BitWith Bite

PYGAME MASTERY WORKSHEET

# Save & Load Systems

Module 17 of 20 · Level ★★☆ · Time: 40 min

SCORE ___ / 20

Name _____ Class _____ Date _____

AFTER THIS WORKSHEET YOU CAN

✓ Save and load game data with json ✓ Keep settings separate from progress ✓ Build a sorted high-score table ✓ Implement idempotent achievement checks

QUICK RECAP

A save file is a snapshot of state written to disk with Python's json module. Gather everything worth saving into one dictionary, dump it with json.dump(), and load it back with json.load() — always guarding against a missing or corrupted file with sensible defaults.

## Section A · Concept Check • BEGINNER 4 × 1 = 4

**1** Which module is used to save and load game data as readable text in this course?

◯ A. pickle ◯ B. json ◯ C. csv ◯ D. sqlite3

**2** What does json.dump(data, f, indent=2) do?

◯ A. Loads data from a file

◯ B. Deletes the save file

◯ C. Writes data to the file, pretty-printed

◯ D. Compresses the save data

**3** Why wrap save/load file access in try/except?

◯ A. It's a syntax requirement

◯ B. The save file may not exist yet or could be corrupted

◯ C. It makes the game run faster

◯ D. It disables achievements

**4** What must happen to a high-score list before it's saved back to disk?

◯ A. Encrypt it

◯ B. Sort descending and trim to MAX_ENTRIES

◯ C. Shuffle it ◯ D. Nothing, save it as-is

## Section B · Problem Solving • INTERMEDIATE 2 + 3×3 = 11

**5** The function that reads JSON data back into a Python object is _____.

**6** Settings should be stored in a file separate from the _____ file so a New Game doesn't reset them.

**7** Why should settings.json be kept separate from savegame.json?

**8** What two dict-merging steps ensure an old settings.json still works after new setting keys are added in a game update?

**9** In the achievement system, what field in player_data prevents the same achievement from re-triggering its popup?

## Section C · Challenge • CHALLENGE 5

**10** A player's savegame.json becomes corrupted mid-write after a crash. Walk through what load_progress() should do, step by step, so the game doesn't crash on next launch.

Reflection — the most useful thing I learned:

A ___/4 B ___/11 C ___/5 **Total ___/20**

Teacher's Signature

Parent's Signature

✂ ANSWER KEY — FOLD OR CUT BEFORE HANDING OUT

1-B 2-C 3-B 4-B | 5 json.load() 6 savegame.json (the progress file) 7 = settings like volume/resolution should persist across a "New Game," but progress resets — mixing them into one file would wipe preferences the player didn't intend to lose 8 = start from a copy of DEFAULT_SETTINGS, then call merged.update(data) with whatever keys were found in the loaded file — defaults fill in any missing new keys while the player's saved values still override them 9 = the "achievements" list/set stored in player_data — check_achievements() only unlocks ids that aren't already present there | 10 = check whether the file exists first; open and json.load() it inside a try/except block; if json.JSONDecodeError or OSError is raised, treat it like a first launch and return the sensible default progress dict instead of letting the exception crash the game