

PuzzleForge Cloud — Word Search, Crossword & Maze Guide

Included with **PuzzleForge Cloud** at no extra cost — there is no separate expansion pack to buy. Alongside the original Sudoku engine (see the Sudoku guide), PuzzleForge Cloud includes three more puzzle generators — **word search, crossword, and maze**. Every generator emits a puzzle page **and** an answer-key / solution page, exported as print-ready KDP interior PDFs through the same pipeline geometry as the Sudoku output (same trim sizes, margins, title/footer style).

What's included

File	Purpose
app/puzzleforge_expansion.py	The three generators + KDP PDF export
app/cli.py	Command-line runner
app/factory_media.py	PuzzleForge's original module, included **unmodified** for reference (the Sudoku pipeline these generators match)
tests/test_expansion.py	Programmatic validity tests (see BUILD-NOTES.md)
samples/wordsearch_animals.txt	Sample word list (one word per line)
samples/crossword_clues.csv	Sample word+clue file (word,clue header)
output/sample_*_book.pdf	Pre-built sample books, 12 puzzles + 12 keys = 24 pages each

Setup

Python 3.11+. Dependencies: reportlab, pypdf (tests only), pytest (tests only). Generation itself is pure Python stdlib.

```
pip install reportlab pypdf pytest
```

Running each generator

Run from the toolkit's root folder.

Word search

```
python -m app.cli wordsearch --words-file samples/wordsearch_animals.txt --count 12 --out output/my_wordsearch.pdf  
python -m app.cli wordsearch --theme space --count 12 --out output/space.pdf
```

- --words-file: text file, one word per line (any case; stored uppercase).
- --theme: built-in themed lists instead of a file — animals, space, food, ocean (10 words each).
- --size: grid size, default 15 (15×15). Words are placed in all 8 directions (horizontal, vertical, diagonal, forwards and backwards). Cells may be shared only where letters match; remaining cells are filled with random letters.
- The printed page lists the words to find; the answer-key page highlights every placed word.

Crossword

```
python -m app.cli crossword --clues-file samples/crossword_clues.csv --count 12 --out output/my_crossword.pdf
```

- --clues-file: CSV with header word,clue, one pair per row.
- --size: grid size, default 15.
- The generator builds a **real interlocking grid**: words are placed by a backtracking layout search so every word after the first crosses at least one other word on a matching letter. If a word genuinely cannot be placed it is **reported in the output metadata's skipped list and left out** — it is never drawn into the grid or the clue list.
- Puzzle page: numbered blank grid + ACROSS / DOWN clue lists. Answer-key page: the filled grid.

Maze

```
python -m app.cli maze --rows 15 --cols 15 --count 12 --out output/my_mazes.pdf
```

- --rows / --cols: maze dimensions in cells, default 15×15.
- Mazes are generated with a depth-first backtracker, producing a **perfect** maze: the passage graph is a tree, so there is **exactly one** solution path from the S (start, top-left) to the F (finish, bottom-right). The solution page draws that unique path.

Common options

- --count: puzzles per book (default 12 → 24 pages, which meets KDP's 24-page paperback minimum; the metadata reports minimum_paperback_24pages_met).
- --seed: integer seed; same seed + same inputs = same book.
- --trim: 6x9 (default) or 8.5x11.

The CLI prints a JSON summary (pages, per-puzzle placement counts, skipped words, byte size) after writing the PDF.

Output specs

Identical to PuzzleForge's Sudoku interiors (factory_media.kdp_pdf):

Spec	6×9 trim	8.5×11 trim
Page size	432 × 648 pt	612 × 792 pt
Grid margin	58 pt	65 pt
Title	Helvetica-Bold 16, top center	same
Footer	PuzzleForge \ Page N \ Independent print-format review required	same
Page order	all puzzle pages, then all answer keys	same

As with the base product, the PDF is a **digital interior proof**: verify final page count, KDP margins, bleed, trim and ISBN settings in KDP before uploading.

Limits (measured, not guessed — Merg1/VM test runs, 2026-10-27)

- **Word search**: 30 words placed 30/30 in a 20×20 grid; 10/10 in 15×15.

Words longer than the grid are skipped and reported. Practical envelope:
keep word count $\leq$ grid area $\div$ 4 for reliable placement on the first pass.

- **Crossword:** the 12-word sample set places 12/12 on a 15×15 grid.

A 20-word set places only 11/20 at 15×15 but **20/20 at 19×19 and 21×21**
— denser sets need a bigger grid. At 13×13 the 12-word sample places 9/12
(GRID, BOOK, INK skipped and reported). Words must be 2+ letters, A–Z
only, and no longer than the grid. Unplaceable words are always reported
in skipped, never silently dropped from the metadata.

- **Maze:** verified perfect (edges = cells – 1) and uniquely solvable at
10×10 through 25×25 in tests; generation of a 40×40 maze takes ~0.02 s
with a 289-cell solution path. No practical size cap in the code; very
large mazes make small cells on a 6×9 page.

- Generation speed: a full 12-puzzle book of any type builds in ~1 s.

Programmatic use

```
from app.puzzleforge_expansion import (
    generate_word_search, generate_crossword, generate_maze,
    build_word_search_book, build_crossword_book, build_maze_book)

pz = generate_word_search(["ELEPHANT", "GIRAFFE"], size=15, seed=42)
pz.placements # [{word, row, col, dr, dc}] — verifiable coordinates
pz.skipped    # words that could not be placed

cw = generate_crossword(["PYTHON", "Programming language"], size=15)
cw.entries    # [{word, clue, row, col, dir, number}]
cw.skipped    # [{word, clue}] not placed

mz = generate_maze(rows=15, cols=15, seed=42)
mz.solution   # the unique start->finish cell path

pdf_bytes, meta = build_maze_book(count=12, rows=15, cols=15, seed=42)
```