Regex Essential Concepts & Questions

Regex (Regular Expression) is a **special pattern-matching language** used to search, filter, and manipulate text.

- It uses Regex Metacharacters, Quantifiers, and etc to do so.

Metacharacters:
\d: Matches any decimal digit; this is equivalent to the class [0-9].
\D: Matches any non-digit character; this is equivalent to the class [^0-9].
\s: Matches any whitespace character; this is equivalent to the class [\t\n\r\f\v].
\S: Matches any non-whitespace character; this is equivalent to the class [^ \t\n\r\f\v].
\w: Matches any alphanumeric character; this is equivalent to the class [a-zA-Z0-9_].
\W: Matches any non-alphanumeric character; this is equivalent to the class [^a-zA-Z0-9_].

Quantifiers:
+: + indicates that the previous character must occur at least one or more times.
?: ? indicates that the preceding character is optional. It means the preceding character can occur zero or one time.
***:** Matches zero or more of the preceding character.
{n}: Matches exactly n occurrences of the preceding character.
{n,}: Matches n or more occurrences of the preceding character.
{n,m}: Matches between n and m occurrences of the preceding element

Official Documentation: https://docs.python.org/3/howto/regex.html#matching-characters

---

1. Extract all Email IDs from a text msg:

Python Code

```
import re

text = "Contact us at support@gmail.com or sales.team@company.org"
emails = re.findall(r"[\w\.-]+@[\w\.-]+\.\w+", text)
print(emails)
```

Explanation

- We look for: **username + @ + domain + extension**
- Like: **username (support) + @ + domain (gmail) + '.' + extension (com)**
- `\w` matches any **word character**.
- \w + \'.'
- \w + at least one such word

---

2. Extract Indian Phone Numbers

Python Code

```
import re

text = "Call me at +91-9876543210 or +91-98765-43210"
phones = re.findall(r"(\+91[- ]?)?[0-9]{10}", text)
print(phones)
```

Explanation

- \+, keep + as it is don't resolve it so escape (\) is used
- `+91` is optional.
- After that, we need **exactly 10 digits**.
- Also handles `+91-` or `+91 `.

---

3. Extract Dates (DD/MM/YYYY or DD-MM-YYYY)

Python Code

```
import re

text = "Dates: 12/05/2024, 15-06-2023, and 01/01/2025"
dates = re.findall(r"\b\d{2}[-/]\d{2}[-/]\d{4}\b", text)
print(dates)
```

Explanation

- `\d{2}` → 2 digits, day
- `[-/]` → either `-` or `/`
- `\d{2}` → 2 digits, month
- `\d{4}` → 4 digits, year
- Works for both `12/05/2024` and `12-05-2024`.

---

4. Extract Time (HH:MM or HH:MM:SS)

Python Code

```
import re

text = "Train leaves at 14:30 and arrives by 18:45:20"
times = re.findall(r"\b\d{2}:\d{2}(:\d{2})?\b", text)
print(times)
```

Explanation

- \bxyz\b: match ONLY the word “xyz” and nothing attached before or after it.
- `HH:MM` is required.
- `(:SS)?` → seconds are optional.
- Works for `14:30` and `18:45:20`.

---

5. Extract URLs

Python Code

```
import re

text = "Visit https://google.com or http://www.example.org/about"
urls = re.findall(r"https?://[^\s]+", text)
print(urls)
```

Explanation

- `http` or `https due to s is optional`
- `‘://’` fixed data
- `[^\s]+` → take characters until a space appears. (Should include at least one such character)
  - \s matches any whitespace character
- Captures complete URLs.

---

6. Extract ALL CAPS Words

Python Code

```
import re

text = "Python, SQL, HTML, and CSS are useful."
caps_words = re.findall(r"\b[A-Z]+\b", text)
print(caps_words)
```

Explanation

- `[A-Z]+` → letters must be uppercase.
- `\b` word boundaries ensure full words like SQL, HTML, CSS.
  - Ensures that we match **whole words only**, not part of a word.

---

7. Extract Words Starting With a Vowel

Python Code

```
import re

text = "Apple is awesome and elephant is big but mango is sweet"
vowel_words = re.findall(r"\b[aeiouAEIOU]\w*", text)
print(vowel_words)
```

Explanation

- \b Matches the start of a word.
- `[aeiouAEIOU]` → starts with vowel (case-insensitive)
    - Matches any vowel, lowercase or uppercase (a, e, i, o, u and A, E, I, O, U). So the word must start with a vowel.
- `\w*` → rest of the letters
    - \w matches letters, digits, or underscore.
    - * non or any number of such letters
- Finds: Apple, is, awesome, etc.

---

8. Extract All Numbers (Integer + Decimal)

Python Code

```
import re

text = "Prices are 45, 67.50, 100.25 and 120"
numbers = re.findall(r"\d+(\.\d+)?", text)
print(numbers)
```

Explanation

- `\d+` → integer part
- `(\.\d+)?` → decimal optional
- Detects: 45, 67.50, 100.25, 120.

---

9. Validate Strong Password

Python Code

```
import re

password = "Abcd@123"

pattern = r"^(?=.*[A-Z])(?=.*[a-z])(?=.*\d)(?=.*[@$!%*?&]).{8,}$"

if re.match(pattern, password):
    print("Strong password")
```

```
else:
    print("Weak password")
```

Explanation

- Must contain:
    - `(?=.*[A-Z])` → uppercase
    - `(?=.*[a-z])` → lowercase
    - `(?=.*\d)` → number
    - `(?=.*[@$!%*?&])` → special character
- Length must be ≥ 8.

---

10. Extract Hashtags

Python Code

```
import re

text = "Loving the trip! #travel #Goa2024 #sunset_view"
hashtags = re.findall(r"#[A-Za-z0-9_]+", text)
print(hashtags)
```

Explanation

- Starts with `#`
- Followed by letters, digits, or _
- Extracts full hashtags.
