🎨Regular Expressions (Regex) in Python

🏐What is Regex?

✨Regular Expressions (**Regex**) are **patterns** used to **match, search, extract, and manipulate text**.

In Python, the `re` module is used:

```
import re
```

🏐Raw Strings (`r""`)

Regex patterns often use many backslashes ⚡. Use **raw strings** to avoid confusion:

```
pattern = r"\d+"   # 🦜 raw string (recommended)
```

Without raw string, you'd need `"\\d+"`.

🏐Core Functions in `re`

🦠Function	🐞Description	🏖️Example
`re.match()`	Match only at the **start** of string	`re.match(r"Hi", "Hi there")`
`re.search()`	Find **first match** anywhere	`re.search(r"\d+", "Age 25")`
`re.findall()`	Find **all matches** → list	`re.findall(r"\d+", "A1 B2")`
`re.finditer()`	Iterator of match objects	Loop for details
`re.sub()`	Replace text	`re.sub(r"cat", "dog", "cat runs")`
`re.split()`	Split by regex	`re.split(r"\W+", "apple,banana;grape")`

🏐 Common Regex Patterns

🦙 Pattern	🐞 Meaning	🏖️ Example
`.`	Any character (except newline)	`"a.c"` → matches `"abc"`, `"axc"`
`\d`	Digit (0–9)	`"a\d"` → matches `"a1"`
`\D`	Non-digit	`"a\D"` → matches `"ax"`
`\w`	Word char (a–z, A–Z, 0–9, _)	`"a\w"` → matches `"a1"`, `"ab"`
`\W`	Non-word char	`"a\W"` → matches `"a!"`
`\s`	Whitespace	`"a\s"` → matches `"a "`
`\S`	Non-whitespace	`"a\S"` → matches `"ax"`
`^`	Start of string	`"^Hi"`
`$`	End of string	`"bye$"`
`*`	0 or more	`"a*"` → `""`, `"a"`, `"aaa"`
`+`	1 or more	`"a+"` → `"a"`, `"aaa"`
`?`	0 or 1	`"a?"` → `""`, `"a"`
`{n}`	Exactly n times	`\d{3}` → `"123"`
`{n,}`	At least n	`\d{2,}` → `"12"`, `"1234"`
`{n,m}`	Between n and m	`\d{2,4}` → `"12"`, `"1234"`
`	`	OR	`(cat dog)` matches either cat or dog`
`()`	Group	`(abc)+`
`[]`	Character set	`[aeiou]` → matches vowels

🏐 Dash (`-`) in Regex

Inside `[ ]` → Range

```
print(re.findall(r"[a-z]", "abcXYZ123"))   # ['a', 'b', 'c']
print(re.findall(r"[A-Z]", "abcXYZ123"))   # ['X', 'Y', 'Z']
print(re.findall(r"[0-9]", "abcXYZ123"))   # ['1', '2', '3']
```

Outside `[ ]` → Literal Dash

```
print(re.findall(r"-", "123-456-789"))   # ['-', '-']
```

Escaped or at Edges → Literal Dash

```
print(re.findall(r"[-a-z]", "a-b-c"))   # ['a', '-', 'b', '-', 'c']
```

🦜 Quick Recap:

- `[a-z]` → lowercase letters
- `[0-9]` → digits
- `-` outside `[]` → just a dash

🏅 Practical Regex Examples

🏐 1. Find Numbers

```
text = "Order 66 in year 2023"
print(re.findall(r"\d+", text))   # ['66', '2023']
```

🌳 Pattern `\d+` means **one or more digits**. Here it matches numbers `66` and `2023`.

🏐 2. Validate Phone Number

```
phone = "123-456-7890"
print(bool(re.match(r"^\d{3}-\d{3}-\d{4}$", phone)))  # True
```

🌴Pattern `^\d{3}-\d{3}-\d{4}$` means:

- `^` → start of string
- `\d{3}` → exactly 3 digits
- `-` → literal dash
- repeat again `\d{3}-\d{4}`
- `$` → end of string 🦜Matches `123-456-7890` format.

🏐3. Extract Emails

```
text = "Emails: alice@mail.com, bob@gmail.com"
print(re.findall(r"[\w\.-]+@[\w\.-]+\.\w+", text))
```

🌴Pattern `[\w\.-]+@[\w\.-]+\.\w+` means:

- `[\w\.-]+` → one or more word chars, dots, or dashes (username)
- `@` → at symbol
- `[\w\.-]+` → domain name
- `\.\w+` → dot + word chars (e.g., `.com`, `.org`) 🦜Extracts full email addresses.

🏐4. Replace Text

```
print(re.sub(r"hate", "love", "I hate bugs"))  # I love bugs
```

🌴Pattern `hate` finds exact word `hate` and replaces it with `love`.

🏐5. Split by Non-Word Characters

```
print(re.split(r"\W+", "apple,banana;grape|melon"))
```

🌴Pattern `\W+` means **one or more non-word characters**. Splits text into clean words: `["apple", "banana", "grape", "melon"]`.

🏐6. Words Starting with Capital Letter

```
text = "London is in England"
print(re.findall(r"\b[A-Z][a-z]+\b", text))
```

🌴Pattern `\b[A-Z][a-z]+\b` means:

- `\b` → word boundary
- `[A-Z]` → first letter must be capital
- `[a-z]+` → followed by one or more lowercase letters
- `\b` → word boundary ends 🦜Finds words like `London`, `England`.

🏐7. Strong Password Rule

```
password = "StrongPass1"
pattern = r"^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z]).{8,}$"
print(bool(re.match(pattern, password)))  # True
```

🌴Pattern explanation:

- `^` → start of string
- `(?=.*[0-9])` → must contain at least one digit
- `(?=.*[a-z])` → must contain at least one lowercase
- `(?=.*[A-Z])` → must contain at least one uppercase
- `.{8,}` → at least 8 characters
- `$` → end of string 🦜Validates strong passwords.

Summary

Regex in Python is used for:

- 🦗Searching
- 🌻Validating
- 🦌 Extracting
- Replacing
- 🐓Splitting

⚡Master these to become a **Regex Ninja** in Python!