Department of Mathematics & Computer Science **BSc**

Data Visualisation with Python

Prerequisite Diagnostic Test — **MODEL ANSWERS**

Prepared by: Dr. D Bhanu Prakash **Total Marks: 25** **Time: 30 minutes**

SECTION A Python & Data Fundamentals

Q1. What will the following Python code print? Write your answer clearly, step by step. **[3 marks]**

```
data = [4, 7, 2, 7, 1, 9, 3]
data.sort()
print(data[0], data[-1], len(data))
```

Solution: Step by step:

1. `data = [4, 7, 2, 7, 1, 9, 3]` — list of 7 integers.
2. `data.sort()` — sorts in-place (ascending): `[1, 2, 3, 4, 7, 7, 9]`.
3. `data[0]` — first element = 1.
4. `data[-1]` — last element = 9.
5. `len(data)` = 7.

Output: `1 9 7`

Q2. Given the dictionary below, write **one line of Python** to extract a list of all city names. **[2 marks]**

```
cities = {"Hyderabad": 9.7, "Mumbai": 20.7, "Delhi": 32.9}
```

Solution:

```
city_names = list(cities.keys())   # result: ['Hyderabad', 'Mumbai', 'Delhi']
```

Alternatively (also valid): `city_names = [city for city in cities]`

Q3. A CSV file has columns: `Name, Age, Score, Grade`. A colleague hands you the first five rows using `pandas`. Write the **two pandas commands** you would use to: **(a)** load the file, **(b)** see the first 5 rows. **[2 marks]**

Solution:

```
(a) df = pd.read_csv('filename.csv')
(b) df.head()          # shows first 5 rows by default; or df.head(5)
```

Note: You would also need `import pandas as pd` at the top of your script.

SECTION B Chart Sense & Design Thinking

Q4. Match each dataset description to the **most appropriate chart type**. Write the letter in the box. **[3 marks]**

Chart types:	A. Bar chart	B. Scatter plot	C. Line chart	D. Pie chart	E. Histogram
C	1.	Monthly rainfall totals over 12 months (trend over time).			
E	2.	Distribution of exam scores of 200 students.			
B	3.	Relationship between study hours and test marks across 50 students.			
D	4.	Market share (%) of five mobile companies.			
A	5.	Comparing average salaries across four departments.			

Solution: **Rationale:** (C) Line chart is ideal for continuous trends over time. (E) Histogram shows frequency distribution of continuous data. (B) Scatter plot reveals correlations between two numeric variables. (D) Pie chart shows part-to-whole proportions summing to 100%. (A) Bar chart compares discrete categorical groups.

Q5. A fellow student creates a 3D bar chart with garish colours to show a simple two-column comparison. Name **two specific data-visualisation principles** they have likely violated, and briefly explain why each principle matters. **[3 marks]**

Solution: **Principle 1 — Data-ink ratio / Chartjunk (Tufte):** The 3D effect adds visual depth that carries no data information. It introduces distortion (bars at the back appear shorter), making accurate comparison harder. Every visual element should serve a purpose; decoration that distorts perception misleads the reader.

Principle 2 — Appropriate use of colour: Garish or arbitrary colours impose unnecessary cognitive load and may be inaccessible to colour-blind viewers. Colour should encode meaning (e.g., distinguish categories) or guide attention, not decorate. For a simple two-column comparison, two clearly distinct, accessible colours (or even a single neutral colour) suffice.

Bonus principle (if mentioned): **Chart type appropriateness** — a simple bar chart (2D) is the right tool for a two-group comparison; 3D adds complexity without benefit.

SECTION C Code & Interpretation

Q6. The following `matplotlib` snippet has **three errors**. Identify and correct each one. **[4 marks]**

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8]
plt.plot(x, y, color='crimson')
plt.title["Sales over Time"]
plt.xlabel("Month")
plt.show
```

Error 1: `y = [2, 4, 6, 8]` has only 4 elements but `x` has 5. Fix: `y = [2, 4, 6, 8, 10]`

Error 2: `plt.title["Sales over Time"]` uses square brackets. Fix: `plt.title("Sales over Time")`

Error 3: `plt.show` is missing parentheses; it references but never calls the function. Fix: `plt.show()`

Solution: **Corrected code:**

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]          # Error 1 fixed:  5 elements to match x
plt.plot(x, y, color='crimson')
plt.title("Sales over Time")   # Error 2 fixed:  parentheses not brackets
plt.xlabel("Month")
plt.show()                    # Error 3 fixed:  added ()
```

Q7. You are given a dataset of 1000 students' exam scores. Describe in **plain English** (no code needed) how you would use Python to: **(a)** check if there are any missing values, **(b)** visualise the score distribution, **(c)** investigate whether scores differ by gender. Name the specific function or chart you would use for each. **[5 marks]**

Solution: **(a) Checking for missing values:** After loading the data with `pandas`, call `df.isnull().sum()` on the DataFrame. This returns a count of missing (`NaN`) values per column. If the count for the "Score" column is zero, there are no missing values; otherwise you know how many need to be handled (filled or dropped with `df.dropna()` or `df.fillna()`).

(b) Visualising score distribution: Use a **histogram** (`plt.hist(df['Score'], bins=20)` in `matplotlib`, or `df['Score'].hist()` in `pandas`). A histogram groups scores into intervals and shows how many students fall in each, revealing whether the distribution is bell-shaped, skewed, has outliers, etc. Adding a KDE curve (`seaborn.histplot` with `kde=True`) further smooths the shape.

(c) Investigating whether scores differ by gender: Use a **grouped box plot** (also called side-by-side box plots) — e.g., `seaborn.boxplot(x='Gender', y='Score', data=df)`. This displays the median, spread, and outliers for each gender group simultaneously, making it easy to visually compare distributions. Alternatively, overlapping histograms or violin plots (`seaborn.violinplot`) show the full distribution shape per group.