UNIT - III

Regular Expressions

- Introduction / Motivation
- Special Symbols and characters
- REs and Python.

OBJECT Oriented Programing in PYTHON

- classes , self-variable
- Methods , constructor Method
- Inheritance
- Overriding Methods
- Data hiding

ERROR and Exceptions

- Difference b/w an error and Exception
- Handling exceptions
- try, except block
- Raising Exceptions
- User Defined Exceptions.

Regular Expression :— A Regular Expression (RegEx) is a sequence of characters that defines a search pattern.

— RegEx can be used to check if a string contains the specified search pattern. It can detect the presence or absence of a text by matching it with a particular pattern, and also can split a pattern into one or more sub patterns.

— Python provides a re module that supports the use of regex in Python. Its primary function is to offer a search, where it takes a regular expression and a string.

```
import re
str = 'docs.python.org : A python Portal '
match = re.search(r'portal', str)
print('Start Index:', match.start())
print('End Index:', match.end())
```

```
o/p
start Index : 27
End Index: 33
```

RegEx Module :—

Python has a built-in package called re, which can be used to work with Regular Expressions.

import re

when we imported the re module, we can start using regular expressions.

RegEx Functions :—

The re module offers a set of functions that allows us to search a string for a match.

— findall — search — split — sub

findall() — returns a list containing all matches.

```
import re
txt = "The rain in ~~Hyd~~ spain"
x = re.findall("ai", txt)
print(x)
```

O/P

```
['ai', 'ai']
```

- Here the list contains the matches in the order they are found.
- If no matches are found, an empty list is returned.

Search() — searches the string for a match, and returns a Match object if there is a match.

- If there is more than one match, only the first occurrence of the match will be returned.

eg Search for first white-space character in the string

```
import re
txt = "The rain in spain"
x = re.search("\s", txt)
print("The first white-space character is located in position:",
                                    x.start())
```

print(x) → returns matchobject

O/P: The first white-space character is located in position: 3

- If no matches are found, the value None is returned.

split() — returns a list where the string has been split at each match. eg split at each white-space character

```
import re
txt = "The rain in spain"
x = re.split("\s", txt)
print(x)
```

```
['The', 'rain', 'in', 'spain']
```

- we can control the no.of occurrences by specifying the maxsplit parameter.

eg split the string only at the first occurrence.

```
import re
txt = "The rain in spain"
x = re.split("\s", txt,1)
print(x)
```

o/p

```
['The', 'rain in spain']
```

sub() — This function replaces the matches with the text of your choice.

eg: replace very white-space character with the no. 9.

```
import re
txt = "The rain in spain"
x = re.sub("\s", "9", txt)
print(x)
```

o/p

```
The9rain9in9spain
```

- we can control the no.of replacements by specifying the count parameter.

eg replace the first 2 occurrences:

```
import re
txt = "The rain in spain"
x = re.sub("\s", "9", txt,2)
print(x)
```

o/p

```
The9rain9in spain
```

Match object :— It is an object containing information about the search and the result. If there is no match, the value None will be returned, instead of the Match object.

– The Match object has properties and methods used to retrieve information about the search, and the result.

. span() – returns a tuple containing the start and end positions of the match.

. string() – returns the string passed into the function.

. group() – returns the part of the string where there was a match.

eg print the position (start and end position) of the first match occurrence.

The regular expression looks for any words that starts with an upper case "S".

```
import re
txt = " The rain in spain"
x = re.search(r"\bS\w+", txt)
print(x.span())
```

o/p

```
(2,17)
```

here it search for an upper case "S" character in the beginning of a word, and print its position.

eg print the string passed into the function.

```
import re
txt = "The rain in spain"
x = re.search( r"\bS\w+", txt)
print(x.string)
```

o/p

```
The rain in spain
```

here the string property returns the search string.

eg

~~eg~~ print the part of the string where there was a match.

Regular expression looks for any words that starts with a upper case "S".

```
import re
txt = "The rain in Spain"
x = re.search(r"\bS\w+", txt)
print(x.group())
```

o/p
Spain

Here search for an upper case "S" character in the beginning of a word, and print the word.

Meta characters :—

Meta characters are characters with a special meaning.

\ - used to drop the special meaning of character following it.
[ ] - represent a character class.
^ - matches the beginning
$ - Matches the end.
. - matches any character except newline.
| - any matches with any of the characters separated by it.
? - Matches zero or one occurrence.
* - Any no.of occurrences (including 0 occurrences)
+ - one or more occurrences Indicate it
{ } - Indicate the no.of occurrences of a preceding regex to match.
( ) - Enclose a group of Regex.

① \ - Backslash - is used to escape various characters including all meta characters. The backslash (\) makes sure that the character is not treated in a special way. for eg,

```
import re
txt = "cost is 59 dollars"
x = re.findall("\d", txt)
print(x)
```

o/p

```
['5', '9']
```

② [ ] - square brackets - specifies a set of characters you wish to match.

expression	string	matched?
[abc]	a	1 match
	ac	2 matches
	Hey	No match
	abcdeca	5 match.

Here, [abc] will match if the string you are trying to match contains any of the a, b or c.

⇒ we can also specify a range of characters using - inside square brackets.

- [a-e] - same as [abcde]
- [1-4] - same as [1234]

⇒ we can complement (invert) the character set by using caret ^ symbol at the start of a square-bracket.

- [^abc] - means any character except a or b or c
- [^0-9] - means any non-digit character.

⇒ eg

```
import re
txt = "The rain in spain"
x = re.findall("[a-m]", txt)
print(x)
```

o/p

```
['h', 'e', 'a', 'i', 'i', 'a', 'i']
```

③ **^ — caret** - matches the beginning of the string i.e checks whether the string starts with the given character(s) or not.

expression	string	matched?
^a	a	1 match
	abc	1 match
	bac	No match
^ab	abc	1 match
	acb	No match (starts with a but not followed by b).

```
import re
txt = "hello planet"
x = re.findall("^hello", txt)
if x:
    print("Yes, the string starts with 'hello'")
else:
    print("No match")
```

o/p

```
Yes, the string starts
     with 'hello'
```

④ **$ - dollar** - used to check if a string ends with a certain character.

expression	string	matched?
a$	a	1 match
	formula	1 match
	cab	No match.

```
import re
txt = "hello planet"
x = re.findall("planet$", txt)
if x:
    print("yes, the string ends with 'planet'")
else:
    print("No match")
```

o/p

```
yes, the string ends with
        'planet'.
```

⑤ <u>. — Period</u> — matches any single character (except newline '\n').

— a.b will check for the string that contains any character at the place of the dot such as acb, acbd, abbbck.

— .. will check if the string contains at least 2 characters.

```
import re
txt = "hello planet"
x = re.findall("he..o", txt)
print(x)
```

o/p
```
['hello']
```

Here, search for a sequence that starts with "he", followed by two (any) characters, and an "o"

⑥ <u>| — Alternation</u> — used for alternation (or operator). or symbol is present in the string or not.

expr	string	matched?
	cde	No match
a|b	ade	1match (match at ade)
	acdbea	3matches (at acdbea)

```
import re
txt = "The rain in spain falls mainly in the plain!"
x = re.findall("falls|stays", txt)
print(x)
if x:
    print("Yes, there is at least one match")
else:
    print("No match")
```

o/p
```
['falls']
Yes, there is atleast one match.
```

— Here, we are checking if the string contains either falls or stays.

⑦ ? - Question mark - matches zero or one occurrence of the pattern left to it.

- checks if the string before the question mark in the regex occurs atleast once or not at all.

eg ab?c will be matched for the string ac, acb, dabc but will not be matched for abbc because there are two b. similarly, it will not be matched for abdc because b is not followed by c.

expr	string	matched?
	mn	1 match
ma?n	man	1 match
	maaan	no match (more than one a char)
	main	no match (a is not followed by n)
	woman	1 match.

```
import re
txt = "hello planet"
x = re.findall("he.?o", txt)
print(x)
```

no match.

Here, search for a sequence that starts with "he", followed by 0 or 1 (any) character, and an "o":

- here we got no match because there were not zero, not one, but two characters b/w "he" and "o"

⑧. * - star - matches zero or more occurrences of the pattern left to it. eg ab*c will be matched for the string ac, abc, abbbc, dabc, etc. but will not be matched for abdc because b is not followed by c.

~~impor~~ expr	string	matched?
	mn	1 match
	man	1 match
ma*n	maaan	1 match
	main	no match (a is not followed by n)
	woman	1 match.

```
import re
txt = "hello planet"
x = re.findall("he.*o", txt)
print(x)
```

o/p
['hello']

- search for a sequence that starts with "he", followed by 0 or more (any) characters, and an "o":

(9) **+ - plus** - matches one or more occurrences of the pattern left to it. eg ab+c will be matched for the string abc, abbc, dabc, but will not be matched for ac, abdc because there is no b in ac and b is not followed by c in abdc.

expr	string	matched?
	mn	no match (no a character)
	man	1 match
ma+n	maaan	1 match
	main	no match (a is not followed by n)
	woman	1 match.

```
import re
txt = "hello planet"
x = re.findall("he.+o", txt)
print(x)
```

```
['hello']
```

- Search for a sequence that starts with "he", followed by 1 or more (any) characters, and an "o".

(10) {} - Braces - match any repetitions preceding regex from m to n both inclusive. eg a{2,4} will be matched to the string aaab, baaaac, gaad, but will not be matched for strings like abc, bc because there is only one a or no a in both the cases.

expr	string	matched?
	abc dat	no match
	abc daat	1 match (at daat)
a{2,3}	aabc daaat	2 matches (at aabc and daaat)
	aabc daaaat	2 matches (at aabc and daaaat)

```
import re
txt = "hello planet"
x = re.findall("he.{2}o", txt)
print(x)
```

```
['hello']
```

- search for a sequence that starts with "he", followed by exactly 2 (any) characters, and an "o".

(11) ( ) - Group - used to group sub patterns. eg (a|b|c)xz match any string that matches either a or b or c followed by xz

regx	string	matched?
	ab xz	no match
(a|b|c)xz	abxz	1 match (match at abxz)
	axz cabxz	2 match (at axzbc cabxz)

eg (a|b)cd will match for strings like acd, abcd, gacd, etc.

Special Sequences :—

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

\A - Returns a match if the specified characters are at the beginning of the string

eg "\AThe" - Check if the string starts with "The"

```
import re
txt = "The rain in spain"
x = re.findall("\AThe", txt)
print(x)
if x:
    print("Yes, there is a match!")
else:
    print("No match!)
```

o/p

```
['The']
Yes, there is a match!
```

\b - returns a match where the specified characters are at the beginning or at the end of a word

(r in the beginning is making sure that the string is being treated as a raw string).   r"\bain"
r"ain\b"

```
import re
txt = "The rain in spain"
x = re.findall("r"\bain", txt) → []
                                  no match
print(x)          check if 'ain' is present at beginning of a word.
if x:
    print("Yes, atleast one match")
else:
    print("no match")
```

---

x = re.findall(r"ain\b", txt) - ['ain', 'ain']
· yes atleast one match
↓
check if ain is present at the end of a word

\B - returns a match where the specified characters are present, but not at the beginning (or at end) of a word.

r"\Bain"
r"ain\B"

x = re.findall(r"\Bain", txt) - ['ain', 'ain']
↓
check if 'ain' is present, but not at beginning of a word

x = re.findall(r"ain\B", txt) - [] no match.
↓
check if 'ain' is present, but not at end of a word.

\d — returns a match where the string contains digits (0-9)

x = re.findall("\d", txt) — [ ] no match

\D — returns a match where the string does not contain digits

```
import re                                  o/p
txt = " apple "                            ['a','p','p','l','e']
x = re.findall("\D", txt)                  matched"
print(x)
if (x):
    print(" matched")
else:
    print("no match")
```

\s — returns a match where the string contains a white space character.

x = re.findall("\s", txt) — [' ', ' ', ' ']

\S — returns a match where the string does not contain a white space character.

x = re.findall("\S", txt) — ['T','h','e' . . . . 'i','n']

\w — returns a match where the string contains any word characters (chars from a to z, digits -0-9, and _).

x = re.findall ("\w", txt) — ['T','h', ... 'i','n']

— returns a match at every word character

\W - returns a match where the string does not contain any word characters

x = re.findall("\W", txt) - [' ', ' ', ' ']

↓

- returns a match at every non-word character.

\Z - returns a match if the specified characters are at the end of the string. "spain\Z"

x = re.findall("Spain\Z", txt) - ['spain']

↓

check if the string ends with spain.

Sets: A set is a set of characters inside a pair of square brackets with a special meaning.

[arn] - returns a match where one of the specified characters (a, r, or n) is present

x = re.findall("[arn]", txt) - ['r', 'a', 'n', 'n', 'a', 'n']

[a-n] - returns a match for any lowercase character, alphabetically b/w a and n.

x = re.findall("[a-n]", txt) - ['h', 'e', 'a', 'i', 'n', 'i', 'n', ...

↓

check if the string has any characters b/w a and n

[^arn] - returns a match for any character except a, r, n

x = re.findall("[^arn]", txt) - ['T', 'h', 'e', ' ', ...]

↓

check if the string has other characters than a, r, n

[0123] - returns a match ~~for~~ any digit blw any of the specified digits 0,1,2,3,~~3~~ are present

x= re.findall("[0123]", txt) - [ ]

[0-9] - returns a match for any digit b/w 0 and 9

x = re.findall("[0-9]", txt) - [ ], No match
↓
check if the string has any digits.

[0-5][0-9] - returns a match for any two-digit no's from 00-59

```
import re
txt = "8 times before 11:45 A.M"
x = re.findall("[0-5][0-9]", txt)
print(x)
if(x):
    print("matched")
else:
    print("nomatch")
```

| ['11', '45']
| matched

[a-zA-Z] - returns a match for any characters alphabetically b/w a-z, A-Z

x= re.findall("[a-zA-Z]", txt)
↓
check if the string has any characters from a to z lowercase and A to Z uppercase.

[+] - In sets +, *, ., |, ( ), $, { } has no special meaning
So [+] means: - returns a match for any + character in the string

x= re.findall("[+]", txt)

| o/p
| - [ ]
| No match.

↓
check if the string has any + characters.