INNOMATICS
TECHNOLOGY HUB

Data Science Internship – February 2026

Internship Task Documentation

Task Instructions

1. Log in to your **LMS** and navigate to:
   Assessment & Task → Task 3: Python Programming Assignment Data Processing and Analysis
2. Open the **Google Form** provided in the task section to access your assigned Python problem.
3. Solve the problem using either **Jupyter Notebook** or **Google Colab**.
   Save your solution file in **.ipynb** format.
4. Upload (push) the `.ipynb` file to your **GitHub repository**.
   Ensure the repository link is in **HTTPS format** (e.g., `https://github.com/username/repository-name`).
5. Complete the **Google Form** by entering your required details and pasting your **GitHub repository HTTPS link**, then submit the form.

Submission Guidelines

- Your code must be **clean, well-structured, and properly organized**.
- Include **clear comments** explaining your logic wherever necessary.

Only submissions with a valid **GitHub HTTPS link submitted through the Google Form** will be considered for evaluation.

INNOMATICS TECHNOLOGY HUB

Python Programming Assignment Data Processing and Analysis

1. Introduction

This assignment focuses on applying Python programming concepts to real-world scenarios involving data analysis, validation, and processing. Students will use Python data structures such as dictionaries, lists, strings, and tuples to solve practical problems.

2. Problem Statements

Problem Statement 1: Employee Performance Bonus Eligibility

Description:
A company evaluates employee performance scores at the end of the year. You are given a dictionary containing employee names and their performance scores.

Requirements:

- Identify the highest performance score.
- Handle ties if multiple employees have the same highest score.
- Display all employees eligible for the top performance bonus.

Input:

```
employees = {
"Ravi": 92,
"Anita": 88,
"Kiran": 92,
"Suresh": 85
}
```

Expected Output:

```
Top Performers Eligible for Bonus: Ravi, Kiran (Score: 92)
```

Problem Statement 2: Search Query Keyword Analysis

Description:
An e-commerce website stores customer search queries. You are given a search query sentence entered by a user.

Requirements:

- Convert the input to lowercase.
- Ignore common punctuation.
- Count the frequency of each keyword.
- Display only keywords searched more than once.

Input:

"Buy mobile phone buy phone online"

Expected Output:

{'buy': 2, 'phone': 2}

Problem Statement 3: Sensor Data Validation

Description:
A factory collects sensor readings every hour. Each reading is stored in a list where the index represents the hour and the value represents the sensor reading.

Requirements:

- Identify readings that are even numbers (valid readings).
- Store them as (hour_index, reading_value) pairs.
- Ignore odd readings (invalid readings).

Input:

sensor_readings = [3, 4, 7, 8, 10, 12, 5]

Expected Output:

Valid Sensor Readings (Hour, Value):
[(1, 4), (3, 8), (4, 10), (5, 12)]

Problem Statement 4: Email Domain Usage Analysis

Description:
A company wants to analyze which email providers its users are using. You are given a list of employee email IDs.

Requirements:

- Count how many users belong to each email domain.
- Calculate the percentage usage of each domain.

Input:

```
emails = [
"ravi@gmail.com",
"anita@yahoo.com",
"kiran@gmail.com",
"suresh@gmail.com",
"meena@yahoo.com"
]
```

Expected Output:

```
gmail.com: 60%
yahoo.com: 40%
```

Problem Statement 5: Sales Spike Detection

Description:
A retail company tracks daily sales. Sudden spikes in sales may indicate promotions or unusual activity.

Requirements:

- Calculate the average daily sales.
- Detect days where sales are more than 30% above average.
- Display the day number and sale value.

Input:

sales = [1200, 1500, 900, 2200, 1400, 3000]

Expected Output:

Day 4: 2200
Day 6: 3000

INNOMATICS TECHNOLOGY HUB

Problem Statement 6: Duplicate User ID Detection

Description:
A system stores user IDs during registration. Duplicate IDs can cause data integrity issues.

Requirements:

- Identify duplicate user IDs.
- Display how many times each duplicate appears.

Input:

```
user_ids = ["user1", "user2", "user1", "user3", "user1", "user3"]
```

Expected Output:

```
user1 → 3 times
user3 → 2 times
```

3. Submission Instructions

- Write Python programs for all six problems.
- Ensure code is properly formatted and commented.
- Submit the assignment as a single file or repository as instructed.

End of Document