**Задание: "Разработка системы мониторинга ресурсов компьютера с визуализацией"**

**Цель задания**

Разработать программу на Python для мониторинга системных ресурсов в реальном времени с выводом в консоль и генерацией графических отчетов.

**Технические требования**

**1. Архитектура системы**

Система должна состоять из следующих основных компонентов:

**Дескриптор валидации**

- **Класс**: PercentageValidator
- **Назначение**: Обеспечивает валидацию процентных значений
- **Требования**:
  - Проверять, что значение является числом (int или float)
  - Гарантировать, что значение находится в диапазоне 0-100
  - Автоматически округлять значения до 2 знаков после запятой
  - Использовать протокол дескрипторов Python (__get__, __set__, __set_name__)

**Абстрактная система метрик**

- **Абстрактный класс**: ResourceMetric
- **Назначение**: Определяет общий интерфейс для всех метрик системы
- **Требования**:
  - Содержать дескриптор current_usage для хранения текущего значения
  - Иметь конструктор, принимающий имя метрики
  - Объявлять абстрактные методы:

        - update() - для обновления данных метрики
        - Свойство value - для предоставления доступа к текущему значению

**Конкретные реализации метрик**

Реализовать три класса метрик, наследуемых от ResourceMetric:

1. **CPUMetric**
    - Собирает данные о загрузке процессора с помощью psutil
    - Использует psutil.cpu_percent()
2. **RAMMetric**
    - Мониторит использование оперативной памяти
    - Использует psutil.virtual_memory()
3. **DiskMetric**
    - Отслеживает использование дискового пространства
    - Должен принимать путь к диску в конструкторе
    - Использует psutil.disk_usage()

**Требования ко всем классам метрик**:

- Реализовать обязательные абстрактные методы
- Использовать дескриптор для установки значений
- Предоставлять доступ к данным через свойство value

**Система визуализации**

Реализовать два класса для разных типов отображения:

1. **ConsoleVisualizer**
    - **Статический метод**: draw_bar(percentage, label, width=30)
    - Формирует ASCII-график в виде столбца
    - Использует символы '█' для заполненной части и '▒' для пустой
    - Форматирует вывод с выравниванием
2. **ImageVisualizer**
    - Создает графические отчеты с использованием библиотеки Pillow

    - **Метод**: create_report(metrics, filename="system_report.png")
    - Генерирует PNG-изображение с столбчатой диаграммой
    - Размещает на изображении:
        - Заголовок отчета
        - Цветные столбцы для каждой метрики
        - Подписи с названиями метрик и значениями
    - Использует разные цвета для различных типов метрик

**Управляющий класс**

- **Класс**: MonitoringManager
- **Назначение**: Координирует работу всей системы мониторинга

**Требования**:

- Содержать список отслеживаемых метрик
- Иметь методы:
    - update_all_metrics() - полиморфное обновление всех метрик
    - display_console() - вывод данных в консоль
    - run_console_monitor(interval=1) - запуск непрерывного мониторинга
    - generate_image_report() - генерация графического отчета

**2. Функциональные требования**

**Основной функционал:**

- Реализовать интерактивный консольный монитор с обновлением в реальном времени
- Очистка консоли при каждом обновлении для эффекта "живого" дисплея
- Корректная обработка прерывания (Ctrl+C) для graceful shutdown
- Генерация графических отчетов по требованию

**Интерфейс консольного вывода должен содержать:**

- Заголовок системы мониторинга
- ASCII-графики для каждой метрики с подписями

- Процентные значения использования
- Инструкцию по управлению программой

### 3. Дополнительные задания

#### Задание 3.1: Сетевая метрика

- Реализовать класс NetworkMetric, отслеживающий сетевую активность
- Использовать psutil.net_io_counters()
- Отображать скорость передачи данных (в Мбит/с или МБ/с)

#### Задание 3.2: Система логирования

- Реализовать запись данных в файл при каждой генерации отчета
- Формат лога: timestamp, значения всех метрик
- Использовать прямую запись в CSV-файл

### 4. Технические указания

#### Используемые библиотеки:

- **psutil** - для сбора системной информации
- **Pillow (PIL)** - для генерации графических отчетов
- Стандартные библиотеки Python: abc, time, os

### 6. Рекомендации по выполнению

1. Начните с реализации дескриптора валидации
2. Создайте абстрактный класс и конкретные реализации метрик
3. Реализуйте консольную визуализацию
4. Добавьте графическую визуализацию с Pillow
5. Интегрируйте все компоненты в управляющий класс
6. Протестируйте систему на разных платформах
7. Выполните дополнительные задания