Повні структуровані інструкції для проекту Z-Score бота

На основі аналізу наданих матеріалів і вимог, наведені нижче інструкції забезпечують цілісну структуру проекту, уникнення дублювань і відповідність найкращим практикам програмування на Python.

---

1. Загальні інструкції

1. Відповідай українською мовою.
2. Коментуй кожен блок коду, пояснюючи його логіку.
3. Документуй модулі та функції в README.md або у відповідних docstring.
4. Дотримуйся стандарту PEP 8 для стилю коду.
5. Використовуй зрозумілі назви для модулів, класів, функцій і змінних.

---

2. Робота з базою даних

- **Технологія:** SQLite для збереження історичних даних.
- **Структура даних:**
  - **Таблиця криптовалют:**
    `name`, `timestamp`, `price`.
  - **Таблиця обраних пар:**
    Включає `pair`, `zscore`, `cross_rate`, `max_zscore`, `max_cross_rate`, `dynamic_log` (історія змін), `entry_cross_rate`.
- **Функціональність:**
  - CRUD-операції для таблиць.
  - Перевірка наявності та автоматичне створення таблиць під час запуску.
  - Оновлення даних через API Binance.
- **Модуль:** `database/db_manager.py`

---

3. Модуль роботи з даними

- **Ініціалізація та актуалізація даних при першому запуску програми, або при паузі у роботі.**
- **Оновлення даних:**
  - Автоматичне оновлення кожні 15 хвилин.
  - Зберігання останніх 672 інтервалів (за місяць) на таймфреймі 15 хвилин.
  - Перевірка на некоректні дані перед обчисленням.

- **Розрахунки:**
  - **Z-Score:** Обчислення стандартного відхилення, середнього значення та Z-score.
  - **Синтетичний курс (Cross-Rate):** Вираховується для кожної пари.
  - **Фільтрація:** За кореляцією, бета-коефіцієнтом, перцентилями.
  - **Перцентили:** Розрахунок 10% і 90%.
- **Модулі:**
  - `data_processing/z_score_calculator.py`
  - `data_processing/cross_rate.py`
  - `data_processing/filters.py`
  - `data_processing/percentile.py`

---

4. Telegram-бот

- **Основна логіка:**
  Реалізується в `telegram/telegram_bot.py`.
- **Сцени:**
  Окремі файли для кожної функціональності:
  - Моніторинг: `telegram/scenes/monitor_scene.py`
  - Торгівля: `telegram/scenes/trade_scene.py`
  - Налаштування: `telegram/scenes/settings_scene.py`
- **Клавіатури:**
  - Головне меню: `telegram/keyboards/main_menu.py`
  - Моніторинг: `telegram/keyboards/monitor_keyboard.py`
  - Торгівля: `telegram/keyboards/trade_keyboard.py`
  - Налаштування: `telegram/keyboards/settings_keyboard.py`
- **Команди:**
  - `/monitor_all` — примусовий запуск моніторингу.
  - `/list_focus` — звіт про обрані пари.
  - `/remove_focus` — видалення пари з моніторингу.
  - `/open_trade` — відкриття угоди.
  - `/close_trade` — закриття угоди.
  - `/settings` — доступ до налаштувань.
- **Особливості:**
  - Обмеження повідомлень до 4096 символів та розділення їх на частини, якщо необхідно.
  - Обробка некоректного введення користувачем.
- **Інтерактивна взаємодія** з користувачем за допомогою команд, клавіатури та сцен.
    i. Додавання пар для загального моніторингу за допомогою окремої телеграм сцени для парсингу повідомлень.

ii. Додавання пар для моніторингу обраних за допомогою телеграм сцени.
    1. Автоматичне(за відсутності ручного вказання) вирахування Z-score та крос курсу для обраної пари. Формат введення “Введіть пару у форматі 'BASE/QUOTE' (наприклад, BTC/ETH). За бажанням додайте початкове відхилення та кроскурс (наприклад, 'BTC/ETH 4.5 1.234'). Для виходу надішліть /cancel.”
iii. Видалення пар із моніторингу обраних за допомогою команди (“/remove_focus”)
iv. Відображення результатів загального моніторингу за розкладом (1 раз на 15 хвилин).
v. Виконання загального моніторингу примусово за допомогою відповідної команди з відображенням результатів моніторингу.(“/monitor_all”)
vi. Відкриття угод за допомогою окремої сцени по команді з клавіатури
vii. Закриття відкритих угод за допомогою окремої команди з клавіатури
viii. Надання локального звіту про суттєві зміни у відкритих угодах за розкладом (1 на 15 хвилин, тобто з кожним оновленням даних)
ix. Надання локального звіту про суттєві зміни серед обраних пар за розкладом (1 на 15 хвилин, тобто з кожним оновленням даних)
x. Надання загального звіту про стан **відкритих** угод за розкладом (1 на годину)
xi. Надання загального звіту про стан **відкритих** угод примусово за допомогою окремої команди з клавіатури
xii. Надання загального звіту про стан **обраних** пар за розкладом (1 на годину)
xiii. Надання загального звіту про стан **обраних** пар примусово за допомогою окремої команди (“/list_focus”)

---

5. Логування

- **Модуль:** `utils/logger.py`
- **Рівні логів:** DEBUG, INFO, WARNING, ERROR.
- **Що логувати:**
    - Процеси оновлення бази даних.
    - Розрахунки Z-score та інших індикаторів.
    - Інтеракцію з Telegram-ботом.

---

6. Тестування

- **Фреймворк:** `pytest`.
- **Модулі тестування:**
  - `tests/test_database.py` — для роботи з базою даних.
  - `tests/test_data_processing.py` — для розрахунків.
  - `tests/test_filters.py` — для перевірки фільтрації.
  - `tests/test_telegram.py` — для перевірки логіки Telegram.
- **Перевіряй:**
  - Коректність роботи бази даних (створення, оновлення, видалення записів).
  - Точність розрахунків (Z-score, Cross-Rate) та інших статистичних індикаторів.
  - Логіку взаємодії Telegram-бота (форматування повідомлень, обробка команд).
- **README для тестів:**
  Додай інструкції з налаштування та запуску тестів.

---

7. CI/CD

- **Інтеграція:** GitHub Actions.
- **Що автоматизувати:**
  - Запуск тестів при коміті.
  - Перевірка форматування коду (PEP 8).

---

8. Форматування

- Дотримуйся PEP 8.
- Зрозумілі назви для всіх елементів.
- Документуй кожен модуль і основні функції.

Додаткові рекомендації

1. **Розширюваність:**
   Забезпеч легке додавання нових фільтрів, сцен і команд.
2. **Обробка помилок:**
   Реалізуй винятки для критичних процесів (API, база даних).

Ці інструкції готові до впровадження. Якщо потрібно, можу допомогти з розробкою першого модуля або тестів.