Python and engineering judgment

Python / Invent and Simplify / Have Backbone; Disagree and Commit

Full guide: 11h 50m | Core route: 4h | Read this map first, then use the final recall strip.

BEFORE STUDY Contracts/types → merge/sums/windows → JSON/CSV/dataframes → replay and testing → simplification + a lost-argument story.

01 / Code with a contract

Clarify input/output types, order, identity, duplicates, nulls, invalid rows, mutation, size and sortedness. Work one example before coding.

State the invariant and a readable baseline; test normal, empty, boundary and adversarial cases. Explain time, retained state and output memory.

Python traps: bool subclasses int; sort() returns None; shallow copies share nested state; mutable defaults persist.

02 / Required algorithm patterns

Merge sorted arrays: two cursors, preserve duplicates, drain suffix. O(n+m). Stream variant: one head per iterator and a unique exhaustion sentinel.

k-way merge: one head per source in a heap; deterministic comparable tie keys precede payloads. O(N log k) after setup.

Two-sum: complement lookup before insert; expected O(n). Three-sum: sorted copy, fixed value, two pointers, skip duplicates; $O(n^2)$.

Nearby duplicates: define distance<=k; retain previous k positions. Sliding distinct counts need multiplicity. Time windows need timestamp ordering/lateness policy.

03 / JSON, CSV and weighted ranking

Parse with json/csv, validate required schema, then transform. Missing, null, zero and numeric text are different. csv.writer handles quotes, commas and newlines.

Nested JSON: define empty arrays and group grain. json.loads materializes the document; NDJSON streams records, but group state may grow.

Across files: dedup event IDs, skip exact retries, reject conflicts. Validate headers/fields/counts; distinguish no files from header-only files.

Rank sum(successes)/sum(attempts), not average file rates. Define zero denominator and deterministic ties.

04 / Choose honest complexity

dict/set: average constant lookup with O(unique keys) state. deque: FIFO ends. Counter: multiplicities. heap: bounded best-k selection.

Generators can retain growing state. Chunking does not make global dedup/grouping constant-memory.

For larger-than-RAM exact work, partition, spill/external-sort or distribute while preserving identity and ordering contracts.

05 / Dataframe and data-lifecycle transfer

Pandas: validate many-to-one merges; groupby(dropna=False) retains unknowns. Null matching differs from ordinary SQL-style joins.

Spark: native select/filter/groupBy/window functions; validate right-key uniqueness; avoid unbounded collect/toPandas. Explode can change parent population.

Resolve latest version before removing deletes. Preserve unmatched events. Use Counter-style reconciliation when duplicates matter.

Replay: stable event identity + idempotent sink or transactional state/output. A checkpoint alone cannot make additive writes exactly once.

06 / Test and repair the mechanism

Test empty/singleton, duplicates, exact boundaries, heap ties, bad schema, one-shot input and nonmutation.

Compare small random cases with an independent brute-force oracle. Repeat a retry file, reorder files and split batches to test invariants.

Use Decimal from strings with explicit rounding for exact decimal contracts. Catch expected parsing errors; surface programming errors.

On failure, find a minimal counterexample, fix the invariant and rerun old/new tests. Fixtures do not prove every input.

07 / Invent and Simplify

Recurring customer burden → your diagnosis → simpler contract/mechanism → pilot and independent use → supported outcome → lesson.

Show what you removed or reused and what stayed explicit. A generic framework can hide complexity; fewer lines alone do not prove simplification.

Prepare two real incidents with actual adoption, effort and quality evidence. Keep hypothetical rehearsal numbers out of personal claims.

08 / Disagree, decide and commit

Understand the other constraint. Challenge with evidence, a concrete alternative and trade-offs. Identify the decision owner and outcome.

Prepare a story where you lost the argument. Show next-day commitment: changed plan, implemented chosen design, owned recovery and communicated consistently.

Reopen on new customer/SLA/recovery evidence, not unchanged preference. Escalate unresolved integrity/security obligations appropriately.

FINAL 5-MINUTE RECALL Code merge + window cold • explain JSON/CSV validation • test retry/conflict • count ALL memory • rehearse simplification + losing an argument • say the invariant before the code.

ELI5: Make the function work, challenge its assumptions and own the decision.

Sources: Python data structures | Python CSV | Leadership Principles