Python Strings: From Scratch to Mastery

1. What is a String?

A **string** in Python is a **sequence of characters** enclosed in **single quotes (' ')**, **double quotes (" ")**, or **triple quotes (''' or """)** for multi-line text.

```
str1 = 'Hello'
str2 = "World"
str3 = '''Triple
quoted string'''
```

2. String Basics

Concept	Syntax/Example	Notes
Declare	`s = "hello"`	Strings are immutable
Access	`s[0]`	Indexing starts at 0
Slice	`s[1:4]`	Slicing is `[start:end]`
Length	`len(s)`	Counts characters
Concatenate	`"hi" + "bye"`	Joins two strings
Repeat	`"ha" * 3`	Returns `"hahaha"`
Membership	`'h' in 'hello'`	Returns `True` or `False`

3. Slicing & Indexing

```
s = "Python"
s[0]     # 'P'
s[-1]    # 'n'
s[0:2]   # 'Py'
s[:3]    # 'Pyt'
s[2:]    # 'thon'
s[::-1]  # 'nohtyP' (reverse string)
```

4. String Methods

Here are **most-used string methods**, with examples and descriptions:

🔠 Case Conversion

Method	Description	Example
`.lower()`	All lowercase	`'Hello'.lower()` → `'hello'`
`.upper()`	All uppercase	`'hello'.upper()` → `'HELLO'`
`.capitalize()`	First letter capital	`'python'.capitalize()` → `'Python'`
`.title()`	Capitalizes every word	`'python is fun'.title()` → `'Python Is Fun'`
`.swapcase()`	Swap uppercase ↔ lowercase	`'PyTHon'.swapcase()` → `'pYthON'`

📏 Check String Characteristics

Method	Checks for...	Example
`.isalpha()`	Letters only	`'abc'.isalpha()` → `True`
`.isdigit()`	Digits only	`'123'.isdigit()` → `True`
`.isalnum()`	Letters or digits	`'abc123'.isalnum()` → `True`
`.isspace()`	Only whitespace	`' '.isspace()` → `True`
`.istitle()`	Titlecase	`'Hello World'.istitle()` → `True`
`.islower()`	All lowercase	`'hello'.islower()` → `True`
`.isupper()`	All uppercase	`'HELLO'.isupper()` → `True`

🔍 Searching & Replacing

Method	Description	Example
`.find(sub)`	First index of `sub`, else `-1`	`'hello'.find('e')` → `1`
`.rfind(sub)`	Last index of `sub`	`'hello hello'.rfind('e')` → `10`
`.index(sub)`	Like find(), but raises error if not found	`'hi'.index('i')` → `1`
`.replace(a, b)`	Replace `a` with `b`	`'hi'.replace('i', 'ello')` → `'hello'`

🪄 Formatting

Method	Description	Example
`.format()`	Insert values	`"My name is {}".format('Bob')`
`f"{}"`	f-string (Python 3.6+)	`f"My name is {name}"`
`.zfill(width)`	Pads with zeros	`'5'.zfill(3)` → `'005'`
`.center(n)`	Center string	`'hi'.center(10)` → `' hi '`

🧹 Strip & Split

Method	Description	Example
`.strip()`	Removes whitespace from both ends	`' hello '.strip()` → `'hello'`
`.lstrip()`	Removes leading space	`' hello'.lstrip()` → `'hello'`
`.rstrip()`	Removes trailing space	`'hello '.rstrip()` → `'hello'`
`.split()`	Splits on whitespace (or a delimiter)	`'a,b,c'.split(',')` → `['a','b','c']`
`.rsplit()`	Splits from right	`'a b c'.rsplit(' ', 1)` → `['a b', 'c']`
`.join(iterable)`	Joins list with string as separator	`','.join(['a','b'])` → `'a,b'`

🔤 Check Start/End

Method	Description	Example
`.startswith('sub')`	Checks prefix	`'hello'.startswith('he')` → `True`
`.endswith('sub')`	Checks suffix	`'hello'.endswith('lo')` → `True`

🤯 5. Escape Characters 🎭

Escape characters are used to insert illegal characters:

Escape	Meaning
`\n`	Newline
`\t`	Tab
`\\`	Backslash
`\'`	Single quote
`\"`	Double quote
`\r`	Carriage Return
`\b`	Backspace

🧩 6. String Formatting 🎨

```
name = "Darshan"
age = 21
```

```
# Old style
print("My name is %s and age is %d" % (name, age))

# format()
print("My name is {} and I am {} years old".format(name, age))

# f-string (recommended)
print(f"My name is {name} and I am {age} years old")
```

🛠️🌐 7. Advanced: String Encoding & Unicode

```
# Encoding
s = "हेलो"
encoded = s.encode('utf-8')     # to bytes
decoded = encoded.decode('utf-8')  # back to string
```

🔢 8. String to Other Types

```
int("123")       # 123
float("3.14")    # 3.14
str(123)         # "123"
list("abc")      # ['a', 'b', 'c']
```

🧠🎯 9. Common Interview Questions

Question	Example Code
Reverse a string	`s[::-1]`
Count vowels in a string	`sum(1 for i in s if i in "aeiouAEIOU")`
Palindrome check	`s == s[::-1]`
Anagram check	`sorted(s1) == sorted(s2)`
Remove duplicates	`''.join(set(s))` or via OrderedDict

🧪💡 10. Practice Challenge Ideas

- 🔁 Count frequency of each character in a string
- 🔍 Find longest word in a sentence
- 🔄 Check if two strings are rotations
- ✂️ Remove all special characters
- 🔢 Convert words to numbers (e.g., `"one"` → `1`)

Bonus: Useful Built-ins with Strings

```
# chr() and ord()
print(chr(97))   # 'a'
print(ord('a'))  # 97

# eval()
print(eval("2 + 3"))  # 5
```

Summary Table

Task	Code Example
Reverse string	`s[::-1]`
Title case each word	`s.title()`
Remove whitespace	`s.strip()`
Convert to list of chars	`list(s)`
Replace substring	`s.replace("a", "b")`
Check if digit	`s.isdigit()`
Pad with zeros	`s.zfill(5)`
Split by delimiter	`s.split(',')`

Final Tips

- Strings are **immutable**
- Use **f-strings** for formatting
- Use **join/split/strip** frequently in real-world tasks
- Learn **regex** (`re` module) for text processing (advanced)

Part 1: Regular Expressions (Regex) in Python

Regex = *Pattern matching magic!*

What is Regex?

> Regular Expressions (Regex) are **patterns** used to **match strings** or **substrings**.

Python has a built-in module:

```
import re
```

Common Regex Patterns

Pattern	Meaning	Example Match
`.`	Any character except newline	`a`, `b`, `1`, `!`
`^`	Start of string	`^Hello` → matches `"Hello World"`
`$`	End of string	`end$`
`*`	0 or more occurrences	`a*` → `""`, `"a"`, `"aaa"`
`+`	1 or more	`a+` → `"a"`, `"aaa"`
`?`	0 or 1	`a?` → `""`, `"a"`
`[]`	Any one of the characters	`[aeiou]`
`[^]`	Not in set	`[^aeiou]`
`{n}`	Exactly n times	`a{3}` → `"aaa"`
`{n,}`	n or more	`a{2,}` → `"aa"`, `"aaa"`
`{n,m}`	Between n and m times	`a{1,3}` → `"a"`, `"aa"`
` ` `	` ` `	OR `cat dog`
`\d`	Digit (0–9)	`"123"`
`\D`	Non-digit	`"abc"`
`\w`	Word character (a-zA-Z0-9_)	`"hello_123"`
`\W`	Non-word character	`"@! "`
`\s`	Whitespace	`" "`
`\S`	Non-whitespace	`"abc"`

Common Functions in `re` module

```
import re

# ☑ Search pattern in string
re.search("pattern", "text")

# ☑ Match only from start
re.match("Hello", "Hello World")

# ☑ Find all matches
```

```
re.findall("\d+", "a1 b22 c333")  # ['1', '22', '333']

# ☑ Substitute
re.sub("\s+", "-", "hello   world")  # 'hello-world'

# ☑ Compile for reuse
pattern = re.compile("[a-z]+")
pattern.findall("abc XYZ 123")  # ['abc']
```

🩹 Example Regex Problems

Task	Regex	Example
Validate email	`^\w+@\w+\.\w{2,3}$`	`abc@mail.com` ☑
Extract digits	`\d+`	`'abc123'` → `['123']`
Find hashtags	`#\w+`	`'hi #world'` → `['#world']`
Find all words	`\b\w+\b`	`'I am GPT'` → `['I','am','GPT']`

🎁 Pro Tip:

Test regex online at: regex101.com 🩹

🧠 Part 2: String Interview Problems in Python

🚀 Easy Level

1. **Reverse a string**

```
s[::-1]
```

2. **Check Palindrome**

```
def is_palindrome(s): return s == s[::-1]
```

3. **Count Vowels**

```
sum(1 for ch in s.lower() if ch in "aeiou")
```

4. **Remove duplicates**

```
''.join(sorted(set(s), key=s.index))
```

Medium Level

5. **First non-repeating character**

```
from collections import Counter
def first_unique(s):
    count = Counter(s)
    for ch in s:
        if count[ch] == 1:
            return ch
    return None
```

6. **Anagram check**

```
sorted(s1) == sorted(s2)
```

7. **Most frequent character**

```
max(set(s), key=s.count)
```

8. **Longest word in sentence**

```
max(sentence.split(), key=len)
```

Harder Level

9. **Compress string (run-length encoding)** `"aaabbccc"` → `"a3b2c3"`

```
def compress(s):
    res = ""
    count = 1
    for i in range(1, len(s)):
        if s[i] == s[i-1]:
            count += 1
        else:
```

```
            res += s[i-1] + str(count)
            count = 1
    res += s[-1] + str(count)
    return res
```

10. **Check if rotation** `"abc"` and `"cab"` → ☑

```
def is_rotation(a, b):
    return len(a) == len(b) and b in a + a
```

11. **Group anagrams**

```
from collections import defaultdict
def group_anagrams(words):
    d = defaultdict(list)
    for w in words:
        d[''.join(sorted(w))].append(w)
    return list(d.values())
```

📦 Bonus Challenge

12. **Valid Parentheses**

```
def is_valid(s):
    stack = []
    mapping = {')':'(', ']':'[', '}':'{'}
    for ch in s:
        if ch in mapping.values():
            stack.append(ch)
        elif ch in mapping:
            if not stack or mapping[ch] != stack.pop():
                return False
    return not stack
```

☑ Summary Table

Problem	Concept Used
Reverse / Palindrome	Slicing
Anagram	`sorted()`
Compress	Run-length encoding

Problem	Concept Used
Parentheses Matching	Stack
Unique / Frequent char	`collections.Counter`
Rotation	String + Concatenation