Instructions for using SAT Solver

This project implements in `Python 3` three classic algorithms for solving the Boolean satisfiability (SAT) problem:

- The **Davis–Putnam (DP)** elimination procedure,
- The **Davis–Putnam–Logemann–Loveland (DPLL)** backtracking search,
- A pure **Resolution**–based SAT solver.

Each script can be run interactively on user-entered CNF formulas or in benchmark mode over a directory of DIMACS CNF files.

System Requirements

- **Python 3.7+** (no external packages—only the standard library).
- Compatible with Windows, macOS, or Linux.
- A **terminal** environment:
  - *Linux/macOS*: Bash, Zsh, or any POSIX-style shell
  - *Windows*: Command Prompt (`cmd.exe`) or PowerShell
- Optional: multi-core CPU to take advantage of parallel benchmarks.

Files Provided

`dp.py` implements the DP elimination algorithm.
`dpll.py` implements the DPLL search algorithm.
`resolution.py` implements the standard Resolution procedure.

General CNF Format

When entering CNF formulas manually or providing files, adhere to DIMACS conventions:

1. Clauses are lists of integer literals terminated by `0`.
2. Positive integers denote variables; negative integers denote negated variables.
3. In files, comment lines start with `c` or `%`, and the problem line starts with `p cnf`.
4. Each non-comment line contains one clause.

Usage Patterns

Interactive Mode

Solve a user-entered CNF via standard input:

```
$ python dp.py
$ python dpll.py
$ python resolution.py
```

The program will prompt:

1. `Enter number of clauses:` — total clauses n.
2. Then enter each clause on its own line, e.g. `1 -3 4 0`
3. The solver reports:
   - **SATISFIABLE** or **UNSATISFIABLE**,
   - *Time elapsed*,
   - *Peak memory usage.*

Benchmark Mode

Run a bulk benchmark over a directory of DIMACS `.cnf` files:

```
$ python dpll.py /path/to/cnf_dir --sample-per-block 10 --workers 4
```

(or replace with `dp.py` or `resolution.py`)

How it works:

- Divides the sorted list of `*.cnf` files into 10 blocks.
- Samples up to `N` files per block (default N=10).
- Processes samples in parallel using up to `-workers` processes (default: all CPU cores).
- Prints per-file results like: `filename.cnf: SAT in 0.123s, 1.23MiB`
- Then a summary of total files, #SAT, #UNSAT, total time, average time, fastest/slowest solve, and min/max memory usage.

Accepted Arguments

`cnf_dir` (positional) Path to folder containing `.cnf` files. Omit for interactive mode.
`-sample-per-block` `N` Number of files sampled per block (default: 10).
`-workers` `W` Number of parallel worker processes (default: CPU count).

Important Notes

- These scripts use only Python's standard library—no `pip` installation required.
- For large benchmarks, ensure sufficient memory; pure resolution can be memory-intensive.
- Interrupt any running benchmark with `Ctrl+C`, which will cleanly terminate workers.