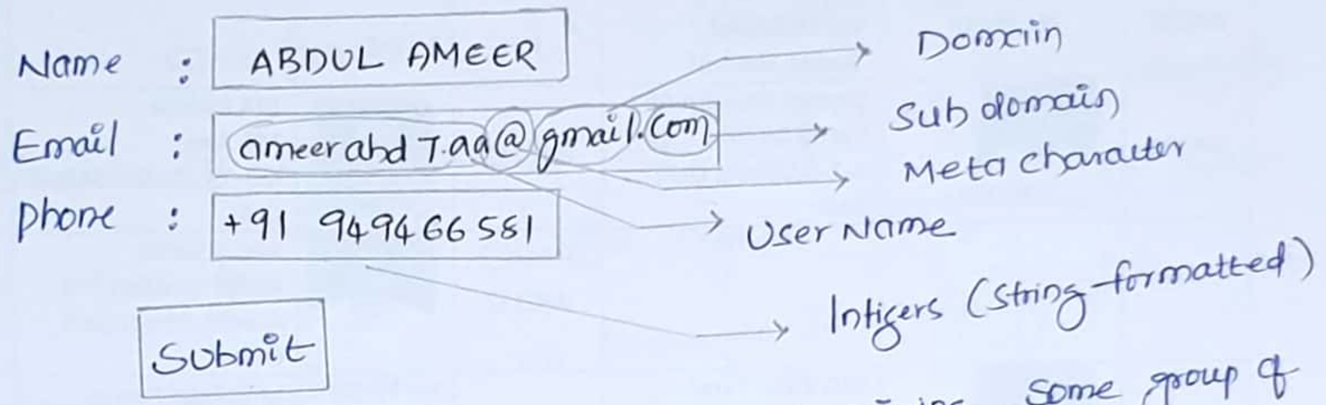
Regular Expressions :-

If we want to represent a group of Strings that fallows a Particular pattern then we Use regular expression This Regular Expressions Used in different formats Such as pdf, files, Search in text document or pdf files etc.

Ex : if we fill a application form where Some blocks are shown below—



i) if We Observe above application Containing Some group of Strings in that intigers, Special characters, alphabet present for Suppose if we Consider Email Column which Contain Some pattern like User name which have (intiger, alphabet, Character Special) then meta character @ is Compulsary if this doesn't appear then the email id is doesn't valid, Domain and Sub domain if any pattern miss this doesnt valid Shows not valid be bind the Comparision operation done by regular Expression it self.

ii) if we Consider phone column, for suppose Consider Indian region most of no's fallows 10 digit no and Start with {9, 8, 7, 6} till 10 digits.

This digits are defined by using of Regular Expressions.

before going to applications of regular expressions we have to discuss some topics such as—

Meta - characters :-

. → dot , ^ → cap , $ → dollar , * → star , + → plus

? → Question , { } → Curly brackets , [ ] → Square Brackets

( ) → open brackets , |, \ → (OR, NOt) this all Symbols Consider as meta characters. by Use of it we Simply Identy-fy the pattern, this all are Used in regular Expressions., in python language and Similar to all languages ie meaning of this character doesn't change in any other languages.

Character classes :- TO represent character of the class we use meta character [ ] in regex.

Ex : [a,b,c] → Search either a or b or c.

So here we find out the pattern as alphabet if any word Consisting with this letters that represent word valid.

[^abc] → Search except a,b,c (ie excluding terms a,b,c

Ex : my Name is Ameer (Including Spaces also)

[a-z] → Match everthing in alphabets.

Ex: My_Name_is_Ameer

In this Capital letters as well as Spaces Execluding remaining all ie Small letters only Included.

[A-Z] → Match all the upper case Letters.

[0-9] → Match all the Numerical digits.

[A-Za-z] → Match all the alphabets (upper + lower)

[^0-9a-zA-Z] → Match everything except alphaNumeric

Predefined character classes :-

[0-9] → Comes under predefined character class. ie only Consider digits this term Can be represent like "\d" Predefined.

[^0-9] → Except digit → represent → "\D"

[0-9 a-z A-Z] → Alpha Numeric → represent → "\w"

In alpha-numeric "_" is included.

[^0-9 a-z A-Z] → Except alphanumeric → "\W"

. → Match everything except new line.

Match all the Spaces → "\s"

Match everything except Spaces → "\S"

Match Newline → "\n"

Match tab characters → "\t"

<u>Quantifiers</u> :- It Specifies how many Instances of Character, group or character class must be present in the Input for a match to be found.

<u>Ex</u> :- $\underline{aa}$ $\underline{a}bc$ $\underline{a}b\underline{a}b$ $\underline{aaa}$ ⇒ There are total '5' matches

In mostly Quantifiers we Used meta characters like *, +

a+ → Atleast One a (1 or more)

a * → Any Number of a's Including zero (zero or more)

Ex ① $\underbrace{aaa}_{1}\underset{\times}{b}\underset{\times}{b}\underbrace{a}_{2}b$

Containing Group of 'a's either 1 or more In first Instance Considering three a's as One group in Other only One a Consider that as One Group becouse + represent atleast "1",(a.or any alphabet)

<u>Ex 2</u> :- $\underset{1}{\boxed{aaa}}\underset{2}{\boxed{b}}\underset{3}{\boxed{b}}\underset{4}{\boxed{a}}\underset{5}{\boxed{b}}\underset{6}{\_}$

a* Normally represents zero a's if we Consider above pattern

a? → Exactly Match 0 or 1 a.

a{n} → Exactly Matches n a's.

a{m,n} → it Matches atleast m & atmost 'n' number of a's.

<u>Ex</u>: a{2,3} → $\underset{\times}{a}bb\underbrace{aa}_{1}s\underbrace{aaa}_{2}b\underset{\times}{a}b\underset{\times}{a}$

above pattern we have matching two groups only which Contain minimum of 2 a's and maximum of 3 a's.

Regex pattern :- $, ^ ⇒ $ → End's with
^ → Starts with ✓

Target :- Ameer Abdul

Regex :- ^A l '→ Represent Starts with 'A', ends with 'l' between all letters are matched.

Note :- [^abc] → Except this letters ; ^a → Starts with 'a'

<u>Rules to defined identifiers</u> :-

i) Allowed characters ⇒ Alpha Numeric ie(A-Z a-z 0-9 and _) underscore.

ii) Identifier Should never Start with a digit

iii) Case Sensitivity

iv) NO length limit

v) Can't Used reversed words for identifier (In python language we have 35 inbuild keywords Such as [True, False, def, del] etc this all are Cant Used in Identifiers.

(Ameer | Sameer) → Either choose ameer or Sameer we use "|" Symbol with ( ) meta characters.

There are Some inbuilt functions also Used Such as find all, match Search etc. in Regular Expressions.

re. match( ) :- * Used to match given pattern at the begining of the target String.

* if it find the pattern than returns match Objects.

* Then we use functions like start(), end(), group() with © match objects, if nothing found returns None.

Ex: Import re

```
regex = input ("a+")
  m = re.match (regex, "abcdefgh")
  print (type(m))
  if m == None:
          print ("Match NOT-Available")

  else:
      print (" match the pattern")
      print (f" match at {m.start}, End{m.end}, P.found {m.group}")
```

O/p ⇒ Match found at beggining of the String.

Match at : 0, End: 1, pattern found: a

By the above i mentioned target if found at Intially ie '0' position and we have found pattern 'a' and ends with (+1) position ie "1" and we have found pattern 'a'

O/p - 2 :- if we give input as _ bcd then _

' Match NOt available".

Because this function Only Consider Intial position In Intially target variable having 'a' at '0''s position. So we have no matches.

re.Search ( ) :- * Search the target irrespective of the location in the first Occurances., Otherwise

* if match it returns founded in the first Occurances., Otherwise returns None. , Similarly Use functions like start, end, group.

I/P Target = 'abcdefghi"
pattern = 'bcd

Syn :- re. Search (pattern, "Target")

O/P :- Match is first Occurance is at : 1, End : 4, p-found : bcd

Note :- Match found at Intial position only, Search found at any where in Target where we mention pattern.

re.findall() :- * Find all matches and returns a list.
it is very powerfull function as Compare to Other functions.

i/p → Target = "0@kM29-1"
pattern = "[0-9]"

Syn → re.findall (pattern, "target")

O/P → ["0", "2", "9", "1"]

Search & Replace – sub():- replace the terms Contain target, pattern & replacement if we want to replace Special Character from role (or) String Contain numeric in that we have to use this function.

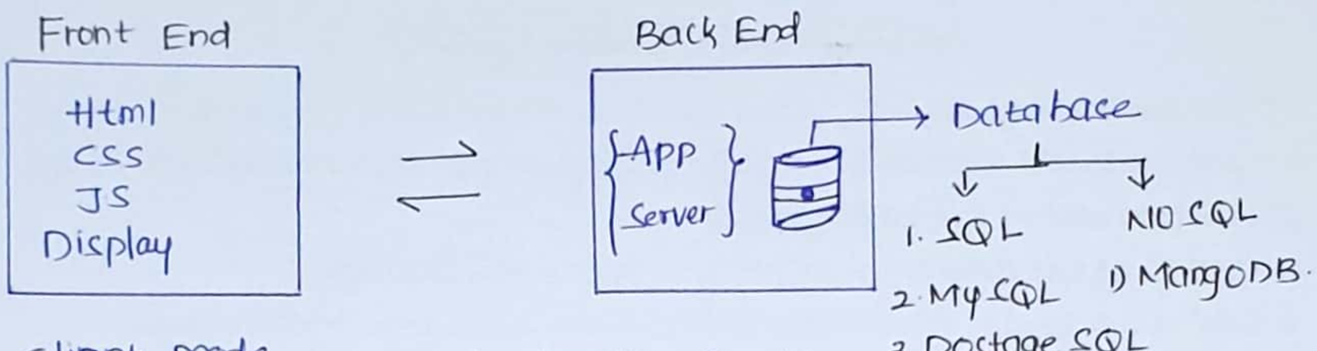
re. Split :- if any Special character diffrential words by Using this Split function we can Solve the issue.

I/p Syn – re. split ('_', "I_Learn_python_Regular-Expressions")

O/P → ['I', 'lear', 'python', 'Regular-Expressions']

Application Development with Flask :-

Web application :- It is a Computer program that Utilizes web browsers and web Technology to perform tasks over the internet.



client mode

Frame works

J.S → React
↳ vue
↳ Angular

Application Servers
↳ programming → python → Flask, Django
↳ Java. → Spring, S.boot
Back End.
Web application
Java Script → Node, Express
C#. → .Net
PHP.

MERN.
- → mango DB
- → Express
- → NODE
- → React → Front End.

mango DB, Express, NODE } BackEnd ⇑ Application Servers

MEAN.
- → MangoDB — Database
- → Express.
- → Angular — front End
- → Node

Back End ⇑ Application servere

FLASK :- Flask is a API of python that allows us to build-up web-Applications.

Flask in VS Code :-

① import library - `from flask import Flask`

② Intialize the flask object - `app = Flask(__name__)`

③ Create an endpoint using a function and assigning a route using a decorator -

```
@app.route('/')
def home():
    return "Hello world"
```

④ Run the application

```
if __name__ = "__main__":
    app.run(debug=True)
```

Dynamic Routing :- It is also called as adaptive routing, is a process where a router can a forward data via diff route Dynamic routing allows as many routes as possible to remain valid in response to change.

3(i) - D.Routing - ("/name"):-

```
@app.route("/<name>")
def about(  ):   → name
    if (name in users):
        return f"welcome {name}"
    else:
        return "User don't exist"
```

i/p -
users ['Ameer', 'Sameer', 'karan']

o/p if we open browser with

ip 127.0.0.1.1500 ⇒ Hello world
1500/Ameer ⇒ welcome Ameer (place holder name we pass Ameer in user it Exist)
1500/Abdul ⇒ User dont exist