📘 Python Regular Expressions (Regex)

1. Introduction

Regular expressions (regex) are sequences of characters that define search patterns, mainly for string searching, validation, and manipulation. Python provides the built-in `re` module to work with regex.

---

2. Importing the `re` Module

```python
import re
```

---

3. Common Regex Functions in Python

Function	Description	Example
`re.match()`	Matches pattern at the beginning of a string	`re.match(r"\d+", "123abc")`
`re.search()`	Finds the first occurrence of the pattern anywhere	`re.search(r"\d+", "abc123")`
`re.findall()`	Returns all matches as a list	`re.findall(r"\d+", "abc123def456")`
`re.finditer()`	Returns an iterator with match objects	`re.finditer(r"\d+", "a1b2")`
`re.sub()`	Replaces matches with a string	`re.sub(r"\d+", "#", "abc123")`
`re.split()`	Splits string by pattern	`re.split(r"\s+", "hello world python")`

---

4. Common Regex Metacharacters

Symbol	Meaning	Example
`.`	Any character except newline	`re.findall(r"a.c", "abc, acc, azc")`
`^`	Start of string	`re.match(r"^Hello", "Hello World")`
`$`	End of string	`re.search(r"world$", "hello world")`

Symbol	Meaning	Example
`\d`	Digit	`"123"` matches
`\D`	Non-digit	`"abc"` matches
`\w`	Word character (alphanumeric + _)	`"hello123"`
`\W`	Non-word	`"!@#"`
`\s`	Whitespace	`" \t\n"`
`\S`	Non-whitespace	`"abc123"`
`*`	0 or more repetitions	`"ab*"` matches `"a", "ab", "abb"`
`+`	1 or more repetitions	`"ab+"` matches `"ab", "abb"`
`?`	0 or 1 occurrence	`"ab?"` matches `"a", "ab"`
`{n}`	Exactly n repetitions	`"a{3}"` matches `"aaa"`
`{n,}`	n or more	`"a{2,}"` matches `"aa", "aaa"`
`{n,m}`	Between n and m	`"a{2,4}"` matches `"aa", "aaa", "aaaa"`
`[]`	Character class	`[abc]` matches `"a" or "b" or "c"`
‘	‘	OR operator ‘“cat dog”‘
`()`	Grouping	`"(abc)+"`

---

5. Compiling Patterns

```
pattern = re.compile(r"\d+")
print(pattern.findall("Order 123, item 456"))
```

---

6. Best Practices

- Always use raw strings (`r""`) for regex.
- Pre-compile patterns if reused multiple times.
- Test regex online (e.g., regex101.com).
- Keep regex readable (avoid over-complicating).

📝 25 Exercises on Python Regular Expressions

Beginner (Basics)

1. Write a regex to check if a string starts with “Hello”.

2. Extract all numbers from `"My phone number is 12345 and zip is 67890"`.
3. Validate if a string contains only digits.
4. Find all words in a sentence.
5. Replace all digits with `*` in `"abc123def456"`.

Intermediate (Validation & Patterns)

6. Validate an email address.
7. Extract domain names from a list of URLs.
8. Match all words that start with a capital letter.
9. Split a string by multiple delimiters (`","` `";"` `":"`).
10. Check if a string is a valid US phone number (`123-456-7890`).

Advanced (Practical)

11. Extract hashtags from a tweet.
12. Validate strong passwords (at least 8 chars, one digit, one uppercase, one symbol).
13. Parse dates in format `dd/mm/yyyy` or `dd-mm-yyyy`.
14. Find repeated words in a text.
15. Extract HTML tags from a string.

Expert (Complex Cases)

16. Match IPv4 addresses.
17. Validate a credit card number format (`xxxx-xxxx-xxxx-xxxx`).
18. Find all time values (`HH:MM` format).
19. Extract all programming keywords from a code snippet.
20. Build a regex to validate a hex color code (`#FFF` or `#FFFFFF`).

Challenge (Mix of Concepts)

21. Extract mentions (`@username`) from a text.
22. Replace all whitespace sequences with a single space.
23. Find words longer than 7 letters in a sentence.
24. Match floating-point numbers.
25. Write a regex to validate an identifier in Python (variable/function names).
