Python Practical Test: OOP + Core Concepts (Beginner to Intermediate)

Duration: 3 Hours
Total Questions: 25
Instructions:

- Answer all questions.
- Code must be syntactically correct and executable.
- Where applicable, include both class definitions and instantiation examples.

SECTION A: Core OOP Concepts (Classes, Objects) – 10 Marks

1. **(2 marks)** Define a class `Book` with attributes `title`, `author`, and `year_published`. Include a method `get_info()` that returns a formatted string with book details.

2. **(1 mark)** Create an instance of the `Book` class for the book "1984" by George Orwell, published in 1949.

3. **(2 marks)** Add a class variable to `Book` that keeps track of the total number of books created. Display the count.

4. **(2 marks)** Modify the `Book` class to accept a list of genres as an attribute. Include a method `has_genre(genre)` that checks if the book belongs to a specific genre.

5. **(3 marks)** Create a class `Library` that contains a list of books and has:

    - `add_book(book)` to add to the list.
    - `get_books_by_author(author)` to return all books by that author.

SECTION B: Inheritance – 10 Marks

6. **(2 marks)** Create a base class `Employee` with attributes `name`, `age`, and `salary`. Add a method `work()` that prints a generic work message.

7. **(2 marks)** Create a subclass `Manager` that inherits from `Employee`. Add an attribute `team_size` and override the `work()` method to include their role and team size.

8. **(2 marks)** Create another subclass `Developer` with an extra attribute `language`. Add a method `code()` that prints a message showing what language they're coding in.

9. **(2 marks)** Create a list of `Employee` objects (some Managers, some Developers). Loop through the list and call the `work()` method on each.

10. **(2 marks)** Write a function `calculate_total_salary(employees)` that takes the list and returns the total salary.

SECTION C: Data Structures inside Classes – 5 Marks

11. **(2 marks)** Create a class `Student` with a dictionary to hold subjects and marks. Add methods to:

- Add a subject and mark
- Calculate average

12. **(1 mark)** Create a class `Classroom` with a list of `Student` objects. Write a method to print names of all students scoring an average above 75.

13. **(1 mark)** Add a set inside the `Classroom` class to keep track of all unique subjects taught.

14. **(1 mark)** Use a tuple to store each student's date of birth (dd, mm, yyyy) in the `Student` class.

SECTION D: Functions, Loops, and Logical Thinking – 10 Marks

15. **(2 marks)** Create a class `BankAccount` with methods:

- `deposit(amount)`

- `withdraw(amount)`
- `get_balance()`

16. **(1 mark)** Use a loop to simulate 5 deposits and 3 withdrawals.
17. **(2 marks)** Create a class `Cart` that uses a dictionary to store item names as keys and quantity as values. Include:

- `add_item(name, qty)`
- `remove_item(name)`
- `total_items()`

18. **(1 mark)** Create a method in `Cart` to display all items and quantities using a loop.
19. **(1 mark)** Write a class `School` with a method `enroll_students(*names)` using variable-length arguments to accept any number of names.
20. **(1 mark)** In the `School` class, create a method that returns the longest student name using a loop.
21. **(1 mark)** Use list comprehension inside a class method to return all student names in uppercase.
22. **(1 mark)** Write a method in `Cart` to return a list of items where the quantity is more than 2.
23. **(1 mark)** Write a method that checks whether a student exists in the list using the `in` operator.
24. **(1 mark)** In the `Student` class, write a method that returns a dictionary of subjects where the score is above 80.
25. **(1 mark)** In any class, demonstrate the use of a `while` loop to simulate a basic countdown timer for an event (from 5 to 1).
