# **Технічне тестове завдання**

**Посада:** Python Engineer (Trainee/Junior)  
**Тематика:** Бектестинг торгових стратегій з використанням VectorBT  
**Тривалість виконання:** 1 повний робочий день  
**Мета:** Оцінити вміння кандидата будувати масштабовану, чисту, документовану інфраструктуру для бектестингу та аналізу стратегій на високочастотних ринкових даних.

## **Вступ**

Необхідно реалізувати систему для аналізу та бектестингу декількох торгових стратегій на основі 1-хвилинних OHLCV-даних для 100 торгових пар до BTC на Binance за **повний лютий 2025 року**.  
Фреймворк для бектестингу: **VectorBT**.

У завданні важлива не лише функціональність, а й архітектура, масштабованість та якість реалізації. Простий та поверхневий код оцінюватися не буде.

## **Загальні вимоги**

* Дані: 1m OHLCV, 100 пар до BTC з [Binance](https://data.binance.vision/?prefix=data/), 01.02.2025 — 28.02.2025
* Бектест: з урахуванням комісії, сліпейджу, можливого дрифту часу між сигналом і виконанням
* Результати: збереження графіків, метрик, таблиць, агрегація статистики
* Код повинен бути модульним, протестованим (unit-тести на ключові частини), оформленим відповідно до PEP8
* Перевага надається структурі, яку можна масштабувати під більший портфель/таймфрейм/біржу

## **Завдання**

### **1. Завантаження та обробка даних**

* Написати модуль data\_loader, який:
  + завантажує історичні дані 1m з [Binance Data Vision](https://data.binance.vision/?prefix=data/) за вказаний період
  + дозволяє обрати 100 найліквідніших пар до BTC
  + зберігає зведений датасет у форматі .parquet з компресією
  + реалізує кешування та перевірку цілісності даних

### **2. Інтерфейс стратегії**

Необхідно реалізувати базовий абстрактний інтерфейс для торгових стратегій (рекомендація, не є обов’язковим):

class StrategyBase(ABC):

def \_\_init\_\_(self, price\_data: pd.DataFrame):

...

@abstractmethod

def generate\_signals(self) -> pd.DataFrame:

pass

@abstractmethod

def run\_backtest(self) -> pd.DataFrame:

pass

@abstractmethod

def get\_metrics(self) -> dict:

pass

### **3. Реалізація стратегій**

Реалізувати щонайменше **три незалежні** торгові стратегії з різним рівнем складності. Приклади можливих реалізацій:

* **SMA Crossover**
  + Класична стратегія перетину двох ковзних середніх (з фільтрами по волатильності)
* **RSI with Bollinger Band confirmation**
  + Вхід по RSI < 30, підтвердження через відскок від нижньої межі BB
* **VWAP Reversion Intraday**
  + Відкриття позиції при значному відхиленні від VWAP з поверненням до нього
* **Multi-timeframe Momentum**
  + Комбінування сигналів з різних таймфреймів (наприклад, 1m + 15m)
* **ATR-based trailing breakout**
  + Стратегія з підтягуванням стопів по ATR та вхідними умовами на пробиття локального максимуму
* **Volume spike breakout**
  + Детекція об'єму через локальні дельти, вхід на пробиття з підвищеним об'ємом

### **4. Бектест**

* Написати модуль backtester, який:
  + приймає стратегію (через StrategyBase)
  + виконує бектест на всіх парах одночасно через VectorBT
  + вираховує ключові метрики: total return, Sharpe ratio, max drawdown, winrate, expectancy, exposure time
  + зберігає результати в таблиці .csv та графіки в .png

### **5. Вивід результатів**

* Для кожної стратегії:
  + Equity curve
  + Heatmap по performance для всіх 100 пар
  + Порівняння метрик по стратегіях
* Опціонально: згенерувати .html звіт з інтерактивними графіками

## **Очікувана структура проєкту**

project/

├── core/

│ ├── data\_loader.py

│ ├── backtester.py

│ └── metrics.py

├── strategies/

│ ├── base.py

│ ├── sma\_cross.py

│ ├── rsi\_bb.py

│ └── vwap\_reversion.py

├── tests/

│ ├── test\_backtester.py

│ └── test\_strategies.py

├── data/

│ └── btc\_1m\_feb25.parquet

├── results/

│ ├── metrics.csv

│ ├── screenshots/

│ │ └── strategy1\_equity.png

├── main.py

├── requirements.txt

└── README.md

## **Технічні вимоги**

* Python ≥ 3.10
* Використання vectorbt, pandas, numpy, scipy, plotly, pyarrow, ta
* Опціонально: numba, ccxt, pytest
* Розбиття на модулі, використання класів
* Документація до кожного класу та функції (docstrings)
* Мінімум 1 unit-тест на кожну стратегію

## **Як здати**

1. Репозиторій на GitHub (публічний)
2. README з:
   * інструкцією по запуску
   * описом кожної стратегії
   * висновками по результатах
3. Графіки та результати в /results
4. Дані в .parquet форматі або скрипт їх отримання

## **Критерії оцінювання**

| **Параметр** | **Вага** |
| --- | --- |
| Архітектура, масштабованість | 40% |
| Якість коду, стиль, типізація | 20% |
| Документація, тестування | 15% |
| Обробка даних і коректність бектесту | 15% |
| Аналіз результатів | 10% |