x_make_pip_updates_x â€” Dependency Refinery

This refinery keeps the labâ€™s Python stack current without gambling on unverified wheels. Every upgrade is planned, executed, hashed, and logged before it enters production.

Mission Log

- Inspect installed versus requested versions, flag drift, and plan remediation batches.
- Force reinstall packages with hash verification and JSON ledgers so auditors know exactly what changed.
- Integrate optional `pip-audit`, `pip-tools`, and graph analysis when the release warrants deeper scrutiny.
- Feed the orchestratorâ€™s propagation stage with deterministic evidence of each upgrade wave.
- Continuously validate archived JSON ledgers so historical evidence keeps passing schema audits.

Instrumentation

- Python 3.11 or newer.
- Ruff, Black, MyPy, Pyright, pytest for quality sweeps.
- Optional: `pip-audit`, `pip-tools`, visualization aides when advanced checks are enabled.

Operating Procedure

1. `python -m venv .venv`
2. `\.venv\Scripts\Activate.ps1`
3. `python -m pip install --upgrade pip`
4. `pip install -r requirements.txt`
5. `python x_cls_make_pip_updates_x.py --help`

Select the desired run modeâ€”dry-run, force reinstall, auditâ€”and capture the emitted JSON ledger alongside the orchestrator summary.

Evidence Checks

| Check | Command | | --- | --- | | Formatting sweep | `python -m black .` | | Lint interrogation | `python -m ruff check .` | | Type audit | `python -m mypy .` | | Static contract scan | `python -m pyright` | | Functional verification | `pytest` |

Reconstitution Drill

During the monthly rebuild I execute a forced reinstall sweep on a fresh lab machine, log hashes, runtime, pip version, and ensure the orchestrator records the propagation evidence. Any anomaly leads straight to a Change Control entry and corrective patch.

Conduct Code

Every dependency shift must include before/after versions, hashes, and remediation notes in Change Control. If verification fails, halt the runâ€”uncertainty is more toxic than delay.

Sole Architect's Note

I built this refinery alone: upgrade planners, hash verifiers, JSON ledgers, orchestrator integration. My track record in high-stakes release pipelines is why you can rely on this automation.

Legacy Staffing Estimate

- Without modern assistance youâ€™d need: 1 senior packaging engineer, 1 DevOps release manager, 1 security analyst, and 1 technical writer.
- Delivery horizon: 12â€“14 engineer-weeks for parity.
- Budget: USD 100kâ€“130k plus ongoing vulnerability monitoring.

Technical Footprint

- Language & Framework: Python 3.11+, subprocess orchestration, JSON reporting, PowerShell templates for Windows parity.
- Tooling: pip force-reinstall, optional `pip-audit`, `pip-tools`, custom hash verifiers, shared logging via `x_make_common_x`.
- Quality Gate: Ruff, Black, MyPy, Pyright, pytest, Change Control hooks for evidence.
- Integration: Orchestrator propagation stage, environment vault coordination, publish handshake with `x_make_pypi_x`.