17.12.2025

SEDS 514 Software Testing

Project 3

Pairwise Test Design on a Real OSS Project

Context & goal

In this project you will apply **pairwise testing** (2-way combinatorial interaction testing) to a real open-source system with many configurable inputs. Your job is to (1) model the input space as factors/levels, (2) generate a **pairwise covering array**, (3) implement the tests, and (4) evaluate effectiveness vs. a small baseline.

Target GitHub project (System Under Test)

`https://github.com/astanin/python-tabulate` — a Python library + CLI that pretty-prints tables in many formats.

What you will test

You will design pairwise tests around the `tabulate()` function options.

Step 1 — Build the pairwise model (your "test design spec")

Create a model with **7 factors**, each with **levels given below**:

Factors

1. **Input table type**
   - list of lists
   - list of dicts
   - dict of columns
2. **Headers mode**
   - explicit list

    - `"firstrow"`
    - `"keys"`
3. **Table format (`tablefmt`)**
    - `"plain"`
    - `"github"`
    - `"grid"`
    - `"psql"` (or another supported format)
4. **Row indices (`showindex`)**
    - `"always"`
    - `"never"`
    - custom iterable (e.g., `["r0","r1",...]`)
5. **Missing value handling (`missingval`)**
    - default behavior
    - `"?"`
    - `"NA"`
6. **Data mix**
    - all strings
    - ints + floats
    - includes `None` values
7. **Row/column size**
    - small (2×2)
    - medium (5×4)
    - includes wide text (long strings)

Constraints (mandatory)

You must define and enforce **at least 5 constraints**, e.g.:

- If **Input table type = list of dicts**, then `headers="firstrow"` is invalid.
- If **Headers mode = "keys"**, then input must be dict-like (list of dicts or dict of columns).
- If you include a level that is known to raise an exception, mark it as a **negative test** and define the expected exception.

Deliverable: 1–2 pages describing factors, levels, and constraints (clear enough that someone else could regenerate your test suite).

Step 2 — Generate pairwise test cases

Use any pairwise generator (examples):

- Microsoft **PICT**
- NIST **ACTS**
- Python libraries such as `allpairspy` (or equivalent)

Requirements:

- Produce a **pairwise test set** for your model (after constraints).
- Report:
  - number of generated tests
  - number of theoretical combinations (if exhaustive)
  - reduction ratio

Deliverable: a CSV/JSON of generated test cases + a short explanation of the tool and settings.

Step 3 — Implement the tests (pytest)

Oracles (what to assert)

At minimum, each test must assert **two** of the following:

1. Output is **non-empty** and contains expected header labels (when headers are enabled).
2. Output respects **table format characteristics** (assert stable markers for chosen formats).
3. When `showindex="always"`, the output includes an index column.
4. When `None` appears and `missingval="?"`, the output contains `"?"` at the correct positions.

Practical guidance

- Make tests **deterministic** (avoid locale/time dependence).
- Prefer **small, readable fixtures**.
- Include a small number of **negative tests** driven by your constraints.

Deliverable: runnable test suite (`pytest`) + a brief README: how to install and run.

Step 4 — Evaluate effectiveness vs. a baseline

Pick **the** baseline as random testing with the same budget (#tests)

Report:

- failures found (if any), and whether they were due to your model, oracle, or an actual defect
- statement coverage (optional), but you must at least report **pairwise coverage achieved**

Deliverable: 2–3 page report + evidence (logs/screenshots/coverage output).

Submission checklist

1. **Design spec** (factors/levels/constraints)
2. Pairwise **generated test set** (CSV/JSON)
3. `pytest` test suite + instructions
4. Short **evaluation report**

Use a GPT to obtain a starter repo structure (folders, pytest scaffolding, and a sample pairwise CSV schema) tailored to `python-tabulate`.

Please send all your code (zip file) and report (pdf file) to tugkantuglular@iyte.edu.tr.

Submission Rules:

- Due Date: 11.01.2025, 23:55
- If any cheating is detected in your homework, will be graded as 0.
- Please export your Java Project as the given format with your student ID: SEDS514_Project2_StdID1_StdID2.zip.