

Day 13



List Methods in Python:

What are List Methods?

List methods are built-in functions that work only on lists to:

- add items
- remove items
- change order
- get information

They are used with dot (.) notation.

Syntax:

list_name.method()

Important Notes:

- List methods modify the original list
- Most methods return None
- Lists are mutable

Example: append()

```
1 l = [1,2,6]
2 print(l) # Output: [1, 2, 6]
3 l.append(3)
4 print(l) # Output: [1, 2, 6, 3]
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

● PS E:\Python\Day11-20\Day13> **python main.py**
[1, 2, 6]
[1, 2, 6, 3]

Example: sort()

```
6     l = [44,5,4,388]
7     l.sort()
8     print(l)

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

● PS E:\Python\Day11-20\Day13> python main.py
[4, 5, 44, 388]
```

Example: sort(reverse=true)

```
6     l = [44,5,4,388]
7     l.sort(reverse=True)
8     print(l)

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

● PS E:\Python\Day11-20\Day13> python main.py
[388, 44, 5, 4]
```

Example: reverse()

```
7
8     l = [44,5,4,388]
9     l.reverse()
10    print(l)

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

● PS E:\Python\Day11-20\Day13> python main.py
[388, 4, 5, 44]
```

Example: index()

```
10     l = [44,5,4,388]
11     print(l.index(4))
12     print(l.index(388))

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
```

● PS E:\Python\Day11-20\Day13> python main.py
2
3

Example: count()

```
13
14     l = [3,4,5,6,7,7,8,8,8,8,8]
15     print(l.count(8)) # Output: 5

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
```

● PS E:\Python\Day11-20\Day13> python main.py
5

Example: keep in mind that if we store one list in another and then modify the later one then the original one get affected.

```
16
17     l = [1,2,3]
18     m = l
19     m[0] = 0
20     print(l) # Output: [0, 2, 3]
21     print(m) # Output: [0, 2, 3]

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
```

● PS E:\Python\Day11-20\Day13> python main.py
[0, 2, 3]
[0, 2, 3]

Example: alternative for above is copy(). It is much safer and good way.

```
23     l = [3,4,5,6]
24     m = l.copy()
25     m[0] = 0
26     print(l)  # Output: [3, 4, 5, 6]
27     print(m)  # Output: [0, 4, 5, 6]
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS E:\Python\Day11-20\Day13> python main.py
[3, 4, 5, 6]
[0, 4, 5, 6]

Example: insert(index,value)

```
29     l = [4,5,6]
30     l.insert(1, 10)
31     print(l)  # Output: [4, 10, 5, 6]
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS E:\Python\Day11-20\Day13> python main.py
[4, 10, 5, 6]

Example: extend()

```
33     l = [1,2,3,4,5]
34     m = [6,7,8]
35     l.extend(m)
36     print(l)  # Output: [1, 2, 3, 4, 5, 6, 7, 8]
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS E:\Python\Day11-20\Day13> python main.py
[1, 2, 3, 4, 5, 6, 7, 8]

Example: another way to concatenate the list

The screenshot shows a code editor interface with a dark theme. On the left, there is a code editor window containing the following Python code:

```
38     l = [1,2,3,4,5]
39     m = [6,7,8]
40     k = l+m
41     print(k)
```

Below the code editor, there is a navigation bar with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORT. The TERMINAL tab is currently selected, indicated by an underline. To the right of the tabs, there is some very faint text.

In the terminal area, there is a single line of text starting with a blue circular icon followed by "PS E:\Python\Day11-20\Day13> python main.py". Below this command, the output of the program is shown in brackets: "[1, 2, 3, 4, 5, 6, 7, 8]".

Summary:

- List methods help manage list data
- append, insert, extend → add items
- remove, pop, clear → remove items
- sort, reverse → arrange items
- index, count, copy → get info

--The End--