Лабораторные работы 8

Присылать на почту daria.zubkova@spbpu.com

Дедлайн: 25.05.2025 до 23:59

Формат (выбирайте один из):

1. файл в формате *.py/*.ipynb, подписывать своей фамилией и именем
2. файл *.txt, подписывать своей фамилией и именем

Задание:

Выполнить тестовое покрытие через pytest или unittest. Количество тестов регламентируется на ваше усмотрение, но для простых задач не менее 5 тестов; для сложных задач не менее 10 тестов. Должны проверяться все функции, описанные в коде. Тесты должны проверять не только корректные случаи, но и случаи, если пользователь ошибся и ввел неправильные значения. Сами задачи были реализованы ранее в предыдущих лабораторных.

Простые задачи:

Задача 1 (Л/р 3, з2).

Напишите функцию с двумя параметрами: первый будет делиться на 10 целочисленным делением, а второй – на 5 обычным делением (округлить до 2 цифр после запятой).

На вход принимаются значения через запятую. На выходе выводятся значения через запятую.

Входные данные:

```
7,18
```

Выходные данные:

```
0,3.60
```

Задача 2 (Л/р 3, з4).

Напишите функцию arithmetic, принимающую 3 аргумента: первые 2 - числа, третий - операция, которая должна быть произведена над ними. Если третий аргумент +, сложить их; если -, то вычесть из первого второе; * — умножить; / — разделить (первое на второе обычным делением). В остальных случаях вернуть строку "Неизвестная операция".

Если делим на 0, то выводить «Деление на 0 запрещено».

На вход принимаются значения через запятую.

Входные данные:

```
10,4,*
```

Выходные данные:

```
40
```

Задача 3 (Л/р 4, з1).

Напишите программу, которая фильтрует список чисел, оставляя только те, которые делятся на 3 и возводят в квадрат каждое из оставшихся чисел, используя lambda-выражения и встроенные функции filter и map.

Входные данные:

```
1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

Выходные данные:

```
9, 36, 81
```

Сложные задачи:

Задача 4 (Л/р 5, з4).

Напишите программу, которая моделирует работу банковского счета с различными проверками. Должны быть реализованы классы и методы:

- Account — основной класс, представляющий банковский счет (по умолчанию начальная сумма = 0):
    - Методы:
    - deposit – пополнение счета на сумму. Сумма должна быть больше 0.
    - withdraw – снятие наличных. Сумма должна быть больше 0 и средств на счете должно быть достаточно.
    - transfer_c2a – перевод средств со счета. Сумма должна быть больше 0 и средств на счете должно быть достаточно.
    - transfer_a2c - перевод средств на счет. Сумма должна быть больше 0.
    - get_balance – возвращает текущий баланс счета.
    - show_transactions – возвращает список совершенных транзакций по счету.
- Transaction — хранит записи транзакций с суммой и типом (deposit, withdraw, transfer_c2a, transfer_a2c).

Пример:

```
account = Account(1000)  # Счет с ненулевой начальной суммой
account.deposit(500)      # Пополнение счета на 500
```

```
account.withdraw(200)      # Снятие 200
print(account.get_balance())     # 1300
```

Задача 5 (Л/р 6, з3).

Реализуйте класс `User` с защищенными данными профиля:

- Приватные атрибуты:
  - `__username` (логин, минимум 5 символов),
  - `__password` (пароль, минимум 8 символов).
- Публичные методы:
  - `update_password(old_pass, new_pass)` (смена пароля при условии правильного старого).
  - `show_info()` (возвращает логин и пароль, но скрывает пароль: логин: `user123`, пароль: `********`)
- Ошибки:
  - Неверный старый пароль -> `ValueError("Неверный пароль")`
  - Слабый новый пароль (меньше 8 символов) -> `ValueError("Пароль слишком короткий")`
  - Слабый логин (меньше 5 символов) -> `ValueError("Логин слишком короткий")`
- Требования:
  - Инкапсуляция через приватные атрибуты
  - Валидация входных данных в методах
  - Запретить прямой доступ к `__username`, `__password`