Lists

1. Creating Lists

Lists are ordered, mutable collections that can contain mixed data types.

```
# Creating lists
numbers = [1, 2, 3, 4, 5]
mixed = [1, "hello", 3.14, True]
print(f"Numbers: {numbers}")
print(f"Mixed types: {mixed}")

# Empty list
empty = []
print(f"Empty list: {empty}")
```

```
Numbers: [1, 2, 3, 4, 5]
Mixed types: [1, 'hello', 3.14, True]
Empty list: []
```

2. Indexing and Slicing

Access list elements using zero-based indexing. Negative indices count from the end.

```
numbers = [1, 2, 3, 4, 5]

# Indexing (zero-based)
print(f"First element: numbers[0] = {numbers[0]}")
print(f"Last element: numbers[-1] = {numbers[-1]}")
print(f"Second to last: numbers[-2] = {numbers[-2]}")

# Slicing [start:end]
print(f"Slice [1:4]: numbers[1:4] = {numbers[1:4]}")
print(f"First three: numbers[:3] = {numbers[:3]}")
print(f"From index 2: numbers[2:] = {numbers[2:]}")
```

```
First element: numbers[0] = 1
Last element: numbers[-1] = 5
Second to last: numbers[-2] = 4
Slice [1:4]: numbers[1:4] = [2, 3, 4]
First three: numbers[:3] = [1, 2, 3]
From index 2: numbers[2:] = [3, 4, 5]
```

3. List Operations

Lists support various operations for adding, removing, and modifying elements.

```
numbers = [1, 2, 3, 4, 5]

# Append - add to end
numbers.append(6)
print(f"After append(6): {numbers}")

# Insert - add at specific position
numbers.insert(0, 0)
print(f"After insert(0, 0): {numbers}")

# Remove - remove first occurrence
numbers.remove(3)
print(f"After remove(3): {numbers}")

# Pop - remove and return last element
popped = numbers.pop()
print(f"Popped: {popped}, List: {numbers}")
```

```
After append(6): [1, 2, 3, 4, 5, 6]
After insert(0, 0): [0, 1, 2, 3, 4, 5, 6]
After remove(3): [0, 1, 2, 4, 5, 6]
Popped: 6, List: [0, 1, 2, 4, 5]
```

4. List Methods

Python provides many built-in methods for working with lists.

```
data = [3, 1, 4, 1, 5, 9, 2]

# Common methods
print(f"Length: len(data) = {len(data)}")
print(f"Sum: sum(data) = {sum(data)}")
print(f"Max: max(data) = {max(data)}")
print(f"Min: min(data) = {min(data)}")
print(f"Count 1s: data.count(1) = {data.count(1)}")

# Sorting
data.sort()
print(f"After sort(): {data}")

# Reverse
data.reverse()
print(f"After reverse(): {data}")
```

```
Length: len(data) = 7
Sum: sum(data) = 25
Max: max(data) = 9
Min: min(data) = 1
Count 1s: data.count(1) = 2
After sort(): [1, 1, 2, 3, 4, 5, 9]
After reverse(): [9, 5, 4, 3, 2, 1, 1]
```

5. List Comprehensions

List comprehensions provide a concise way to create lists.

```
# Create list of squares
squares = [x**2 for x in range(5)]
print(f"Squares: {squares}")

# With condition
```

```
evens = [x for x in range(10) if x % 2 == 0]
print(f"Evens: {evens}")
```

```
Squares: [0, 1, 4, 9, 16]
Evens: [0, 2, 4, 6, 8]
```

Key Takeaways

- Lists are mutable and ordered
- Zero-based indexing, negative indices from end
- Common operations: append(), insert(), remove(), pop()
- List comprehensions for concise list creation