Temporal Python SDK Training: 13-Day Syllabus

Week 1: Core Concepts & Foundations

Day 1: Introduction to Temporal & Setup

Topics Covered:

- What is Temporal and its core value proposition (Durable Execution vs. Job Queues, State Machines, Cron)
- Key architectural components: Workers, Temporal Server, Task Queues, History
- Setting up Temporal locally using Docker
- Introduction to Temporal CLI (tctl) for basic operations
- Exploring the Temporal Web UI for monitoring

Practical: "Hello Workflow" - Writing, running, and inspecting a very basic Python Workflow.

Day 2: Workflows in Python - The Basics

Topics Covered:

- Understanding the concept of a Workflow in Temporal
- Workflow functions vs. Activities: distinguishing their roles
- Determinism: Why it's crucial for Workflow reliability and common pitfalls to avoid
- Workflow lifecycle and the significance of history replay
- Creating and executing your first Python Workflow

Practical: Implementing basic Workflow logic using `await`, `Workflow.sleep()`, and handling Workflow input/output.

Day 3: Activities - The Building Blocks

Topics Covered:

- Defining and registering Activities in Python
- Invoking Activities from Workflows
- Retry policies and backoff strategies for robust Activity execution
- Heartbeats for long-running Activities to prevent timeouts
- Understanding Activity timeouts and comprehensive failure handling

Practical: Developing idempotent Activity logic. Real-world example: A data ingestion or email sending Activity with retry logic.

Day 4: Task Queues and Workers

Topics Covered:

- Deep dive into Task Queues and their role in distributing tasks
- Best practices for separating Workflow and Activity workers
- Scaling workers: concurrency management and sharding strategies
- Worker lifecycle and graceful shutdown behavior
- Managing multiple Workflows across different Task Queues

Practical: Setting up and observing worker behavior, implementing basic logging and status tracking for worker observability.

Day 5: Signals and Queries - Interacting with Workflows

Topics Covered:

- Understanding how to interact with running Workflows
- Defining and handling Signals in Workflows for external input (e.g., user input, webhook triggers)
- Defining and handling Queries to inspect Workflow state in real-time

Practical: Real-world use case: Implementing an order approval Workflow or a status polling mechanism. Live demonstration: Sending a Signal to dynamically alter Workflow behavior.

Week 2: Advanced Patterns & Error Handling

Day 6: Timers, Sleep, and External Waits

Topics Covered:

- Differentiating durable timers from standard `time.sleep()`
- Effective use of `Workflow.sleep()` for delays and reminders
- Strategies for waiting for external conditions or input within a Workflow

Practical: Combining Signals and Timers for human-in-the-loop workflows. Example: A Workflow that waits for user confirmation with a timeout fallback.

Day 7: Comprehensive Error Handling and Retry Strategies

Topics Covered:

- Understanding exception propagation from Activities to Workflows
- Implementing custom retry logic using `RetryPolicy`
- Handling specific vs. generic errors within Workflows
- Effective use of `try/except` blocks in Workflows

Practical: Workflow compensation and cleanup logic. Example: A payment failure Workflow with refund compensation.

Day 8: Child Workflows and Parallel Processing (Fan-out/Fan-in)

Topics Covered:

- Spawning child workflows from a parent workflow
- When and why to leverage child workflows for modularity and scalability
- Implementing fan-out/fan-in patterns for parallel processing (e.g., processing 100 tasks concurrently)
- Aggregating results efficiently from child workflows

Practical: Advanced error handling and rollback strategies in distributed processing with child workflows.

Day 9: Workflow Versioning and Long-Running Flows

Topics Covered:

- Why Workflow versioning is essential for determinism and upgrades
- Using `Workflow.get_version()` safely for logic migration
- Strategies for migrating logic in live workflows
- Designing and managing long-running workflows (spanning days/months)
- Understanding history size and performance considerations for long-running workflows

Practical: Use case: Implementing a subscription renewal and delayed billing workflow.

Week 3: Advanced Topics & Project Work

Day 10: Advanced Workflow Patterns and Best Practices

Topics Covered:

- Dynamic Activity and Workflow registration
- Using Memo and Search Attributes for enhanced visibility and querying
- Implementing custom data converters for complex payload serialization

Practical: Building a Workflow that dynamically selects Activities based on input, and demonstrates advanced search/memo capabilities in the UI.

Day 11: Testing Temporal Workflows and Activities

Topics Covered:

- Strategies for unit testing Workflow and Activity logic
- Mocking external dependencies in Temporal tests
- Integration testing with a local Temporal server

Practical: Writing comprehensive unit and integration tests for a previously developed Workflow and its Activities.

Day 12: Observability, Monitoring, and Deployment Considerations

Topics Covered:

- Integrating Temporal with Prometheus/Grafana for metrics collection

- Advanced logging and tracing in Temporal applications
- Deployment strategies for Temporal Server and Workers (e.g., Kubernetes)
- High availability and disaster recovery for Temporal

Practical: Setting up basic monitoring for a Temporal application and analyzing key metrics. Discussion on production deployment considerations.

Day 13: Capstone Project & Final Q&A

Topics Covered:

- Review of project requirements
- Group and individual project presentations (if applicable)
- Code walkthrough and feedback session
- Recap of key Temporal patterns and architectural considerations
- Tips for deploying Temporal in a production environment
- Final open Q&A and feedback session

Practical: Dedicated time for project work, troubleshooting, and refinement with instructor support.