Unit 4 - Lesson 1A Reference: Adding & Removing Lists

Period 1 - Review of collections with emphasis on list operations

---

Adding Elements

1. insert(index, item) - Add at Specific Position

```
playlist = ["Song A", "Song B", "Song D"]
playlist.insert(2, "Song C")  # Insert "Song C" at index 2
print(playlist)
# Output: ['Song A', 'Song B', 'Song C', 'Song D']
```

2. extend(list) - Add Multiple Items

```
playlist = ["Song A", "Song B"]
new_songs = ["Song C", "Song D", "Song E"]
playlist.extend(new_songs)
print(playlist)
# Output: ['Song A', 'Song B', 'Song C', 'Song D', 'Song E']
```

3. append() vs extend() - THE DIFFERENCE!

append() adds the LIST itself:

```
playlist1 = ["A", "B"]
playlist1.append(["C", "D"])  # Adds the LIST itself
print(playlist1)
# Output: ['A', 'B', ['C', 'D']]  # <- Nested list!
```

extend() adds items FROM the list:

```
playlist2 = ["A", "B"]
playlist2.extend(["C", "D"])  # Adds items FROM the list
print(playlist2)
# Output: ['A', 'B', 'C', 'D']  # <- Flat list!
```

Removing Elements

1. remove(value) - Delete by Value

Removes FIRST occurrence only:

```
playlist = ["Song A", "Song B", "Song C", "Song B"]
playlist.remove("Song B")  # Removes FIRST "Song B"
print(playlist)
# Output: ['Song A', 'Song C', 'Song B']
```

2. DANGER: remove() Crashes if Item Doesn't Exist!

```
playlist = ["Song A", "Song B", "Song C"]
playlist.remove("Song Z")  # NOT in list!
# ❌ ValueError: list.remove(x): x not in list
```

3. THE SAFE WAY - Always Check First!

```
playlist = ["Song A", "Song B", "Song C"]
song_to_remove = "Song Z"

if song_to_remove in playlist:
    playlist.remove(song_to_remove)
    print("✅ Removed!")
else:
    print("⚠️ Song not found")
```

4. pop() - Remove Last Item AND Return It

```
playlist = ["Song A", "Song B", "Song C"]
last_song = playlist.pop()
print(f"Removed: {last_song}")  # Removed: Song C
print(f"Playlist now: {playlist}")  # Playlist now: ['Song A', 'Song B']
```

5. pop(index) - Remove from Any Position AND Return It

```
playlist = ["Song A", "Song B", "Song C", "Song D"]
second_song = playlist.pop(1)
print(f"Removed: {second_song}")  # Removed: Song B
print(f"Playlist now: {playlist}")  # Playlist now: ['Song A', 'Song C', 'Song D']
```

6. clear() - Delete Everything

```
playlist = ["Song A", "Song B", "Song C"]
playlist.clear()
print(f"After clear: {playlist}")
# Output: []
```

Key Patterns

Pattern 1: Safe Remove Function

```
def safe_remove(playlist, song):
    """Remove song from playlist safely."""
    if song in playlist:
        playlist.remove(song)
        return True
    return False
```

Pattern 2: Safe Insert at Position

```
def safe_insert(playlist, index, song):
    """Insert song at index if index is valid."""
    if 0 <= index <= len(playlist):
        playlist.insert(index, song)
        return True
    return False
```

Which Method to Use?

Adding

- **append(item)** → Add ONE item to end
- **insert(index, item)** → Add item at SPECIFIC position
- **extend(list)** → Add MULTIPLE items from another list

Removing

- **remove(value)** → Know the VALUE, remove first occurrence **(CHECK FIRST!)**
- **pop()** → Remove LAST item, get it back
- **pop(index)** → Remove at SPECIFIC position, get it back
- **clear()** → Delete EVERYTHING

Defensive Programming - THE GOLDEN RULE

✅ **ALWAYS check before using remove() or index():**

```
if item in my_list:
    my_list.remove(item)
```

This prevents ValueError crashes!

Testing Pattern

```python
def remove_duplicates(playlist):
    """Remove duplicate songs from playlist."""
    result = []
    for song in playlist:
        if song not in result:
            result.append(song)
    return result

if __name__ == "__main__":
    # Test example
    test = ["Song A", "Song B", "Song A", "Song C"]
    result = remove_duplicates(test)
    expected = ["Song A", "Song B", "Song C"]
    print(f"Test: {'✅ PASS' if result == expected else '❌ FAIL'}")
    print(f"Result: {result}")
```

Summary

Key Takeaways:

- ✅ `insert()` adds at specific positions
- ✅ `extend()` adds multiple items (not `append()`!)
- ✅ **ALWAYS check** before using `remove()`
- ✅ `pop()` removes AND returns the item
- ✅ Defensive programming prevents crashes

Next: Finding, organizing, and references!