Coverage.py Python code coverage

START HERE

`coverage run -m pytest`
Measure a test run

`coverage report -m`
Table with missing lines

`coverage html`
Browsable report in htmlcov/

`coverage run --branch -m pytest`
Also measure branches

`coverage run --source=<pkg> -m pytest`
Only measure your package

`coverage report --fail-under=<n>`
CI gate, exit 2 below n%

`coverage combine`
Merge parallel data files

`coverage erase`
Start from zero

coverage run your tests or script → writes → **.coverage** SQLite data file → read by → **report / html / xml** json, lcov, annotate → for → **Humans and CI** --fail-under gate, upload

CONCEPTS

Two steps: run records execution into a .coverage data file, then report, html, xml, json or lcov read it. Reports never run your code again. Without --append each run overwrites the data.

Settings live in .coveragerc, setup.cfg, tox.ini or pyproject.toml ([tool.coverage.run], [tool.coverage.report]). Command-line flags override them. With pytest-cov, pytest --cov=<pkg> does run and report in one go.

MEASURE

`coverage run <script.py> [args]`
Run a script under measurement

`coverage run -m <module> [args]`
Run a module, like python -m

`coverage run -m pytest`
Measure the test suite

`coverage run --branch -m pytest`
Count branches, not just lines

MEASURE (cont.)

`coverage run --source=<dir-or-pkg> ...`
Limit measuring to this code
Also reports files that were never imported as 0%.

`coverage run --omit=<pat> ...`
Skip files matching a pattern

`coverage run -a ...`
Append to the existing data

`coverage run -p ...`
Separate data file per process

`coverage run --concurrency=<libs> ...`
gevent, eventlet, greenlet, multiprocessing, thread

REPORT

`coverage report`
Text table of coverage per file

`coverage report -m`
Add the missing line numbers

`coverage report --skip-covered`
Hide files at 100%

`coverage report --fail-under=<n>`
Exit 2 if total is below n

REPORT (cont.)

`coverage report --format=markdown`
Markdown table (or text, total)

`coverage report --format=total`
Print only the total percentage

`coverage report --include=<pat>`
Filter files in this report

HTML AND OTHER FORMATS

`coverage html`
Write the HTML report to htmlcov/

`coverage html -d <dir>`
Choose the output folder

`coverage xml -o <file>`
Cobertura XML (default coverage.xml)

`coverage json -o <file>`
JSON report (default coverage.json)

`coverage lcov -o <file>`
LCOV tracefile (coverage.lcov)

`coverage annotate -d <dir>`
Annotated copies, > hit and ! missed

DATA FILES

`coverage combine`
Merge .coverage.* files into .coverage

`coverage combine --keep`
Keep the files that were merged

`coverage erase`
Delete the collected data

`coverage debug data`
Summarise the data file contents

`coverage debug sys`
Version, install and config info

`coverage debug config`
Show the settings in effect

`COVERAGE_FILE=<path> coverage run ...`
Data file location by env var

EXCLUDING CODE

`# pragma: no cover`
Exclude this line, or the block it opens

`# pragma: no branch`
Don't count a missing branch here

`exclude_also = [...]`
Add regexes to the default exclusions

EXCLUDING CODE (cont.)

`[run] omit = [...]`
Config: files never measured

`[report] fail_under = <n>`
Config: default threshold

`[run] branch = true`
Config: always measure branches

`[run] source = ["<pkg>"]`
Config: always limit to your code

PARALLEL AND CI

`[run] parallel = true`
Config: unique data file per process

`[run] relative_files = true`
Relative paths, so CI can merge them

`[paths] src = ["src/", "*/site-packages/"]`
Map paths when combining

`[run] concurrency = ["multiprocessing"]`
Config: measure child processes

RECIPES

Local loop

```
coverage erase
coverage run --branch -m pytest
coverage report -m --skip-covered
coverage html && xdg-open htmlcov/index.html
```

CI gate with an XML upload

```
coverage run --branch --source=<pkg> -m pytest
coverage report --fail-under=90
coverage xml
```

Parallel runs merged

```
# pyproject.toml: [tool.coverage.run] parallel = true
coverage run -m pytest tests/unit
coverage run -m pytest tests/integration
coverage combine
coverage report -m
```

pyproject.toml basics

```
[tool.coverage.run]
branch = true
source = ["<pkg>"]

[tool.coverage.report]
show_missing = true
exclude_also = ["if TYPE_CHECKING:"]
```