Программа курса
«HPPY: Высокопроизводительная обработка данных на Python»

О курсе: Python удобен, но на больших объёмах данных и в «боевых» пайплайнах его скорость быстро становится узким местом. На курсе вы разберётесь, **где именно теряется производительность**, какими способами её вернуть **без переписывания всего проекта на C++**, и **как масштабировать обработку от ноутбука до кластера**.

Фокус — на практических решениях: измеряем, ускоряем, сравниваем подходы и выбираем то, что реально работает в вашей архитектуре.

Аудитория:

- Python-разработчики, data engineers, ML/DS инженеры
- ИТ-архитекторы и архитекторы решений
- Команды, которым нужно ускорить обработку данных, ETL/ELT, аналитические вычисления, сервисы

Уровень подготовки:

- Уверенный Python (пакеты, типы данных, работа с файлами/таблицами)
- Базовое понимание Linux/процессов (на уровне пользователя/разработчика)

Продолжительность курса: 24 академических часов, 5 дней по 4 часа дистанционно

Содержание программы

1. Диагностика производительности: где у Python «болит»

- Интерпретация vs компиляция: что Python делает «под капотом»
- Бутылочные горлышки: CPU / память / I/O / сеть
- Накладные расходы структур Python (list/dict/object) и почему это важно
- Интеграция с Си (и другими языками программирования)

2. Данные решают: представления и форматы, которые ускоряют

- Pandas: сильные стороны, типичные причины деградации, оптимизационные приёмы
- Apache Arrow: колоночное представление, zero-copy, межъязыковая совместимость
- Polars: быстрый DataFrame-подход, ленивые вычисления, векторизация, параллельность «из коробки»
  - **Практика: одну и ту же задачу делаем несколькими способами и сравниваем**

3. «А если SQL?» — когда это быстрее и проще, чем писать Python-циклы

- От SQLite к **DuckDB**: аналитика локально, в файлах, без сервера
- SQL vs "DataFrame" API: что выбирать и почему
- Spark как платформа: DataFrame + SQL, когда он оправдан
  - **Практика:** перенос части вычислений в SQL-движок и оценка выигрыша

4. Параллелизм в Python без мифов

- Процессы и потоки: как это работает
- GIL: почему «потоки не ускоряют python» и что с этим делать
- Параллелизм «за чужой счёт»:
  - Dask / Ray / Spark — когда и какой эффект они дают
  - SQL-движки — параллельные вычисления внутри (DuckDB/Spark/Trino)
    - **Практика:** распараллеливаем задачу и фиксируем эффект

5. Горизонтальное масштабирование: от одной машины к кластеру

- Архитектурные сценарии: batch/stream, ETL/ELT, ad-hoc аналитика, сервисные расчёты
- Варианты кластера и диспетчеризация: Spark, Trino, Ray, Celery, (обзор MPI/SLURM)
- Распределённый SQL (Trino): где он «стреляет», а где нет
- Итоговая карта решений: **что выбрать под ваш кейс**, риски, стоимость владения, пределы

Дополнительно (встроено в курс)

- Разбор ваших задач/архитектуры (по желанию участников)
- Рекомендации по «следующим шагам»: что внедрять первым, чтобы быстро получить выигрыш