Chapter 2: Lists, Dictionaries, and Structuring Data

The List Data Type

Definition: A list is a collection of items in a particular order, enclosed in square brackets `[]`.

Characteristics:
- Lists are mutable (can be changed).
- Elements can be of any data type (e.g., integers, strings, other lists).
- Indexing and slicing are supported.

Example:

```
fruits = ['apple', 'banana', 'cherry']
print(fruits[0])  # Output: apple
```

Working with Lists

Adding Elements

- `append(item)` adds an item to the end.
- `insert(index, item)` adds an item at a specified position.
- **Removing Elements**:
- `remove(item)` removes the first occurrence of the item.
- `pop(index)` removes and returns an item at a specific index (default is the last item).

-Other Operations
- `sort()` sorts the list in place.
- `reverse()` reverses the list in place.

Examples

 # Adding Elements

```
numbers = [1, 2, 3]
numbers.append(4)
print(numbers)  # Output: [1, 2, 3, 4]

# Removing Element
numbers.remove(2)
print(numbers)  # Output: [1, 3, 4]
```

Augmented Assignment Operators

Definition: Simplify the process of modifying a variable.
Common Operators
- `+=`: Add and assign.
- `-=`: Subtract and assign.
- `*=`: Multiply and assign.

Example

```
x = 5
x += 3  # Equivalent to x = x + 3
print(x)  # Output: 8
```

List Methods
Common Methods
- `index(value)`: Returns the index of the first occurrence of the value.
- `count(value)`: Returns the count of the specified value.
- `extend(iterable)`: Appends elements from another iterable.

Example

```
colors = ['red', 'blue', 'green', 'blue']
print(colors.index('blue'))  # Output: 1
print(colors.count('blue'))  # Output: 2
```

Example Program: Magic 8 Ball with a List
Code

```
import random

responses = [
    "It is certain",
    "Reply hazy, try again",
    "Don’t count on it",
    "Yes, definitely",
    "Ask again later",
    "My reply is no",
    "Outlook not so good",
    "Signs point to yes"
]

print("Ask the Magic 8 Ball a question: ")
input()
print(random.choice(responses))
```

List-like Types: Strings and Tuples

Strings: Immutable sequences of characters.
- Similar to lists but cannot be changed.

Tuples: Immutable sequences.
- Created with parentheses `()`.

Examples

```
# Strings
text = "hello"
print(text[1])  # Output: e

# Tuples
data = (1, 2, 3)
print(data[0])  # Output: 1
```

References

Definition: Lists are assigned by reference, meaning changes to a copied list affect the original.
To avoid this: Use `copy()` or `list()` to create a shallow copy.

Example
 # Reference Example
original = [1, 2, 3]
copy = original
copy[0] = 99
print(original) # Output: [99, 2, 3]

 # Avoid with copy()
copy = original.copy()
copy[0] = 42
print(original) # Output: [99, 2, 3]

The Dictionary Data Type

Definition: A collection of key-value pairs enclosed in curly braces `{}`.
Characteristics
- Keys must be unique and immutable.
- Values can be of any type.

Example
student = {"name": "Alice", "age": 20, "grade": "A"}
print(student["name"]) **# Output:** Alice

Pretty Printing
- Use the `pprint` module to display complex data structures in a readable format.

Example
import pprint

data = {"name": "Alice", "subjects": ["Math", "Science"], "grades": {"Math": "A", "Science": "B"}}

```
pprint.pprint(data)
```

Using Data Structures to Model Real-World Things
Example Representing a phone book using dictionaries.

```
phone_book = {
   "John": "555-1234",
   "Alice": "555-5678",
   "Bob": "555-8765"
}
print(phone_book["Alice"])  # Output: 555-5678
```

Practice Problems

1. Basic List Operations

- Create a list of your favourite movies and:
  - Add a new movie.
  - Remove a movie by name.
  - Find the index of a specific movie.

2. Magic 8 Ball Extension

- Modify the Magic 8 Ball program to allow the user to ask multiple questions.

3. Dictionary Manipulation

- Create a dictionary representing a library with book titles as keys and their availability (`True/False`) as values.
- Implement functionality to:
  - Check if a book is available.
  - Borrow a book (set its value to `False`).
  - Return a book (set its value to `True`).

4. Real-World Modeling

- Use a dictionary to represent a menu for a restaurant. Include:
  - Item names as keys.
  - Prices as values.
  - Add functionality to calculate the total cost for a given list of items.