**eMAIL VALIDATOR DOCUMENTATION**

1. **Investigation Phase :**

**Rules for email validation :**

1. An email address must contain an @ symbol, which separates the local part from the domain part of the address.
2. The @ symbol cannot be consecutive
3. There cannot be more than one @ symbol in an email
4. The @ symbol is allowed in between single quotes but if another one is added in the email it won’t be valid
5. The local part of the email address (before the @ symbol) can contain letters, numbers, and special characters such as !#$%&'\*+-/=?^\_{|}~, as well as periods (.`), as long as they are not the first or last character and do not appear consecutively.
6. The domain part of the email address (after the @ symbol) must be a valid domain name, which means it must contain at least one period (.) and consist of letters, numbers, and hyphens (-).
7. The top-level domain (the part after the last period) must be a valid top-level domain such as .com, .org, or .edu.
8. The total length of an email address must not exceed 254 characters.
9. The @ symbol has to be more than 6 letters from the end of the email
10. @ symbol is a must
11. No spaces between
12. The local name may be a maximum of 64 letters long
13. Icon Characters aren’t allowed
14. Underscore is not allowed in domain part
15. Spaces, quotes and backslashes must be contained by quotes
16. Last portion of domain must be at least two characters e.g. .com
17. Punctuation marks can not come consecutively
18. Can’t have all punctuation marks
19. Must have the full stop separating the local and domain name
20. @ sign can’t be first
21. Hyphens can’t  be first or last character
22. Normally capital and lowercase letter versions of email work and are allowed
23. Space between “  ” is allowed
24. Any punctuation can go in quotes

**Function Description:**

* 1. **len():** The function is used for finding the length of string passed. It returns the integer value which indicates the length of an object.

For example-len(TestString) is used to find the length of test string.

* 1. **find():** This function is used to find the location of a substring in a given

string. It returns the lowest index of substring found in the string and -1

if the substring is not found.

For example- period=domain.rfind('.') is used to find the period in the domain part of email.

* 1. **rsplit():** rsplit() method returns a list of strings after breaking the given string from the right side by the specified separator.

For example- domain.rsplit('.') is used to split the domain part of test string.

1. **Design Phase:**

**Pseudocode:**

Function checkString(TestString):

Input: TestString (string)

Output: 0 if TestString is not a string, 1 otherwise

1. Set res to the result of checking if TestString is of type string.

2. If res is not True:

a. Return 0.

Function consecutive(TestString):

Input: TestString (string)

Output: 0 if consecutive @ symbols or length exceeds 254, otherwise continue

checking

1. Set t to 0.

2. If the length of TestString is greater than 254:

a. Return 0 and print "Invalid email address".

3. For each element in TestString:

a. If the element is '@', increment t.

b. If the element is ' ', return 0 and print "Invalid email address".

c. If t is equal to 2:

i. Return 0 and print "Invalid email address".

4. If t is equal to 0:

a. Return 0 and print "Invalid email address".

Function localDomain(local, domain):

Input: local (string), domain (string)

Output: 0 if invalid email address, otherwise continue checking

1. Set last to the last character of local.

2. If the first or last character of local is any of '!#$%&',\*+\*=/=?^\_{|~,.',':

a. Return 0 and print "Invalid email address".

3. Initialize a list of special characters.

4. For each element in local:

a. For each character in the list of special characters:

i. If the current element and the next element in local are both equal to the

character:

- Return 0 and print "Invalid email address".

5. Set t to 0.

6. For each element in domain:

a. If the element is '.', increment t.

b. If the element is '\_', return 0 and print "Invalid email address".

c. If the element is not alphabetic, numeric, or '-' return 0 and print "Invalid email

address".

7. If t is less than 1:

a. Return 0 and print "Invalid email address".

8. If local contains any of ' /' ' " ' ' ; ' ' : ' ' ! ' ' ? ' ' - ' ' ' or ' ( ' ') ' ' [ ' ' ] ',

and it is not enclosed in double quotes:

a. Return 0 and print "Invalid email address".

9. If the domain length is less than 6 or the last part of the domain does not match

'com', 'edu', or 'org':

a. Return 0 and print "Invalid email address".

Function ValidateEmail(TestString):

Input: TestString (string)

Output: None (Prints "Invalid email" if conditions are not met)

1. Find the index of '@' in TestString and store it in loc.

2. Extract the local part of the email up to the '@' symbol.

3. Extract the domain part of the email starting from the '@' symbol.

4. If the length of the domain is less than or equal to 6:

a. Print "Invalid email".

5. If the length of the local part is greater than 64:

a. Print "Invalid email".

6. Call localDomain function with local and domain as arguments.

1. Accept user input for the email address and store it in the variable 'email'.

2. Call checkString with email as an argument and store the result in 'valid'.

3. Call consecutive with email as an argument and store the result in 'valid'.

4. If 'valid' is not equal to 0:

a. Print "Valid email".

5. Otherwise:

a. Print "Invalid email".

6. Call ValidateEmail with email as an argument.