

Day 12



Introduction to Lists in Python:

What is a List?

A list is a collection of multiple values stored in one variable.

Lists can store:

- Numbers
- Strings
- Mixed data types
- Even other lists

Creating a List:

```
numbers = [1, 2, 3, 4]
names = ["Aditya", "Python", "Code"]
mixed = [1, "Hello", 3.5, True]
```

Important Points to Remember:

- Lists use square brackets []
- Indexing starts from 0
- Lists are mutable
- Can store mixed data types
- Can be nested

Example: creating a basic list

A screenshot of a Python code editor. The code in the editor is:

```
1 l = [1,3,2]
2 print(l)
```

The 'TERMINAL' tab is selected at the bottom. In the terminal, the output is:

```
PS E:\Python\Day11-20\Day12> python main.py
[1, 3, 2]
```

Example: printing the type of list

```
1 l = [1,3,2]
2 print(type(l))

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
● PS E:\Python\Day11-20\Day12> python main.py
<class 'list'>
```

Example: printing using the index.

```
4 l = [1, 3, 2]
5 print(l[0])
6 print(l[1])

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
● PS E:\Python\Day11-20\Day12> python main.py
1
3
```

Example: lists can be edited

```
4 l = [1, 3, 2]
5 l[1] = 5
6 print(l)

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
● PS E:\Python\Day11-20\Day12> python main.py
[1, 5, 2]
```

Example: other data types can also be stored in the lists

```
8     l = [1, 3, 2, "Aditya", 'Utsav', True, 5.6]
9     print(l)

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

● PS E:\Python\Day11-20\Day12> python main.py
[1, 3, 2, 'Aditya', 'Utsav', True, 5.6]
```

Example: negative indexing in lists

```
11    l = [1, 3, 2]
12    print(l[-1]) # This will print the last element of the list which is 2
13    print(l[-2]) # This will print the second last element of the list which is 3

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

● PS E:\Python\Day11-20\Day12> python main.py
2
3
```

Example: Index out of range error

```
14
15    l = [1, 3, 2, 5, 4]
16    l[5] = 6 # This will raise an IndexError because index 5 is out of range
17    print(l)

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

● PS E:\Python\Day11-20\Day12> python main.py
Traceback (most recent call last):
  File "E:\Python\Day11-20\Day12\main.py", line 16, in <module>
    l[5] = 6 # This will raise an IndexError because index 5 is out of range
           ^
IndexError: list assignment index out of range
```

Example: checking if an element is present in the list or not.

```
19     l = [1, 3, 2, 5, 4]
20     if 3 in l:
21         print("3 is present in the list")
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

● PS E:\Python\Day11-20\Day12> **python main.py**
3 is present in the list

Example: slicing in the list

```
23     numbers = [10, 20, 30, 40, 50]
24     print(numbers[1:4])      # [20, 30, 40]
25     print(numbers[:3])      # [10, 20, 30]
26     print(numbers[::-2])    # [10, 30, 50]
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

● PS E:\Python\Day11-20\Day12> **python main.py**
[20, 30, 40]
[10, 20, 30]
[10, 30, 50]

Example: some common list operations

```
29     nums = [1, 2, 3]
30     print(len(nums))      # length
31     print(3 in nums)      # membership
32     print(nums + [4,5])    # concatenation
33     print(nums * 2)        # repetition
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

● PS E:\Python\Day11-20\Day12> **python main.py**
3
True
[1, 2, 3, 4, 5]
[1, 2, 3, 1, 2, 3]

List Comprehension: List comprehension is a short and easy way to create a list using a loop in one line.

Example:

The screenshot shows a code editor with the following Python code:

```
35     numbers = [1, 2, 3, 4, 5]
36     squares = [n * n for n in numbers]
37     print(squares)
```

Below the code, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is selected, displaying the command and its output:

```
PS E:\Python\Day11-20\Day12> python main.py
[1, 4, 9, 16, 25]
```

- `for n in numbers` → loop through the list
- `n * n` → square each number
- `[]` → store results in a new list

Basically,

- List comprehension creates lists in one line
- Syntax: [expression for item in iterable]
- Can include if condition
- Cleaner and faster than normal loops

Summary:

- List stores multiple values in one variable
- Created using []
- Supports indexing and slicing
- Values can be changed
- Used to store and manage collections

--The End--