Strings

Java Reference

String methods

A `String` 字符串 is text. Useful methods: `.length()`, `.charAt(i)`, `.substring(a, b)`, `.indexOf(x)`, `.toUpperCase()`, `.equals(...)`. Strings are immutable 不可变 — each method returns a **new** String.

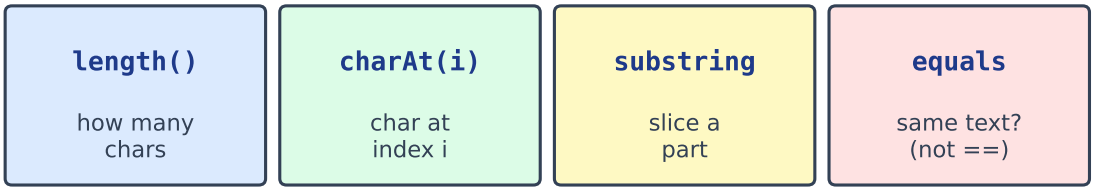
```
public class Main {
    public static void main(String[] args) {
        String s = "Python";
        System.out.println(s.length());          // 6
        System.out.println(s.charAt(0));         // P
        System.out.println(s.substring(0, 3));   // Pyt
        System.out.println(s.toUpperCase());     // PYTHON
    }
}
```

The methods you will use every day:

Method	Meaning	Example → result
`.length()`	how many characters	`"Hi".length()` → `2`
`.charAt(i)`	one character	`"Hi".charAt(0)` → `H`
`.substring(a, b)`	part, stops before `b`	`"Python".substring(0, 3)` → `Pyt`
`.substring(a)`	from `a` to the end	`"Python".substring(3)` → `hon`
`.indexOf(x)`	first position, `-1` if absent	`"banana".indexOf("na")` → `2`
`.equals(s)`	same text?	`"hi".equals("hi")` → `true`
`.compareTo(s)`	order: negative / 0 / positive	`"apple".compareTo("banana")` → negative

`compareTo` puts Strings in dictionary order — the AP exam uses it for sorting questions:

```
public class Main {
    public static void main(String[] args) {
        String a = "apple", b = "banana";
        System.out.println(a.compareTo(b) < 0);    // true (apple comes
↪   first)
        System.out.println(a.compareTo("apple"));  // 0 (equal)
    }
}
```



Exam tip: Strings are objects — use equals for text, not ==; indices start at 0

Key String methods: length, charAt, substring, equals

Building & traversing strings

Join strings with + (concatenation 拼接). Visit each character with a loop and `.charAt(i)`.

```
public class Main {
    public static void main(String[] args) {
        String word = "banana";
        int count = 0;
        for (int i = 0; i < word.length(); i++) {
            if (word.charAt(i) == 'a') count++;
        }
        System.out.println(count);   // 3
    }
}
```

When you build a long String in a loop, `StringBuilder` is much faster: append the pieces, then call `.toString()` once.

```
public class Main {
    public static void main(String[] args) {
        StringBuilder sb = new StringBuilder();
        for (int i = 1; i <= 5; i++) {
            sb.append(i).append(" ");
        }
        System.out.println(sb.toString().trim());   // 1 2 3 4 5
    }
}
```

Common mistakes

- Strings are immutable: `s.toUpperCase()` returns a new String, so store the result.
- Read a character with `s.charAt(i)`; the length is `s.length()` (a method, with `()`).
- Building a String with `+=` in a big loop is slow; use a `StringBuilder`.