Regular Expression

1)pattern is: any five-letter string starting with a and ending with s.

Match()

```
import re

pattern = '^a...s$'

test_string = 'apass'

result = re.match(pattern, test_string)
```

Metacharacters:

Metacharacters are characters that are interpreted in a special way by a RegEx engine. Here's a list of metacharacters:

[] . ^ $ * + ? {} () \ |

[] - Square brackets

Pattern="[abc]”

```
1) import re
pattern="[0-9]"
string_str="abc123"
result=re.match(pattern,string_str)
print(result)
```

. – Period

A period matches any single character (except newline '\n').

```
import re
pattern=".."
string_str="ab"
result=re.match(pattern,string_str)
print(result)
```

^ - Caret

The caret symbol ^ is used to check if a string starts with a certain character.

```
import re
pattern=".."
string_str="ab"
```

```
result=re.match(pattern,string_str)
print(result)
```

$ - Dollar

The star symbol * matches zero or more occurrences of the pattern left to it.

```
import re

pattern = "a$"
string_str = "formula"
result = re.findall(pattern,
string_str)
print(result)
```

*** - Star**

The star symbol * matches zero or more occurrences of the pattern left to it.

```
import re

pattern = "Thi*"
string_str = "This if Python Tutorial
This"
result = re.findall(pattern,
string_str)
print(result)
```

+ - Plus

The plus symbol + matches one or more occurrences of the pattern left to it.

```
import re

pattern = "Thi+"
string_str = "This if Python Tutorial
Thi"
result = re.findall(pattern,
string_str)
print(result)
```

? - Question Mark

The question mark symbol? matches zero or one occurrence of the pattern left to it.

```
import re

pattern = "ma?n"
string_str = "mn"
result = re.findall(pattern,
string_str)
print(result)
```

{} – Braces

Consider this code: {n,m}. This means at least n, and at most m repetitions of the pattern left to it.

Matches exactly the specified number of occurrences

```
import re

pattern = "a{2,3}"
string_str = "abc dat"
result = re.findall(pattern,
string_str)
print(result)
```

This RegEx [0-9]{2, 4} matches at least 2 digits but not more than 4 digits

```
import re

pattern = "[0-9]{2,4}"
string_str = "12 and 345673"
result = re.findall(pattern,
string_str)
print(result)
```

Not Match:

```
import re

pattern = "[0-9]{2,4}"
string_str = "12 and 345673 1"
result = re.findall(pattern,
string_str)
print(result)
```

| - Alternation

Vertical bar | is used for alternation (or operator).

```
import re

pattern = "a|b"
string_str = "ade"
result = re.match(pattern, string_str)
print(result)
```

() – Group

(a|b|c)xz match any string that matches either a or b or c followed by xz

```
import re

pattern = "(a|b|c)xz"
string_str = "axz cabxz"
result = re.findall(pattern,
string_str)
print(result)
```

\ - **Backslash**

Backlash \ is used to escape various characters including all metacharacters.

```
import re

pattern = "\$a"
string_str = "$abc"
result = re.findall(pattern,
string_str)
print(result)
```

search () function

Scans a string for a regex match

```
import re

pattern = "This"
string_str = "This is android studio"
result = re.search(pattern, string_str)
print(result)
```

split() function

Splits a string into substrings using a regex as a delimiter

```
import re

pattern = "\s"
string_str = "This is android studio"
result = re.split(pattern, string_str)
print(result)
```

Special Sequences

\A - Matches if the specified characters are at the **start** of a string.

```
import re

pattern = "\Athe"
string_str = "the sun"
result = re.findall(pattern,
string_str)
print(result)
```

\b - Matches if the specified characters are at the beginning or end of a word.

```
import re

pattern = r"\bfoo"
string_str = "football"
result = re.findall(pattern,
string_str)
print(result)
```

```
import re

pattern = r"foo\b"
string_str = "the afoo test"
result = re.findall(pattern,
string_str)
print(result)
```

\B - Opposite of \b. Matches if the specified characters are not at the beginning or end of a word.

```
import re

pattern = r"\Bfoo"
string_str = "afootball"
result = re.findall(pattern,
string_str)
print(result)
```

\d - Matches any decimal digit. Equivalent to [0-9]

```
import re

pattern = r"\D"
string_str = '1ab34"50'
result = re.findall(pattern,
string_str)
print(result)
```

\s - Matches where a string contains any whitespace character. Equivalent to [\t\n\r\f\v].

```
import re

pattern = r"\s"
string_str = 'Python RegEx'
result = re.findall(pattern,
string_str)
print(result)
```

\S - Matches where a string contains any non-whitespace character. Equivalent to [^ \t\n\r\f\v].

```
import re

pattern = "\S"
string_str = 'ab'
result = re.findall(pattern,
string_str)
print(result)
```

\w - Matches any alphanumeric character (digits and alphabets). Equivalent to [a-zA-Z0-9_]. By the way, underscore _ is also considered an alphanumeric character.

```
import re

pattern = "\w"
string_str = '12&": ;c_'
```

```
result = re.findall(pattern,
string_str)
print(result)
```

\W - Matches any non-alphanumeric character. Equivalent to [^a-zA-Z0-9_]

```
import re

pattern = "\W"
string_str = '1a2%c'
result = re.findall(pattern,
string_str)
print(result)
```

\Z - Matches if the specified characters are at the end of a string.

```
import re

pattern = "Python\Z"
string_str = 'I like Python'
result = re.findall(pattern,
string_str)
print(result)
```

re.sub()

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

The method returns a string where matched occurrences are replaced with the content of replace variable.

```
import re

pattern = "\s+"
string_str = 'I like Python'
replace = ''
result = re.sub(pattern,
replace,string_str)
print(result)
```

re.subn()
is similar to re.sub() except it returns a tuple of 2 items containing the new string and the number of substitutions made.

```
import re

pattern = "\s+"
string_str = 'I like Python '
replace = ''
result = re.subn(pattern,
replace,string_str)
print(result)
```