PAGE:
DATE: / /

Mission Name : Advanced Regular Expression.

* Capture Groups :- we define a capture group by wrapping the part we want to capture in parenthesis.

Eg:-

```
pattern = r"(\w+SQL)"
sql_flavors = titles.str.extract(pattern, flags= re.I)
sql_flavours_freq = sql_flavors.value_counts()
print(sql_flavors_freq)
```

* Another example of capture groups:-

Eg:-
```
pattern = r"[Pp]ython ([\d\.]+)"
py_versions = titles.str.extract(pattern)
py_versions_freq = dict(py_version.value_counts())
```

The above code will extract the version followed by the word Python/python and make a frequency dictionary with it.

* Eg:- pattern = r" \b[Cc][^\.\+\w]\b"

* Lookarounds:-

Lookaround	Pattern	Explanation
Positive Lookahead	zzz (?=abc)	Matches only when zzz is followed by abc.
Negative Lookahead	zzz(?!abc)	Matches only when zzz is not followe by abc.
Positive lookbehind	(?<=abc) zzz	Match zzz only if it is preceded by ab
Negative ~~Positive~~ lookbehind	(?<!abc)zzz	Match zzz only if it is NOT precede by abc.

PAGE:
DATE: / /

Eg:- pattern = r"(?<!Series\s)\b[Cc]\b(?![\.\+])"

*** Back References:-**

If we want to repeat any capture group in a RegEx pattern, we can use back references.

we can number capture groups from left to right and just mention the number wherever we want to repeat it.

Eg:- (Hello)(Goodbye)\2\1

This will match HelloGoodbyeGoodbyeHello.

(\w)\1

With match 2 repeating characters, like "ee", "oo" etc.

Eg 2:- pattern = r"\b(\w+)\s\1\b"

*** re.sub()**

Function is similar to string.replace().

Syntax:- re.sub(pattern, repl, string, flags=0)

regex   string to replace   string

For pandas series:- Series.str.replace(pat, repl, flags=0)

```
Eg:- titles_clean = titles.str.replace(pat = r"e-?\s?mail",
                                       repl = "email",
                                       flags = re.I)
```

* Extracting domain names:- pattern = r".{4}s?://([\w\.\-]+)/?"

* Naming extracted columns :- r"(?P<date>.+) (?P<time>.+)"