Regular Expression Examples

[] – Square Brackets

Square Brackets ([]) represents a character class consisting of a set of characters that we wish to match. For example, the character class [abc] will match any single a, b, or c.

We can also specify a range of characters using – inside the square brackets. For example,

- [0, 3] is sample as [0123]
- [a-c] is same as [abc]

We can also invert the character class using the caret(^) symbol. For example,

- [^0-3] means any number except 0, 1, 2, or 3
- [^a-c] means any character except a, b, or c

Sets

A set is a group of characters given inside a pair of square brackets. It represents the special meaning.

SN	Set	Description
1	[arn]	Returns a match if the string contains any of the specified characters in the set.
2	[a-n]	Returns a match if the string contains any of the characters between a to n.
3	[^arn]	Returns a match if the string contains the characters except a, r, and n.
4	[0123]	Returns a match if the string contains any of the specified digits.
5	[0-9]	Returns a match if the string contains any digit between 0 and 9.
6	[0-5][0-9]	Returns a match if the string contains any digit between 00 and 59.
10	[a-zA-Z]	Returns a match if the string contains any alphabet (lower-case or upper-case).

Example

```
import re

str = "How are you. How is everything"

matches = re.findall("How", str)

print(matches)

print(matches)
```

Output:

```
['How', 'How']
```

Example

```
import re

str = "How are you. How is everything"

matches = re.search("How", str)

print(type(matches))

print(matches) #matches is the search object
```

Output:

```
<class '_sre.SRE_Match'>
<_sre.SRE_Match object; span=(0, 3), match='How'>
```

Example

```
import re

str = "How are you. How is everything"

matches = re.search("How", str)

print(matches.span())

print(matches.group())

print(matches.string)
```

Output:

```
(0, 3)
How
How are you. How is everything
```

re.split():

This function helps to split string by the occurrences of given pattern. The returned object is the list of slices o

```
>>> import re
>>> string="Simple is better than complex."
>>> obj=re.split(r' ',string)
>>> obj
['Simple', 'is', 'better', 'than', 'complex.']
```

The string is split at each occurrence of a white space ' ' returning list of slices, each corresponding to a word. I
output is similar to split() function of built-in str object.

```
>>> string.split(' ')
['Simple', 'is', 'better', 'than', 'complex.']
```

re.sub():

This function returns a string by replacing a certain pattern by its substitute string. Usage of this function is :

```
re.sub(pattern, replacement, string)
```

In the example below, the word 'is' gets substituted by 'was' everywhere in the target string.

```
>>> string="Simple is better than complex. Complex is better than complicated."
>>> obj=re.sub(r'is', r'was',string)
>>> obj
```

'Simple was better than complex. Complex was better than complicated.'

Finding word starting with vowels

```
>>> string='Errors should never pass silently. Unless explicitly silenced.'
>>> obj=re.findall(r'\b[aeiouAEIOU]\w+', string)
>>> obj
['Errors', 'Unless', 'explicitly']
```

Replace domain names of all email IDs in a list.

```
>>> emails=['aa@xyz.com', 'bb@abc.com', 'cc@mnop.com']
>>> gmails=[re.sub(r'@\w+.(\w+)','@gmail.com', x) for x in emails]
>>> gmails
['aa@gmail.com', 'bb@gmail.com', 'cc@gmail.com']
```