Regex

`import re`

Meta characters that need to be escaped :

`. ^ $ + * ? { } [ ] \ | ( )`

like if we want to find out shivam.com..

`we would write re.compile(r'shivam\.com')`

`.` dot matches any character except a new line

`\d` matches a single digit

`\D` matches other than digits

`\w` matches word character (a-z, A-Z, 0-9, _)

`\W` not a word character

`\s` whitespace (space, tab, newline)

`\S` not whitespace (space, tab, newline)

`\b` matches word boundary(eg: `re.compile(r'\bHa')` will match both `Ha` and `Ha`Ha in `Ha` `Ha`Ha)

`\B` not above

`^` matches a text which is the begining of a string

`$` matches end of a string (eg: `re.compile(r'end$')`

`[. -]` only matches what's inside these brackets(this will match either a dash or a dot not both together)

`[1-5]` will only match range of digits bw 1 and 5

`[a-zA-Z]` matches upper case or upper case letters

`[^b]at` will match everything that doesn't start wit b

`Mr\.?` matches Mr with a dot or not a dot coz of `?`

`(Ms|Mr|Mrs)` will match Mr or Ms or Mrs

Quantifiers

`*` 0 or more number of whatever given things `Mr\.?\s[a-zA-z]\w*`

`+` 1 or more number of things

`?` 0 or 1 number of things

`{3}` exact number

`{1,5}` range of number

`print(r'\tTab')` , using r wont treat \t as tab

```
sentence = 'a string '
pattern = re.compile(r'abc')
matches = pattern.finditer(text_to_searc): pattern.match(text_to_searc)
    for match in matches:
        print(match)
```

we can also use `pattern.search(text_to_searc)` it would return only the exact match None otherwise

`match.group(0)` matches whole text under the group or `()`

`match.group(1)` matches 1st group under the group meaning this notation `()`

`match.group(2)` similarly matches 2nd group under the group meaning this notation `()`

if we have expression like this `(somethin)\.(some other thing)?(something)(something)`

```
subbed = pattern.sub(r' \2\3 ', urls)
```