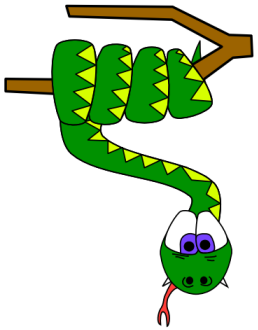

Regular Expression

Session-X

Regular Expressions

```
"""identifiers
\w any character \W anything but character
\s for whitespace \S anything but whitespace
\d number \D anything but number
Modifiers
. any character, * 0 or more, + 1 or more, ? 0 or one
$ end of the string, ^ begining of the string, [] variance
{} range, () group """
import re
ex_s="""jessica is 9 year old and her email id is jessi@gmail.com
and jes@hotmail.com, 234 dwnstreet"""
mt=re.search(r'ssi', ex_s)
print(mt)
print(mt.group())
mt=re.search(r'\d{3}\s\w+', ex_s)
print(mt.group())
mt=re.search(r'[\w.]+@[\w.]+', ex_s)
print(mt.group())
```

Regular Expression...

```
import re
#re.split() returns list and work similar to string with added
functionalities
print(re.split(r"\s*", "here are some words"))
print(re.split(r"s*", "here are some words"))
print(re.split(r"(s*)", "here are some words"))
print(re.split(r"[a-fA-F]", r"adaslkf
dewebekwcbeffertyuyujinHHklhjkFFFjloo[o"))
print(re.findall(r'\d\s\D*\d?', 'BYL93 LIG Barra-2 kanpur'))
match = re.findall(r'([\w.]+)@([\w.]+)', 'my email is
sumit.chandra@psit.in and chandra_sumit@rediffmail.com')
print(match)
"""
* 0 or more
+ 0 or more
? 0 or One
{5} exactly five
{1,5} one to five
\w alphanumeric
"""
```

Regular Expressions...

- **Re.sub.** In regular expressions, sub stands for substitution. The re.sub method applies a method to all matches.
- Re.sub can replace a pattern match with a simple string.

```
import re

# An example string.
v = "running eating reading"

# Replace words starting with "r" and ending in "ing"
# ... with a new string.
v = re.sub(r"r.*?ing", "ring", v)
print(v)
```

Regular Expression...

```
import re

def multiply(m):
    # Convert group 0 to an integer.
    v = int(m.group(0))
    # Multiply integer by 2.
    # ... Convert back into string and return it.
    return str(v * 2)

# Use pattern of 1 or more digits.
# ... Use multiply method as second argument.
result = re.sub("\d+", multiply, "10 20 30 40 50")
print(result)
```

Regular Expressions

- Subn. Usually re.sub() is sufficient. But another option exists. The re.subn method has an extra feature. It returns a tuple with a count of substitutions in the second element.

```
import re
def add(m):
    # Convert.
    v = int(m.group(0))
    # Add 2.
    return str(v + 1)
# Call re.subn.
result = re.subn("\d+", add, "1 2 3 4 5")
print("Result string:", result[0])
print("Number of substitutions:", result[1])
```

O/p

- Result string: 2 3 4 5 6
- Number of substitutions: 5