

Lists & Strings

Lists and Strings

Both are **collections of data**, but behave a bit differently.

Let's go step-by-step and make it crystal clear.

What Are Lists?

A **list** is like a **box with multiple compartments** — you can store *many values* in one variable.

Example:

```
fruits = ["apple", "banana", "cherry"]
```

Here,

- `fruits` is the list name.
- It holds 3 values (called **elements**).
- Each element has a **position** (called **index**).

Element	Index
apple	0
banana	1
cherry	2

Accessing Elements

You can access elements using **index numbers** (starting from 0).

```
print(fruits[0]) # apple
print(fruits[1]) # banana
```

Negative indexing

You can also count from the **end** using negative numbers.

```
print(fruits[-1]) # cherry
print(fruits[-2]) # banana
```



Changing List Values

Lists are **mutable**, meaning you can *change* them.

```
fruits[1] = "mango"
print(fruits)
```

Output:

```
['apple', 'mango', 'cherry']
```

+ Adding and Removing Items

Add elements:

```
fruits.append("orange")    # adds to end
fruits.insert(1, "grapes") # adds at position 1
```

Remove elements:

```
fruits.remove("apple") # removes by value  
fruits.pop()          # removes last item
```

Looping through a List

You can loop through all elements using `for`.

```
for fruit in fruits:  
    print(fruit)
```

Output:

```
apple  
mango  
cherry
```

List Length

To know how many items are inside:

```
print(len(fruits))
```

List Slicing

You can extract a part of the list using `[start:end]`.

```
numbers = [10, 20, 30, 40, 50]  
print(numbers[1:4]) # [20, 30, 40]
```

```
print(numbers[:3]) # [10, 20, 30]
print(numbers[2:]) # [30, 40, 50]
```

Some Useful List Functions

Function	Description	Example	Output
<code>append()</code>	Add to end	<code>fruits.append("kiwi")</code>	adds kiwi
<code>insert()</code>	Add at position	<code>fruits.insert(1, "mango")</code>	adds at index 1
<code>remove()</code>	Remove by value	<code>fruits.remove("apple")</code>	removes apple
<code>pop()</code>	Remove last	<code>fruits.pop()</code>	removes last
<code>len()</code>	Count items	<code>len(fruits)</code>	number of items
<code>sort()</code>	Sort items	<code>fruits.sort()</code>	alphabetical order
<code>reverse()</code>	Reverse order	<code>fruits.reverse()</code>	reversed list

Mental Model of Lists

Think of a **list** as:

"A container that can hold many different things, in order, that you can modify anytime."

STRINGS

Strings are **sequences of characters** —

like `"Hi"`, `"Hello world"`, `"1234"`, etc.

Even though they *look like text*, you can think of them as **lists of characters**.

Creating Strings

```
name = "Alex"  
quote = 'Be your own hero'
```

You can use either single `' '` or double `" "` quotes.

Accessing Characters

Just like lists, strings are **indexed**.

```
word = "Python"  
print(word[0]) # P  
print(word[1]) # y  
print(word[-1]) # n
```

String Slicing

You can slice strings the same way as lists.

```
print(word[0:3]) # Pyt  
print(word[1:]) # ython  
print(word[:4]) # Pyth
```

Strings Are Immutable

Unlike lists, **you can't change a string directly**.

```
word = "Hello"  
word[0] = "J" # ❌ Error
```

If you want to modify it, you must create a new one:

```
new_word = "J" + word[1:]  
print(new_word) # Jello
```

Useful String Functions

Function	Description	Example	Output
<code>len()</code>	Count characters	<code>len("Hello")</code>	5
<code>.upper()</code>	Convert to uppercase	<code>"hello".upper()</code>	HELLO
<code>.lower()</code>	Convert to lowercase	<code>"HELLO".lower()</code>	hello
<code>.replace()</code>	Replace part	<code>"Hello".replace("H", "J")</code>	Jello
<code>.find()</code>	Find index of word	<code>"Hello".find("e")</code>	1
<code>.split()</code>	Split string by space	<code>"Hi there".split()</code>	['Hi', 'there']
<code>.join()</code>	Join list into string	<code>' '.join(['Hi', 'there'])</code>	Hi there

Looping Through a String

```
for letter in "Python":  
    print(letter)
```

Output:

```
P  
y  
t  
h  
o  
n
```


Lists vs Strings (Side-by-Side)

Feature	List	String
Holds	Many items (any type)	Only text (characters)
Mutable	✓ Yes	✗ No
Indexed	✓ Yes	✓ Yes
Loopable	✓ Yes	✓ Yes
Example	<code>[1,2,3]</code>	<code>"123"</code>

Practice Exercises

1. Create a list of 5 subjects and print the 2nd and last subject.
2. Add one subject, remove another, and print the new list.
3. Take a string `"python"` and print it backward using slicing.
4. Convert `"hello world"` to uppercase and replace `"world"` with `"friend"`.

Mental Picture

 **String** = "a line of letters joined together."

 **List** = "a box that holds multiple things you can change."