

Python Fundamentals - 30-Minute Assessment Test

Duration: 1 hour

Total Problems: 5

Instructions:

- Solve all problems using Python
 - Focus on clean, readable code
 - Test your code with different inputs
 - No external help or AI tools allowed during the test
-

Problem 1: Railway Ticket Validator (10 minutes)

Indian Railways has hired you to write a ticket validation program.

Requirements:

- Accept passenger name, age, and ticket class (Sleeper/AC/General)
- Check if the passenger is eligible for senior citizen discount (age ≥ 60)
- Calculate ticket price based on:
 - Sleeper: ₹500
 - AC: ₹1200
 - General: ₹200
- Apply 40% discount for senior citizens
- Display final ticket details with passenger name, age, class, and final price

Sample Input:

```
Name: Ramesh Kumar  
Age: 65  
Class: AC
```

Expected Output Format:

```
Passenger: Ramesh Kumar  
Age: 65  
Class: AC  
Original Price: ₹1200  
Discount Applied: 40%  
Final Price: ₹720
```

Problem 2: Flipkart Order ID Generator (10 - 15 minutes)

Flipkart needs a unique order ID generator for their system.

Requirements:

- Accept customer name and product category
- Generate an order ID using this pattern:
 - First 3 letters of customer name (uppercase)
 - First 2 letters of product category (uppercase)
 - Current length of the customer's full name (as a number)
 - Reverse of product category (lowercase)
- Display the generated order ID

Example:

```
Customer: Suresh Patel
Category: Electronics

Order ID: SUREL13scinortcele
```

Test with:

- Customer: Mahesh, Category: Clothing
- Customer: Dinesh Kumar, Category: Books

Problem 3: LIC Premium Calculator (5 - 10 minutes)

LIC wants you to calculate monthly premium amounts based on policy details.

Requirements:

- Accept policyholder name, age, and policy type (Term/Endowment/ULIP)
- Accept sum assured amount (₹)
- Calculate monthly premium using these rates:
 - Term: 0.5% of sum assured per year / 12
 - Endowment: 2% of sum assured per year / 12
 - ULIP: 1.5% of sum assured per year / 12
- Add 10% loading charge if age > 45
- Display all details with monthly premium rounded to 2 decimal places

Test Cases:

```
Name: Mukesh Singh
Age: 50
Policy Type: Endowment
Sum Assured: ₹500000
```

Problem 4: Pune Metro Card Balance Checker (5 - 10 minutes)

Pune Metro needs a card balance tracking system.

Requirements:

- Start with an initial balance of ₹500
- Accept number of trips made
- For each trip, ask for:
 - Starting station name
 - Ending station name
 - Calculate fare as ₹10 per character in the longer station name
- Deduct fare from balance after each trip
- After all trips, display:
 - All trip details (from → to, fare)
 - Total fare spent
 - Remaining balance
- If balance becomes negative at any point, show "Insufficient Balance" and stop

Example Flow:

```
Initial Balance: ₹500
Number of trips: 2

Trip 1: Shivajinagar to Vanaz
Trip 2: Deccan to Kothrud

(Calculate and display accordingly)
```

Problem 5: Chennai Temperature Report (5 - 10 minutes)

Chennai Weather Department wants a weekly temperature analyzer.

Requirements:

- Accept temperatures for 7 days (Monday to Sunday) as a single string separated by commas
 - Example: "32,34,31,35,33,36,34"
- Convert the string into individual temperature values
- Calculate and display:
 - Highest temperature and on which day
 - Lowest temperature and on which day
 - Average temperature (rounded to 1 decimal)
 - Count how many days had temperature above 33°C
- Display a simple report format

Expected Output Format:

CHENNAI WEEKLY WEATHER REPORT

=====

Week Data: 32,34,31,35,33,36,34

Highest: 36°C on Friday

Lowest: 31°C on Wednesday

Average: 33.6°C

Days Above 33°C: 4 days

Submission Guidelines

1. Write each solution in a separate Python file
2. Name files as: `problem1.py`, `problem2.py`, etc.
3. Test your code with at least 2 different inputs
4. Ensure your code handles basic edge cases
5. Use meaningful variable names
6. Add comments where logic is complex

Evaluation Criteria:

- Correctness: Does the solution solve the problem? (40%)
- Code Quality: Clean, readable code with proper variable names (30%)
- Logic: Efficient use of operators, loops, and string methods (20%)
- Testing: Works with different inputs (10%)

Good Luck! 🍀

Remember: Focus on solving the problem first, then optimize if time permits. Partial solutions with correct logic earn more marks than incomplete attempts.