التكنولوجيا التطبيقية
APPLIED TECHNOLOGY

مدارس we للتكنولوجيا التطبيقية

# Data structure and Algorithm
## Unit 14
الصف الثاني

IT Department
Web Programming

**Data structure and Algorithm**
Workshops

**2st**

**2024 - 2025**

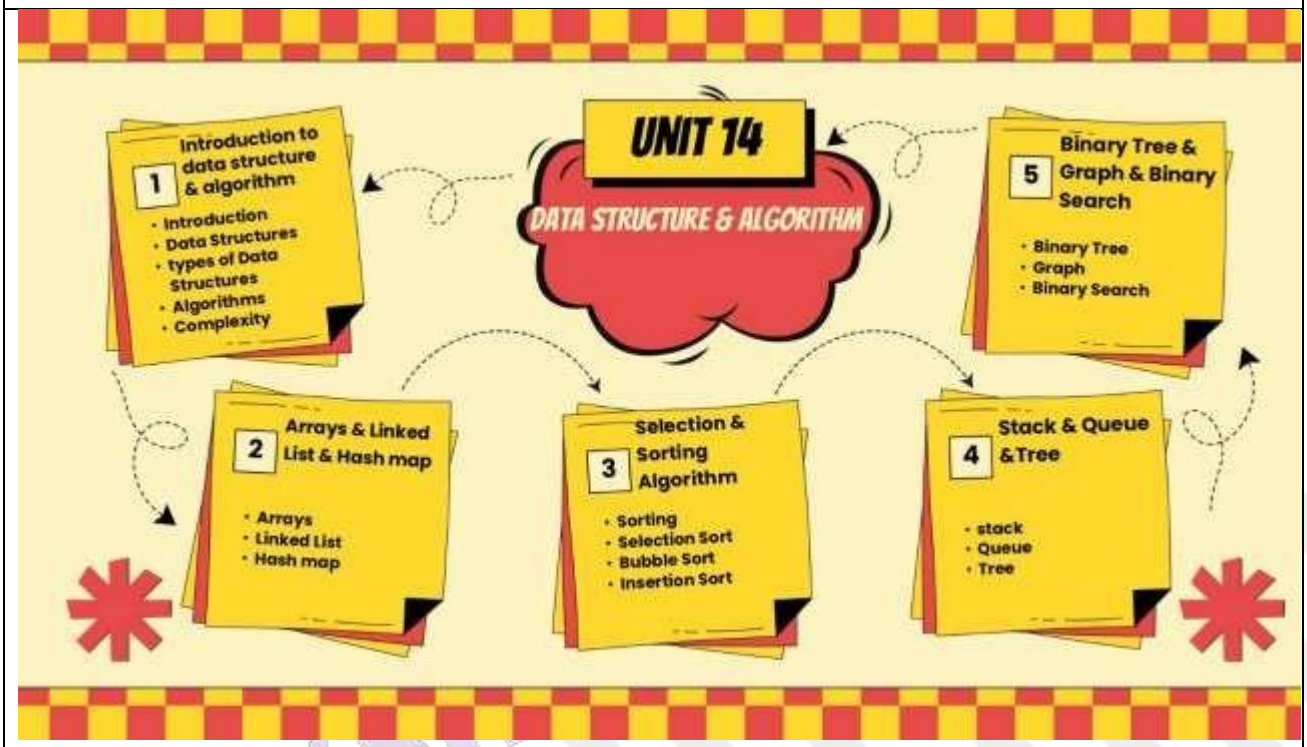
# Data structure and Algorithm

Workshops

| Unit | 14 |
|---|---|
| Name | Data structures and algorithms |
| **Goals / Outcomes** | ➢ **Remembering**<br>1. Define key concepts in data structures and algorithms, such as arrays, linked lists, hash maps, stacks, queues, trees, graphs, and sorting algorithms.<br>2. Identify different types of data structures and their purposes.<br>3. Recall basic sorting techniques like selection sort and bubble sort.<br>4. Memorize the key operations (e.g., insertion, deletion, traversal) for various data structures.<br><br>➢ **Understanding**<br>1. Explain the role of data structures in organizing and managing data efficiently.<br>2. Describe the differences and use cases for arrays, linked lists, and hash maps.<br>3. Understand the logic behind sorting algorithms and how they reorder elements.<br>4. Illustrate the concept of stacks and queues and their real-life applications.<br>5. Interpret how binary trees and graphs are structured and how binary search operates in sorted data.<br><br>➢ **Applying**<br>1. Implement arrays, linked lists, hash maps, stacks, queues, and trees in code.<br>2. Write and execute sorting algorithms like bubble sort and insertion sort.<br>3. Develop programs to search for data efficiently using binary search.<br>4. Apply graphs to solve problems such as finding connections or shortest paths.<br>5. Use data structures in solving practical problems, like organizing data in a program.<br><br>➢ **Analyzing**<br>1. Compare the efficiency of different data structures for specific tasks (e.g., when to use a linked list versus an array). |

<table>
<tr><td></td><td colspan="2">2. Analyze the performance of sorting algorithms based on time complexity.<br>3. Evaluate the suitability of binary trees versus graphs for representing hierarchical versus networked data.<br>4. Break down the steps of binary search and analyze its efficiency compared to linear search<br><br>➢ <u>Evaluating</u><br>1. Assess the efficiency and effectiveness of various data structures and algorithms in solving real-world problems.<br>2. Evaluate trade-offs between time and space complexity for different solutions.<br>3. Judge the appropriateness of sorting techniques for specific data sets.<br>4. Critique the design of a data structure or algorithm implementation for potential improvements.<br><br>➢ <u>Creating</u><br>1. Design and implement custom data structures and algorithms to solve complex problems.<br>2. Create a real-world application that utilizes a combination of arrays, hash maps, and trees.<br>3. Develop a visualization or simulation of sorting algorithms or graph traversal methods.<br>4. Build a program to represent and process hierarchical or networked data using binary trees or graphs.</td></tr>
<tr><td rowspan="3">Knowledge</td><td>Code</td><td>Description</td></tr>
<tr><td>TPK23</td><td>Data structures</td></tr>
<tr><td>TPK24</td><td>Algorithms</td></tr>
<tr><td rowspan="6">Skill</td><td>Code</td><td>Description</td></tr>
<tr><td>TPC6.1</td><td>Using data structures for code optimization and problem-solving</td></tr>
<tr><td>TPC6.2</td><td>Solving problems using algorithms</td></tr>
<tr><td>TPC6.3</td><td>Create classes with custom methods, including initializers and decorated properties</td></tr>
<tr><td>TPC6.4</td><td>Analyze object-based design patterns</td></tr>
<tr><td>TPC6.5</td><td>Handle and produce errors (builtin or custom) to process or signal failure</td></tr>
</table>

## Unit Preface



# Workshop 1

**Question 1:** You are creating a simple order management system for a restaurant. You have a list of initial customer orders. Your task is to write a Python program that allows you to:

1. Start with a predefined list of customer orders.
2. Add a new order to the existing list.
3. Remove a specific order from the list.
4. Print the updated list of orders, displaying each order on a separate line with a preceding "-".

**Expected Output:**

```
Order List:
- Pizza
- Salad
- Drinks
- Ice Cream
- Pasta
```

**Question 2:** You are a teacher managing student grades. You have a list of initial student grades. Your task is to write a Python program that:

1. Starts with a predefined list of student grades.
2. Calculates the average of these initial grades.
3. Add a new grade to the list.
4. Prints the updated list of grades.
5. Prints the calculated average grade, formatted to two decimal places.

Initial Data:

```
grades = [85, 90, 78, 92, 88]
```

**Expected Output:**

```
Student Grades: [85, 90, 78, 92, 88, 95]
Average Grade: 87.60
```

**Question 3:**

You are managing the inventory for a small grocery store. You need to keep track of the quantity of each product in stock. You've decided to use a dictionary to store this information, where the keys are the product names and the values are the corresponding quantities.

Your task is to write a Python program that:

1. Starts with a predefined dictionary representing the initial product inventory.
2. Updates the quantity of a specific product (e.g., "Apple") by reducing it by a certain amount (e.g., 10).
3. Adds a new product (e.g., "Strawberry") to the inventory with its initial quantity (e.g., 25).
4. Prints the current inventory, displaying each product and its quantity in a user-friendly format.

**Initial Data:**

```
"Apple": 50, "Banana": 30, "Grapes": 20,  "Orange": 40
```

**Requirements:**

- Use a dictionary to store the product inventory.
- Update the quantity of "Apple" by subtracting 10.
- Add "Strawberry" to the inventory with a quantity of 25.
- Use a for loop to iterate through the dictionary and print each product and its quantity in the format: {product}: {quantity} (e.g., Apple: 40).

**Expected Output:**

```
Current Inventory:
Apple: 40
Banana: 30
Grapes: 20
Orange: 40
Strawberry: 25
```

**Question 4:** You are managing student data and want to store information about each student, including their age and grade level. You've decided to use a nested dictionary in Python, where the keys are student names and the values are dictionaries containing the student's age and grade.
Your task is to write a Python program that:

1. Starts with a predefined dictionary containing initial student information.
2. Adds a new student to the dictionary with their age and grade.
3. Prints the information for all students, displaying each student's name, age, and grade level in a clear format.

**Initial Data:**

```
students = {
    "Ahmed": {"Age": 15, "Grade": "2th"},
    "Fatima": {"Age": 14, "Grade": "1th"},
    "Khalid": {"Age": 16, "Grade": "3th"}
}
```

**Requirements:**

- Use a nested dictionary as described above.
- Add a new student named "Layla" with an age of 14 and grade "1th" to the students dictionary.
- Use a for loop to iterate through the students dictionary.
- For each student, print their name, age, and grade in the format: Name: {name}, Age: {age}, Grade: {grade}.

**Expected Output:**

```
Student Information:
Name: Ahmed, Age: 15, Grade: 2th
Name: Fatima, Age: 14, Grade: 1th
Name: Khalid, Age: 16, Grade: 3th
Name: Layla, Age: 14, Grade: 1th
```

# Workshop 2

**Question 1:** You are managing student data and want to store detailed information about each student, including their name, age, and grades in different subjects. You've decided to use a list of dictionaries in Python, where each dictionary represents a student and contains their name, age, and a nested dictionary of grades.

Your task is to write a Python program that:

1. Starts with a predefined list of dictionaries, each containing initial student information (name, age, and grades).
2. Adds a new student to the list, including their name, age, and grades.
3. Prints the information for all students in a clear and organized format, including their name, age, and individual grades for each subject

**Question2:** You are tasked with creating a simple linked list to store a sequence of names. Write a Python program that:

1. Defines a Node class with two attributes: data (to store a name) and next (to store a reference to the next node).
2. Creates three Node objects, storing the names "Ahmed", "sara", and "Mohamed" respectively.
3. Links these nodes together in the order they were created, forming a linked list: Ahmed -> sara -> Mohamed.
4. Traverses the linked list and prints the data of each node, separated by " -> ".
5. Prints "End" at the end of the list traversal.

**Requirements:**

- Implement the Node class with the specified attributes.
- Create the Node objects and link them together.
- Use a while loop to traverse the linked list.
- Print the output in the exact format shown in the example output.

**Expected Output:**

```
Ahmed -> sara -> Mohamed -> End
```

**Question3: Write a Python program to simulate a train using a linked list data structure.**

**Requirements:**

1. **TrainCar Class:**
   - Create a TrainCar class to represent a single train car.
   - Each TrainCar should have an attribute number to store its unique identification.
   - Each TrainCar should have an attribute next to point to the next TrainCar in the train (or None if it's the last car).
2. **Train Class:**
   - Create a Train class to represent the entire train.
   - The Train class should have an attribute head to point to the first TrainCar in the train.
   - **Implement the following methods:**
     - **add_car(number):** Adds a new TrainCar with the given number to the end of the train.
     - **display():** Prints the numbers of all cars in the train, starting from the head, in a clear and readable format (e.g., " 1 ->  2 -> 3 -> End of Train").

# Workshop 3

**Question:** You are required to implement a Python function called bubble_sort that sorts a list of integers in **ascending order** using the **Bubble Sort algorithm**. The function should return the sorted list. Use **nested loops** to perform comparisons and swaps. Ensure the function can handle both **positive and negative integers**.

**Requirements:**

1. Define the bubble_sort function that takes a list of integers (arr) as input.
2. Use **nested loops** to compare adjacent elements and swap them if they are in the wrong order.
3. Return the **sorted list** as the output of the function.
4. Handle both positive and negative integers.
5. Test the function with the following list:
   numbers = [5, 3, 8, 4, 6]
6. Print the sorted list after calling the bubble_sort function.
7. The expected output for the test case is:
   [3, 4, 5, 6, 8]

WORKSHOP

# Workshop 4

**You are required to implement the Stack in Python**
**Question1: How can you use a stack to simulate navigation between different menus in a mobile app?**

**Question2: How can you use a stack to keep track of viewed photos in a photo viewer app?**

**You are required to implement the Queue in Python**
**Using Queue to Manage Print Queue**

**Question 3: How can I use Queue to manage print queues, where jobs are processed in the order they are added?**

**Using Queue to Manage Customer Queue in a Bank**
**Question 4: How can Queue be used to manage customer queue in a bank, where customers are served in the order they arrived?**

**Question 5: How can you use a tree in Python to represent product categories in an online store?**

# Workshop 5

**Question1: You are tasked with representing a store's product categories using a tree data structure in Python. Write a program that:**

- Create a Node class with data and children attributes, and an add_child() method.
- Implement a print_tree() function for indented, recursive visualization.
- Build a tree: "Store" (root), "Electronics" & "Clothing" (children), "Phones" & "Computers" (Electronics children), "Men" & "Women" (Clothing children).
- Call print_tree() to display the tree structure.

**Requirements:**

- Implement the Node class and add_child() method.
- Implement the print_tree() function with proper indentation.
- Create the specified tree structure.
- Print the tree structure to the console.

**Expected Output:**

```
└── Store
    └── Electronics
        └── Phones
        └── Computers
    └── Clothing
        └── Men
        └── Wome
```

**Creating a Graph to Represent a Social Network**

**Question2: How can you use a graph in Python to represent a social network of friends?**