Programming and Systems Development  
Final Lab Exam – December 2024

**During the exam**

The exam is open book, meaning you can refer to your practice materials and notes.

As in a normal examination, communication between candidates during the laboratory exam is strictly forbidden.

Any candidate who experiences a hardware or software problem during the examination should summon an invigilator at once.

**Overview**

This exam consists of two parts: part 1 requires you to write a Python program, and part 2 requires you to write a Java program. The instructions, allocated marks, and submission instructions are given below for each part. Note that the exam has a total of **80 marks**; each part has a total of **40 marks** allocated for correctness, as indicated.

**Part 1: Python Question**

This exam consists of three tasks. You are required to write the code for each task in the same Python source file.

## Python Task 1: Python Basics and Data Manipulation [13 Marks]

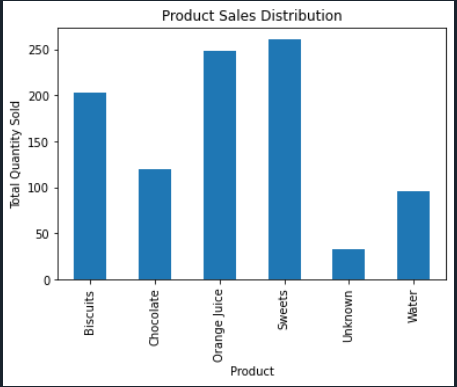
You have been provided with a CSV file **sales\_data.csv** containing sales data for a corner shop. The file contains the following columns: Date, Product, Quantity, Price, and Total. Your task is to write a Python program that performs the following operations:

### Task 1a: Data Loading and Preprocessing

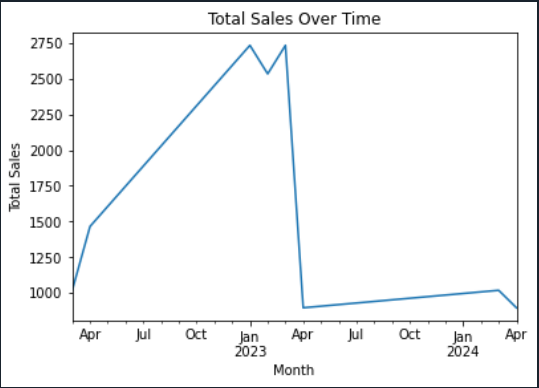
1. Load the Data: Load the CSV file into a Pandas DataFrame. **[2 mark]**
2. Data Cleaning:
   * Ensure there are no missing values. If any are found, fill them with appropriate values. **[3 mark]**
   * Convert the Date column to a datetime object. **[2 mark]**
   * Ensure the Total column correctly represents the product of Quantity and Price. If any discrepancies are found, correct them. **[2 mark]**

### Task 1b: Data Visualisation

1. Product Sales Distribution: Create a bar chart showing the total quantity sold for each product. **[2 mark]**



1. Sales Over Time: Create a line plot to show the trend of total sales over the months of 2023. The x-axis should represent the months, and the y-axis should represent the total sales**. [2 mark]**

****

## Python Task 2: Python Database Management [15 marks]

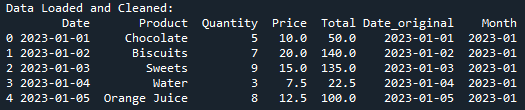
Problem: Managing Sales Data with SQLite. You are required to create a database to store and manage the sales data processed in Part 1.

### Task 2a: Database Creation and Data Insertion

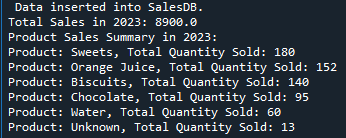
1. Create an SQLite database named **SalesDB**. **[2 mark]**
2. Create a table named **Sales** with appropriate columns to store the data (Date, Product, Quantity, Price, Total). **[3 mark]**
3. Insert the cleaned and processed **data from Part 1** into the Sales table. Ensure no duplicate entries are added. **[4 marks]**

### Task 2b: Querying the Database

1. Total Sales Calculation: Write a query to calculate the total sales for the year 2023 and display the result. **[3 marks]**



1. Product Sales Summary: Write a query to list each product and its total quantity sold in 2023. Display the result in descending order of quantity. **[3 marks]**



## Python Task 3: Basic Neural Network Implementation [12 marks]

In this task, you will define a simple fully connected neural network using PyTorch. You will train the network on synthetic data, evaluate its performance by tracking the loss during training, and visualize the results.

**Instructions:**

**Neural Network Definition:** Define a class SimpleNN that extends torch.nn.Module.

The network should have the following layers:

**Input Layer:** 10 input features.

**Hidden Layer 1:** A fully connected layer with 5 neurons and ReLU activation.

**Hidden Layer 2:** A fully connected layer with 3 neurons and ReLU activation.

**Output Layer:** A fully connected layer with 1 neuron and Sigmoid activation.

### Task 3a: Model Initialization **[4 marks]**

Instantiate the SimpleNN model.

Use torch.nn.MSELoss() as the loss function.

Use torch.optim.SGD as the optimizer with a learning rate of 0.01.

### Task 3b: Data Generation **[2 mark]**

Generate synthetic data with 100 samples, each having 10 features. The target should be a tensor with 100 values.

Use torch.randn to generate the synthetic input data and target values.

### Taks 3c: Training the Network **[3 marks]**

Train the network for 20 epochs.

During each epoch, compute the loss, perform backpropagation, and update the model's weights.

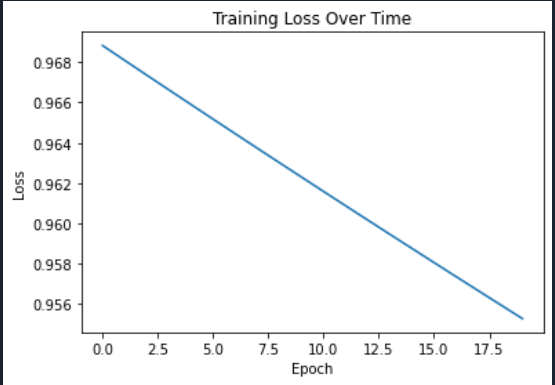
Record the training loss for each epoch.

### Task 3d: Loss Visualisation **[3 marks]**

Plot the training loss over epochs using matplotlib.

**Example Output:**

Your plot should display the training loss decreasing over time, indicating that the model is learning.



**Python submission**

Ensure your submission file is named exactly as it is supposed to (i.e. **your\_name**\_**python\_task.py)**

Go to your Moodle account, and submit your file at the appropriate submission link:

*Moodle > Programming and System Design > Lab Exam > Python Submission*

# **Part 2– Java question**

You are going to make a system that manages a pet care service: that is, a service that looks after animals when the owners may be away. Each animal is housed in its own **enclosure** at the service.

The tasks will get harder as you progress, and build on each other, so try to ensure that you complete each stage before moving on. Be sure to read through the entire specification before you begin to get a sense of all the features that might be needed.

In each task, marks will be awarded for clean code and sensible object-oriented design choices as well as functionality.

## Java Task 1: Representing an animal (10 marks)

We have provided initial versions of an enumerated type **AnimalSize** and the skeleton of the **Animal** class that you should use for this task, which are both in the **petcare** package. You must use these classes: **do not change the name of the classes or the package.**

For our purposes, an animal has the following properties:

* **Name**: Each animal has a name which must be at least 3 characters long.
* **Size**: An animal can be one of three sizes: small, medium, or large.
* **Comfortable temperature**: Animals need to be kept within a specific temperature range to be comfortable. Each animal will have a range they must stay within, which must be expressed as a lower and upper bounding integer, both in the range 0 to 50 – an animal is comfortable in an enclosure as long as the temperature is between those two values.

### Task 1a [4 marks]

Write some code to represent an animal so it can be stored in the system. You should use the provided **Animal** class as the starting point for this class, but the specific details of the implementation are up to you. Make sure to include a constructor that sets all properties.

### Task 1b [2 marks]

Add getter and setter methods for all the values and override the **toString()** method and **equals()/hashCode()** methods. The **equals** method should return true when two animals are the same size and have the same name.

### Task 1c [4 marks]

Ensure that all the values that are stored are valid, following the specification above. If someone tries to create or update an animal with invalid values, then an IllegalArgumentException should be thrown, and the change should not take effect.

## Java Task 2: Enclosures and checks (15 marks)

An **enclosure** is a cage or pen that the pet care service maintains to keep one of the animals they are looking after in. Enclosures have:

* **Size**: Each specific enclosure is rated to look after animals up to the size of small, medium or large.
* **Temperature**: Each enclosure has a specific temperature that it is maintained at.
* **Running costs**: Each enclosure costs a certain amount to run each day, which can be expressed as an integer.
* **Current occupant**: If there is an animal in the enclosure, they should be stored, otherwise the value of occupant should be **null**.

### Task 2a [4 marks]

Create a class called **Enclosure** that stores this information. The class should have a constructor that takes in a size, temperature, and running costs when it is created and set the occupant to null. The class should have getter methods for all the variables but no setters.

### Task 2b [5 marks]

Write a method with the signature:

public boolean checkCompatibility(Animal animal)

When passed an animal, the method checks whether the enclosure can take the animal and returns true if it can. For an animal to be compatible with an enclosure, the animal must be of the same size or smaller than the size an enclosure is rated for, and the enclosure’s temperature should fall within the animal’s comfort range.

### Task 2c [6 marks]

Write a pair of methods with the signatures:

public void addAnimal(Animal animal)

public void removeAnimal()

When called, these methods should, respectively, add an animal to the Enclosure if it is compatible and does not already have an occupant, and remove the animal currently in the enclosure. If the animal cannot be added, then throw an IllegalArgumentExcepion. If **removeAnimal** is called on an empty enclosure, nothing should happen.

## Java task 3: The pet care service (15 marks)

The pet care service manages a collection of enclosures, and animals are allocated to enclosures as they arrive. Additional enclosures can be added to the service as needed.

### Task 3a [3 mark]

Write a class called **PetService** to represent the pet care service, which stores a single piece of information: the collection of enclosures that it manages, which should be initially empty.

### Task 3b [4 marks]

Add methods to the **PetService** class to maintain and print the enclosures that it manages.  
  
Add a method to the class with the following signature:

public void addEnclosure(Enclosure enclosure)

When called, this method should add the given enclosure to the enclosures that the pet service has.

Add a second method with the signature:

public void printAllEnclosures()

This should print out the details of all the enclosures that the service maintains.

### Task 3c [4 marks]

In this task you will handle adding an animal into the pet service’s care.  
  
Add a method with the signature:

public boolean allocateAnimal(Animal animal)

When called, this method should find the best enclosure to allocate an animal based on the following criteria:

* The enclosure must meet all the requirements for the animal (size, temperature).
* The enclosure cannot have another animal staying within it.
* If more than one enclosure meets the requirements, the animal should be allocated to the cheapest one (i.e., the one with the lowest running cost), and the method should return **true**.
* If no appropriate enclosure is available, then no changes should be made to the pet service and the method should return **false**.

### Task 3d [4 marks]

Finally, add a method with the signature:

public void removeAnimal(Animal animal)

When called, this method should search for a matching animal within the enclosures of the pet service. If a matching animal exists, it should be removed from its enclosure. If the animal does not exist in the system, then the method should throw an IllegalArgumentException.