Lambda Function

While normal functions are defined using the def keyword, in Python anonymous functions are defined using the lambda keyword.

Syntax of Lambda Function in python

```
lambda arguments: expression
```

Lambda functions can have any number of arguments but only one expression. The expression is evaluated and returned. Lambda functions can be used wherever function objects are required.

def Demo(): ...

1) Anonymous →
2) One line function object

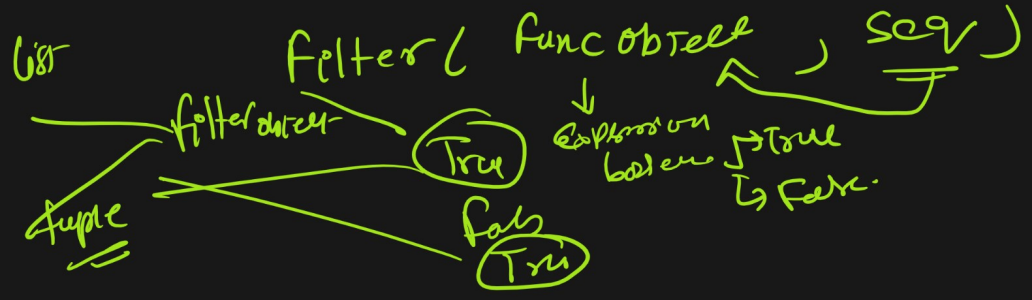
$f(x,y) = x^2 + y^2 + 2xy$

$f(x) = x\%2 == 0$

True / False.

Use of lambda() with filter()

The filter() function in Python takes in a function and a list as arguments. This offers an elegant way to filter out all the elements of a sequence "sequence", for which the function returns True.



input="I43N4D4I43@4123D34E3L44H5I"

Lambda Function

W.A.P remove numbers from string using lambda() or filter() method .

```
Enter string with numbers:-INDIA123DELHI
['I', 'N', 'D', 'I', 'A', 'D', 'E', 'L', 'H', 'I']
INDIADELHI
```

output

INDI@DELHI

Lambda Function

Lambda functions can be used along with built-in functions like **filter()**, **map()** and **reduce()**.

Use of lambda() with map()

The map() function in Python takes in a function and a list as argument. The function is called with a lambda function and a list and a new list is returned which contains all the lambda modified items returned by that function for each item.

Seq = (1, 2, 3,) (5, 6, 7)

fu

map (function object , seq)

Expression → Calculative

List

mapobject

tuple

Reduce the values In one value.

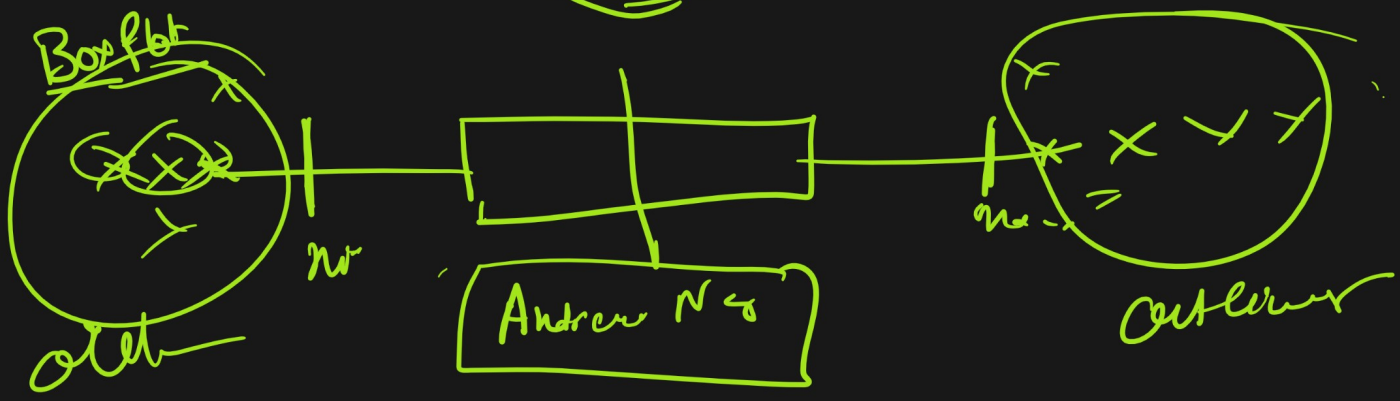
- → Sum
- → multiplication
- → Central tendency → mean, median, mode

21/7

[1, 2, 3, 4, 5, 6] → mean = Sum(values) / Sum(values) = 3.5

[1, 2, 3, 4, 5, 6, 1005] ⇒ (21 + 1005)/7 ≃ 147

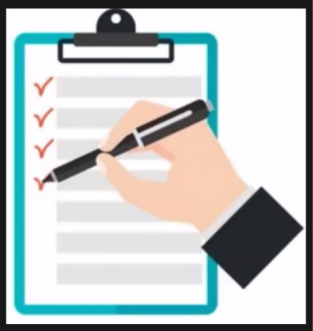
Outlier's



```
/^[Reg]ular[Ex]pression$/
```

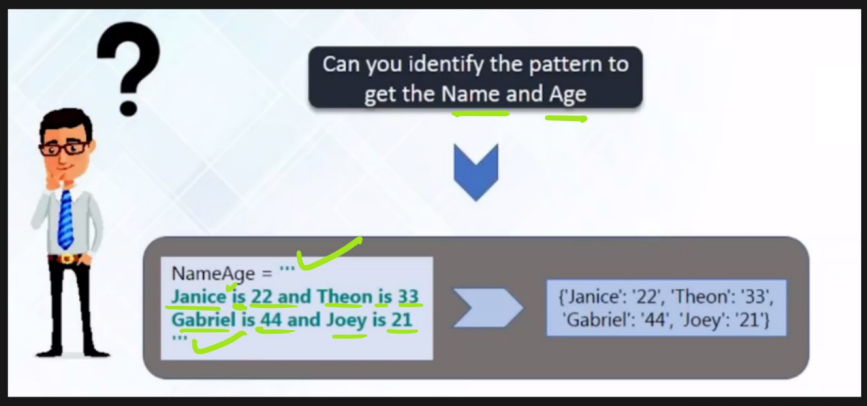
String Object
(Immutable)

- Why we use Regular Expressions?
- What are Regular Expressions?
- Basic Regular Expressions operations



Match Pattern & Return the values.
Matches Pattern & Replace the values.
Validation of the pattern for the values. → Card, Address, Account No, User, Phone etc
Dynamic URL generation. (about / index)

Id → Seq → [0-9]



'[0-9]{1,3}'
or
'\d {1,3}'

Name
'[A-Z][a-z]A'

Range
1. Upper case → A-Z
2. lower case → a-z

Regular Expression → String object
String object → Char By Char

Name
1. Upper case → A-Z
2. lower case → a-z

Sets

[ ] → about one char can be

Chars: Name [A-Z][a-z]+

One Char → [0-9]

[+, - _ @ #] ← use as normal char.
[0-9, A-Z, a-z] ✓ one char

python powered

Sets

A set is a set of characters inside a pair of square brackets [ ] with a special meaning:

Set	Description
[arn]	Returns a match where one of the specified characters (a , r , or n) are present
[a-n]	Returns a match for any lower case character, alphabetically between a and n
[^arn]	Returns a match for any character EXCEPT a , r , and n
[0123]	Returns a match where any of the specified digits (0 , 1 , 2 , or 3) are present
[0-9]	Returns a match for any digit between 0 and 9
[0-5][0-9]	Returns a match for any two-digit numbers from 00 and 59
[a-zA-Z]	Returns a match for any character alphabetically between a and z , lower case OR upper case
[+]	In sets, + , * , . , | , () , $, {} has no special meaning, so [+] means: return a match for any + character in the string

[arn]

a . . .

r . . .

n . .

^ Symbol Inside [ ] brackets or Expression or value

* [^arn] → except those chars → a r n any other char

* ^ Expression / value → / start with same the Expression / value

Use first char matching.

* Range of Char's to be match.

⇒ {min, max}

{max}

→ [0-9] [0-9] [0-9] [0-9]

[0-9] {4}

→ Age

8 yr → 1

25 yr → 2

102 yr → 3

⇒ [0-9] {1, 3}

Special Sequences

A special sequence is a \ followed by one of the characters in the list below, and has a special meaning:

Character	Description	Example
\A	Returns a match if the specified characters are at the beginning of the string	"\AThe"
\b	Returns a match where the specified characters are at the beginning or at the end of a word	r"\bain" r"ain\b"
\B	Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word	r"\Bain" r"ain\B"
\d	Returns a match where the string contains digits (numbers from 0-9)	"\d"
\D	Returns a match where the string DOES NOT contain digits	"\D"
\s	Returns a match where the string contains a white space character	"\s"
\S	Returns a match where the string DOES NOT contain a white space character	"\S"
\w	Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character)	"\w"
\W	Returns a match where the string DOES NOT contain any word characters	"\W"
\Z	Returns a match if the specified characters are at the end of the string	"Spain\Z"

⇒ [0-9] → \d

⇒ [A-Za-z0-9_] → \w

⇒ [ ] → \S ∉ 20}

Metacharacters

Metacharacters are characters with a special meaning:

Character	Description	Example
[]	A set of characters	"[a-m]"
\	Signals a special sequence (can also be used to escape special characters)	"\d"
.	Any character (except newline character)	"he..o"
^	Starts with	"^hello"
$	Ends with	"world$"
*	Zero or more occurrences	"aix*"
+	One or more occurrences	"aix+"
{}	Exactly the specified number of occurrences	"al{2}"
|	Either or	"falls|stays"
()	Capture and group	

[+]

\+

\d → 0-9

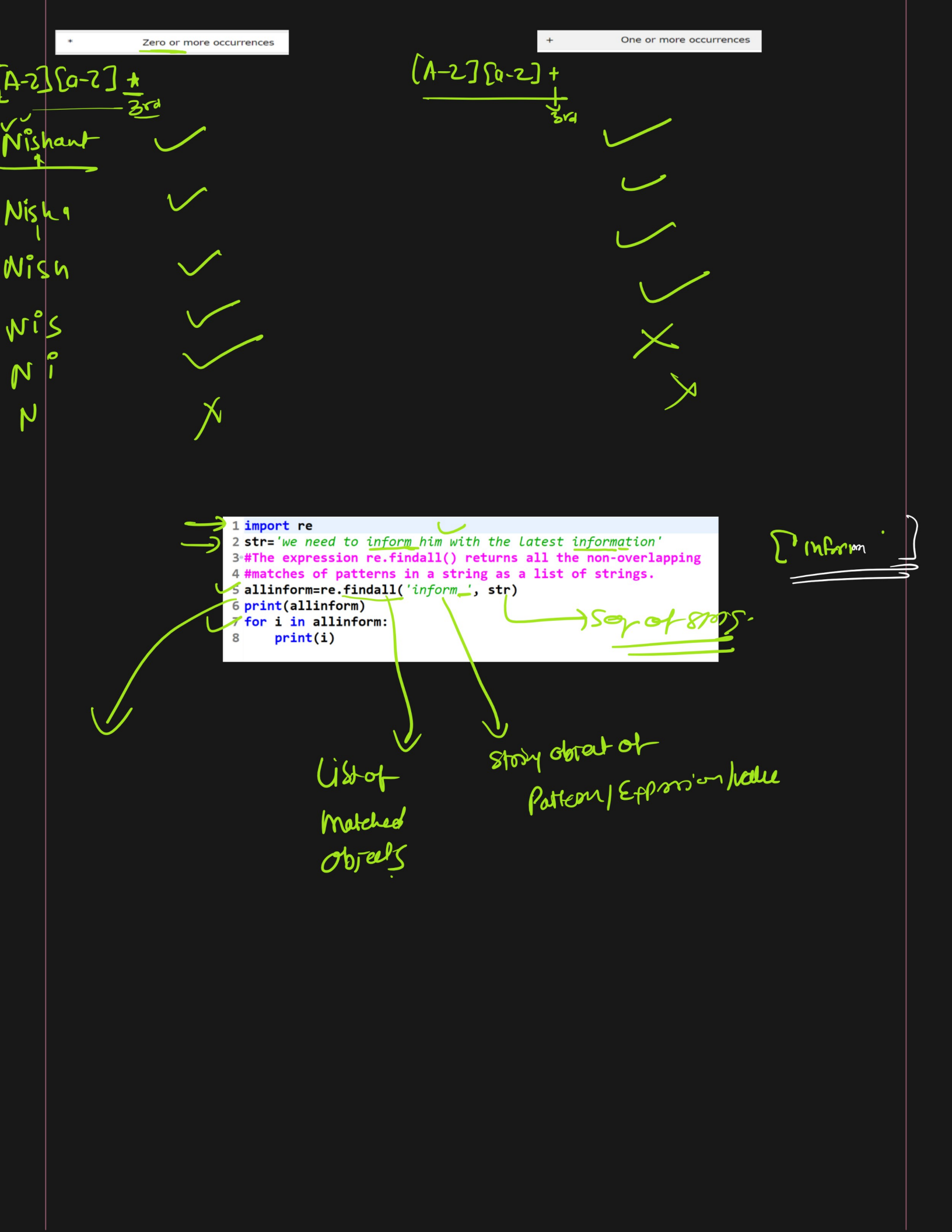
He.. o

Hello

He o

He12o

Heooo


* Zero or more occurrences
[A-Z][a-z] *
3rd
Nishant
Nisha
Nish
Nis
Ni
N
+ One or more occurrences
[A-Z][a-z] +
3rd
1 import re
2 str='we need to inform him with the latest information'
3 #The expression re.findall() returns all the non-overlapping
4 #matches of patterns in a string as a list of strings.
5 allinform=re.findall('inform', str)
6 print(allinform)
7 for i in allinform:
8 print(i)
['inform'
Seq of strings.
List of Matched Objects
String object of Pattern / Expression / value

```python
import re

str="Sat hat mat pat"
out=re.findall('[shmpa]at',str)
print(out)
```

[shmpa]at

⇒ 3 char

→ sat ✗
→ hat ✓
→ mat ✓
→ pat ✓
→ aat ✗

Output

['hat', 'mat', 'pat']

```python
import re

str="Sat hat mat paat #at_at.at 9at (at"
out=re.findall('^[shmpa]at',str)
print(out)
```

[ ] list

Start with

→ ↑ Expression/value

→ ↑ [Shmpa]at

→ ↑ Hello

Sat
hat
mat
pat
aat

No

```python
import re

str="@at hat mat paat #at_at.at 9at (at"
out=re.findall('^[^shmpa]at',str)
print(out)
```

↑ Start with

['@at']

Any char except s h m p a

• at

```python
import re

str="Sat hat mat paat #at_at.at 9at (at"
out=re.findall('^[shmpa]at',str)
print(out)
```

Start with

[ ]

Sat ✓
hat ✓
mat ✓
paat ✓
at ✗

Probs-1

Here we are supposed to verify the phone numbers

Phone Number Verification

['+91-222-81345','+82-333-82456','+56-356-67894','+43-333-89456','+99-888-67897','+56-5678-67890', '+91-676-8905678','1+91-222-81345']

Problem Statement:
To verify the phone numbers

13

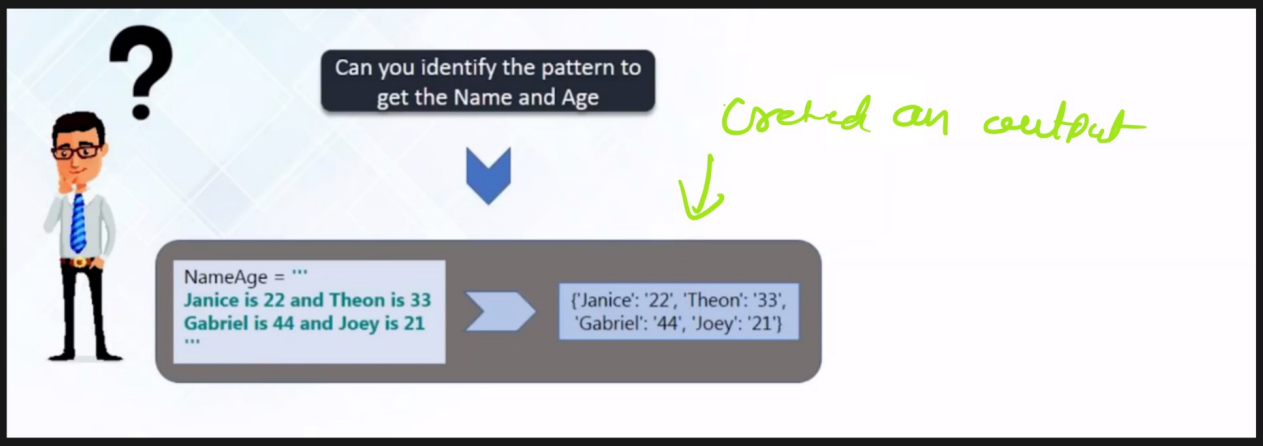
'+54-239-80832'

+ 98 - 1-6 0-9 0-9 0-9 - 0-9 0-9 0-9 0-9 0-9

Output

Valid Number List

Prob 2



Created an output

Prob 3

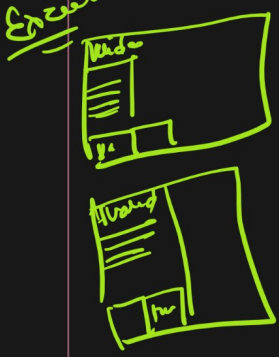
Problem Statement:
To verify the E-main addresses

E-mail Verification

Source of Raw Email Input → Outside Python (OneDrive)

Output
1. Valid Email list with total count
2. Create & write A Valid or Invalid .txt file.
3. Create A Excel sheet that contains
   - Valid Sheet → valid Email
   - Invalid Sheet → Invalid Email

Excel



E-mail:
Saurabh@gmail.com
Reyshma @ com
Kv .com
123 @.com

E-mail address should include:
- 1 to 20 lowercase and uppercase letters, numbers, plus ._%+-
- An @ symbol
- 2 to 20 lowercase and uppercase letters, numbers, plus .-
- A period
- 2 to 3 lowercase and uppercase letters