Day 07

Python Lists & Introduction to Dictionaries

Today's Agenda

1

Python Lists Deep Dive

- List creation and initialization
- Indexing and slicing operations
- List methods for manipulation
- Nested lists and advanced operations

2

List Best Practices

- Common pitfalls to avoid
- Performance considerations
- When to use lists vs. other data structures

3

Introduction to Dictionaries

- Key-value pair concept
- Dictionary creation and initialization
- Basic operations: accessing, updating, and removing items

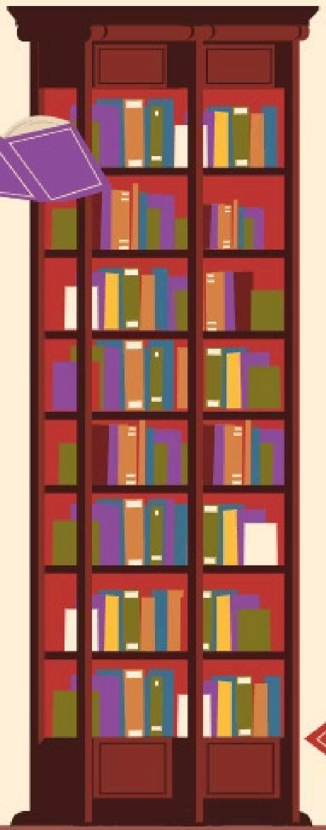
Data Structures

List

An ordered collection of values. Usually used to store a "list" of similar data. One example might be an attendance roster. It's usually in order alphabetically (but it doesn't have to be) and stores a list of names!

Dictionary

An unordered collection of key/value pairs. Usually used to store relational data. One example is a phone book or a contact list in a phone. Each name has a number associated with it!



Python Lists: Understanding the Fundamentals

What Are Lists?

- **Ordered collections** of items
- **Mutable** (can be modified after creation)
- Can store **mixed data types** (integers, strings, objects, even other lists)
- **Zero-indexed** (first element is at position 0)

Creating Lists

```
# Empty list
empty_list = []
empty_list2 = list()

# List with values
numbers = [1, 2, 3, 4, 5]
mixed = [42, "hello", True, 3.14]
```

Lists are like ordered containers that can hold any type of Python object. Think of them as versatile, resizable arrays.

Working with Python Lists: Operations & Examples

Indexing & Slicing

```
fruits = ["apple", "banana", "cherry", "date"]
print(fruits[0])      # Output: apple
print(fruits[-1])     # Output: date
print(fruits[1:3])    # Output: ['banana', 'cherry']
```

Adding Elements

```
fruits.append("elderberry")  # Add to end
fruits.insert(1, "blueberry")  # Insert at index 1
fruits.extend(["fig", "grape"])  # Add multiple items
```

Removing Elements

```
fruits.remove("banana")  # Remove by value
popped = fruits.pop()    # Remove & return last item
del fruits[0]            # Delete by index
```

Sorting & Other Operations

```
fruits.sort() # Sort in place
new_list = sorted(fruits) # Return sorted copy
print(len(fruits)) # Count items
print("apple" in fruits) # Check membership
```

Introduction to Python Dictionaries

What Are Dictionaries?

Dictionaries are unordered collections of key-value pairs that provide lightning-fast lookups based on keys.

- Each key must be unique and immutable
- Values can be any data type
- Implemented as hash tables for O(1) lookups

Basic Dictionary Operations

```
# Creating a dictionary
student = {
    "name": "Alice",
    "age": 21,
    "courses": ["Math", "CS"]
}

# Accessing values
print(student["name"])  # Output: Alice

# Adding/updating entries
student["gpa"] = 3.9
student["age"] = 22

# Removing entries
del student["courses"]
```

When to use dictionaries:

- When you need to map keys to values
- For fast lookups by key (not index)
- When data has a natural key-value structure
- To represent real-world objects with attributes

Unlike lists, dictionaries use **curly braces {}** and require keys for accessing values instead of numeric indices.