Computational Structures

Class : 11th

Subject : Computer Science and Entrepreneurship

Chapter : 4

1. Which function is used to add an item at the end of the list in Python?

a) insert()
b) append() ✅ **(Correct Answer)**
c) remove()
d) pop()
🗒 **Explanation:** `append()` function adds an element at the **end** of a list. Example:

```
my_list = [1, 2]
my_list.append(3)  # Result: [1, 2, 3]
```

---

2. What does the 'in' keyword do when used with a Python list?

a) Adds an item to the list.
b) Removes an item from the list.
c) Checks if an item exists in the list. ✅ **(Correct Answer)**
d) Returns the length of the list.
🗒 **Explanation:** The `in` keyword is used to **check membership** in a list. Example:

```
if 5 in [1, 2, 3, 5]:  # Returns True
```

---

3. Which operation removes an item from the top of the stack?

a) Push
b) Pop ✅ **(Correct Answer)**
c) Peek
d) Add

🗒 **Explanation:** `pop()` removes the top item from the stack. Stack uses **LIFO** (Last In, First Out).

---

4. Which operation is used to add an item to a queue?

a) Dequeue
b) Peek
c) Enqueue ✅ **(Correct Answer)**
d) Remove
🗒 **Explanation:** `enqueue` means to **add** an item at the **end** of the queue (FIFO).

---

5. Which of the following is true about the height of a tree?

a) The height is the number of edges from the root to the deepest node ✅ **(Correct Answer)**
b) The height is the number of nodes from the root to the deepest node
c) The height is the number of children of the root node
d) The height is always equal to the number of nodes in the tree
🗒 **Explanation:** Tree height = number of **edges** on the longest path from the root to a leaf.

---

6. For which scenario would a graph data structure be most appropriate?

a) Managing a to-do list
b) Modeling a line of customers in a store
c) Representing connections in a social network ✅ **(Correct Answer)**
d) All of the above
🗒 **Explanation:** Graphs are ideal for showing **relationships**, such as friends/followers in social networks.

Short Question

1. Explain how the `extend()` function works in Python lists. Provide an example.

Answer:
The `extend()` function is used to **add multiple elements** from one list to the end of another list. It **extends** the original list by appending elements from another iterable (like list, tuple).

Example:

```
list1 = [1, 2, 3]
list2 = [4, 5]
list1.extend(list2)
print(list1)  # Output: [1, 2, 3, 4, 5]
```

🗒 `extend()` changes the original list by adding each item from the second list.

---

2. Explain the potential issues which could arise when two variables reference the same list in a program? Provide an example.

Answer:
If two variables refer to the **same list**, any change made through one variable will **also affect** the other, because they point to the **same memory location**.

Example:

```
a = [1, 2, 3]
b = a      # b and a refer to the same list
b.append(4)
print(a)   # Output: [1, 2, 3, 4]
```

🗒 This can cause bugs if the programmer **does not intend** both variables to share the same data.

---

3. Define a stack and explain the Last-In, First-Out (LIFO) principle.

Answer:
A **stack** is a linear data structure where **the last element added is the first to be removed**. This is called the **LIFO (Last In, First Out)** principle.

Real-life example: Stack of plates – the last plate placed on top is the first one to be taken off.

Python example:

```
stack = []
stack.append('A')  # push
stack.append('B')
stack.pop()        # removes 'B'
```

---

4. Differentiate between the Enqueue and Dequeue operations of a queue.

Answer:

Operation	**Meaning**	**Direction**
Enqueue	Adds an element to the queue	At the **rear**
Dequeue	Removes an element from queue	From the **front**

📝 Queue follows **FIFO (First In, First Out)** rule.

---

5. Name two basic operations performed on a stack.

Answer:

1. **Push** – Add an element to the top of the stack.
2. **Pop** – Remove the top element from the stack.

📝 Optional: `Peek` is also used to view the top element without removing it.

---

6. What is the difference between `enqueue()` and `dequeue()`?

Answer:

- **enqueue()** is used to **insert** an element at the **end** of a queue.
- **dequeue()** is used to **remove** an element from the **front** of the queue.

📝 Example:

```
queue = []
queue.append('A')       # Enqueue
queue.pop(0)            # Dequeue
```

Long Question

Discuss the dynamic size property of lists in Python. How does this property make lists more flexible?

Answer:
In Python, lists are **dynamic in size**, which means you can **add or remove items**

anytime during program execution. You don't need to define a fixed size like in arrays in some other languages (e.g., C or Java).

Advantages:

- **Flexible memory usage**
- **Easy to grow/shrink** based on need
- Supports many **built-in functions** like `append()`, `remove()`, `extend()`, etc.

Example:

```
my_list = []               # Empty list
my_list.append(10)         # Add element
my_list.extend([20, 30]) # Add multiple elements
print(my_list)             # Output: [10, 20, 30]
```

📄 This dynamic nature makes Python lists ideal for managing data that **changes in size**.

---

8. Explain the operations on stack with real-life example and Python code.

Answer:
A **stack** is a data structure that follows **LIFO (Last In, First Out)** rule.

Basic Operations:

1. **Push** – Add item to the top of stack
2. **Pop** – Remove item from top of stack
3. **Peek** – View top item without removing

Real-Life Example:

Imagine a stack of plates:

- The **last plate** placed on top is the **first to be taken out**.

Python Code Example:

```
stack = []

# Push operation
stack.append("Book1")
stack.append("Book2")
print(stack)  # Output: ['Book1', 'Book2']

# Pop operation
stack.pop()   # Removes 'Book2'
print(stack)  # Output: ['Book1']
```

```
# Peek operation
print(stack[-1])  # Output: 'Book1'
```

🗒 Stack is useful in undo-redo systems, browser history, etc.

9. Write a simple program to implement a queue (insertion and deletion).

Answer:

```
# Queue using list
queue = []

# Enqueue (Insert)
queue.append("Person1")
queue.append("Person2")
print("Queue after enqueuing:", queue)

# Dequeue (Remove)
first = queue.pop(0)
print("Dequeued:", first)
print("Queue after dequeuing:", queue)
```

Output:

```
Queue after enqueuing: ['Person1', 'Person2']
Dequeued: Person1
Queue after dequeuing: ['Person2']
```

🗒 Queue follows **FIFO (First In, First Out)**.

10. Define Tree and explain its properties

Answer:
A **tree** is a hierarchical data structure that consists of **nodes** connected by **edges**.

Properties of a Tree:

- **Root:** The topmost node of the tree
- **Parent and Child:** Nodes connected downward from the root
- **Leaf Node:** A node with no children
- **Height:** Number of edges from root to the deepest leaf
- **Subtree:** Any node with its children can be considered a tree

- **No cycles:** Trees cannot have loops like graphs

Example Tree:

```
    A          ← Root
   / \
  B   C        ← Children of A
     / \
    D   E      ← Children of C
```

🗒 Trees are used in file systems, databases, decision making, etc.

---

11. What is a graph? Explain differences between directed and undirected graphs.

Answer:
A **graph** is a collection of **nodes (vertices)** and **edges** where the edges connect pairs of nodes.

Types of Graphs:

Feature	Directed Graph	Undirected Graph
Edge direction	One-way (A → B)	Two-way (A — B)
Representation	Arrows between nodes	Simple lines between nodes
Example use-case	Social media followers	Friendship or road maps

---

Example:

- **Directed Graph:**
  A → B (A follows B)
- **Undirected Graph:**
  A — B (A and B are friends)

🗒 Graphs are used in networking, maps, social networks, and more.