Quiz Game — Instructor & Student Guide

Print / Save as PDF

Comprehensive build, test, and submission instructions for the Quiz Game Python project.

> **Quick Navigation**
>
> 1. Overview & Learning Outcomes
> 2. Setup Requirements
> 3. Folder Structure
> 4. Annotated Sample Code
> 5. Step-by-Step Build Guide
> 6. Testing & Validation
> 7. Submission & Grading
> 8. Extensions & Troubleshooting
>
> *Tip:* Click "Print / Save as PDF" above to export this guide for distribution.

1. Project Overview & Learning Outcomes

The Quiz Game challenges students to design a question-based command-line application where users answer random quiz questions, earn points, and see their score. The project reinforces loops, conditionals, data structures, and input validation.

Learning Outcomes

- Work with lists and dictionaries to manage quiz data.
- Use loops and conditionals for game logic.
- Handle user input gracefully and validate answers.
- Display dynamic scoring and progress feedback.
- Develop a modular, reusable Python script.

2. Prerequisites & Setup

- Python 3.8 or higher (verify with `python --version`).
- A text editor or IDLE (recommended for beginners).
- Basic familiarity with loops, conditionals, and lists.

> **Instructor note:** You can preload quiz questions in a text file or JSON if you want to scale this project later.

3. Folder Structure

```
QuizGame/
```

```
├─ quiz_game.py # Main Python script
└─ README.md # Instructions and reflection
```

4. Annotated Sample Code

```
 # quiz_game.py


import random

Step 1: Prepare the quiz questions

questions = [
{"question": "What is the capital of France?", "answer": "Paris"},
{"question": "What is 5 + 7?", "answer": "12"},
{"question": "Who wrote 'Romeo and Juliet'?", "answer": "Shakespeare"},
{"question": "What is the boiling point of water (in Celsius)?", "answer": "100"},
{"question": "Which planet is known as the Red Planet?", "answer": "Mars"}
]

def run_quiz():
print("Welcome to the Quiz Game!")
score = 0
random.shuffle(questions)

for i, q in enumerate(questions, start=1):
    print(f"\\nQuestion {i}: {q['question']}")
    user_answer = input("Your answer: ").strip()
    if user_answer.lower() == q['answer'].lower():
        print("✅ Correct!")
        score += 1
    else:
        print(f"❌ Wrong! The correct answer is {q['answer']}.")
print(f"\\nYou scored {score}/{len(questions)} correct answers.")
if score == len(questions):
    print("🏆 Excellent! You got all questions right.")
elif score >= 3:
    print("👍 Good job! Keep practicing.")
else:
    print("💡 Try again to improve your score.")


if name == "main":
run_quiz()
```

5. Step-by-Step Build Walkthrough

1. Create a new Python file named `quiz_game.py`.
2. Define a list of dictionaries to store each question and its correct answer.
3. Use `random.shuffle()` to randomize question order each run.
4. Iterate through the list, printing each question and collecting input via `input()`.
5. Compare the user's input (case-insensitive) with the stored answer.

6. Maintain a `score` counter that increments on each correct response.
7. After all questions, print the total score and performance feedback message.

6. Testing & Validation

1. Run `quiz_game.py` from IDLE (press F5).
2. Answer all questions — verify score increments correctly.
3. Test uppercase/lowercase input (e.g. "paris" vs "Paris").
4. Ensure total score matches number of correct answers.
5. Confirm random order changes per run.

7. Submission & Grading Rubric

Submission Package

```
Firstname_Lastname_QuizGame/


├─ quiz_game.py
└─ README.md
```

Rubric

- **Functionality (50%)** — Quiz runs correctly, validates answers, and counts score.
- **Code clarity (25%)** — Proper naming, indentation, and comments.
- **User experience (15%)** — Clear instructions, output formatting, and feedback.
- **Creativity (10%)** — Bonus for additional question categories or timed play.

8. Extensions & Troubleshooting

- **Add more topics:** Create categories (e.g., Science, History, Math).
- **Timed challenge:** Use `time.time()` to limit response time.
- **Randomized scoring:** Give higher points for faster answers.
- **File-based questions:** Store questions in a CSV or JSON and load dynamically.

> **Common error:** Watch out for capitalization in answers. Always use `.lower()` when comparing.

Instructor Checklist

Ensure Python installed ✅

Provide this HTML or PDF ✅

Collect submissions in repo ✅

Test scoring accuracy

Files to Add

- `python-guides/quiz-game-guide.pdf`
- `python-samples/quiz_game.py`
- `site/projects/python-project-2.html`

> **Quick start (for students)**
>
> 1. Open folder in IDLE
> 2. Run `quiz_game.py`
> 3. Answer all questions & check score