта, не может быть отрицательным.

Атрибуты:

- `holder`: хранит имя владельца;
- `_balance`: приватный атрибут для хранения текущего баланса;
- `operations_history`: список или другая структура для хранения истории операций.

Важно: каждая операция должна храниться не просто как число, а как структурированная информация, например, словарь или кортеж. Минимальный набор данных для операции: `тип операции` ('deposit' или 'withdraw'), `сумма`, `дата и время операции`, `текущий баланс после операции`, `статус` ('success' или 'fail').

Этап 2. Реалирзация методов

1. `__init__(self, account_holder, balance=0)`: конструктор;
2. `deposit(self, amount)`: метод для пополнения счёта:
   - принимает сумму (должна быть положительной);
   - в случае успеха обновляет баланс и добавляет запись в историю операций.
3. `withdraw(self, amount)`: метод для снятия средств:
   - принимает сумму (должна быть положительной);
   - проверяет, достаточно ли средств на счёте, если нет — операция не проходит, но ее попытка с статусом 'fail' все равно фиксируется в истории;
   - в случае успеха обновляет баланс и добавляет запись.
4. `get_balance(self)`: метод, который возвращает текущий баланс.
5. `get_history(self)`: метод, который возвращает историю операций.

Важно: продумайте, в каком формате его вернуть. Для работы с датой и временем используйте модуль `datetime`. Получить текущее время можно с помощью `datetime.now()`.

Этап 3. Реализация наследования. Задача повышенной сложности, выполнение по желанию

Создайте класс `CreditAccount(Account)`, который наследует всю функциональность класса `Account` и добавляет новую.

Особенности кредитного счёта:

1. При инициализации принимает дополнительный параметр `credit_limit` (кредитный лимит). Баланс такого счета может быть отрицательным, но не ниже значения `-credit_limit`.
2. По запросу показывает, сколько кредитных средств еще доступно (`текущий баланс + кредитный лимит`).
3. В историю операций добавляется информацию о том, были ли использованы кредитные средства в данной операции.

Критерии оценивания

Критерий	Баллы	Описание
Корректность ООП	3	Класс корректно инициализируется, используются инкапсуляция (приватные атрибуты, геттеры), методы `deposit` и `withdraw` работают без ошибок, логически верно обрабатывают неверные входные данные (отрицательные суммы)
История операций	3	История сохраняется в структурированном виде. Метод `get_history` возвращает информацию в удобном формате.
Работа с датами	2	В каждой операции корректно фиксируется дата и время ее проведения с помощью модуля `datetime`.
Наследование	2	Корректно выполнена задача повышенной сложности
Чистота кода	-1 за каждое нарушение	Код откомментирован, читаемый, соответствует PEP8