Topics

Continuation of regular

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	

In [4]:

```python
## ^ symbol

import re
print(re.search("^APSSDC","APSSDC"))
print(re.search("APS","SSDCDAPSSDC"))
print(re.search("^APS","SSDCAPSSDC"))
```

```
<re.Match object; span=(0, 6), match='APSSDC'>
<re.Match object; span=(5, 8), match='APS'>
None
```

In [6]:

```python
print(re.search("DC$","APSSDC"))
print(re.search("DC","SSDCDAPSSDC"))
print(re.search("DC$","SSDCAPSSDCSS"))
```

```
<re.Match object; span=(4, 6), match='DC'>
<re.Match object; span=(2, 4), match='DC'>
None
```

In [11]:
```python
# *

print(re.search("S*","APSSDC"))
print(re.search("S*","APDC"))
print(re.search("S","APDC"))
print(re.search(".*",""))
print(re.search("."," "))
```

```
<re.Match object; span=(0, 0), match=''>
<re.Match object; span=(0, 0), match=''>
None
<re.Match object; span=(0, 0), match=''>
<re.Match object; span=(0, 1), match=' '>
```

In [14]:
```python
print(re.search("S+","APSDCSS"))
print(re.search("S+","APSDC"))
print(re.search("S+","APDC"))
print(re.search(".+","A"))
print(re.search(".+",""))
```

```
<re.Match object; span=(2, 3), match='S'>
<re.Match object; span=(2, 3), match='S'>
None
<re.Match object; span=(0, 1), match='A'>
None
```

In [19]:
```python
## {min and max}

print(re.search("@{1}","APS@@DCSS"))
print(re.search("@{1,3}","APS@DC"))
print(re.search("@{1,3}","APSDC"))
print(re.search("@{1,3}","AP@@DC"))
print(re.search("@{1,2}","A@@@P"))

```

```
<re.Match object; span=(3, 4), match='@'>
<re.Match object; span=(3, 4), match='@'>
None
<re.Match object; span=(2, 4), match='@@'>
<re.Match object; span=(1, 3), match='@@'>
```

In [22]:
```
## [] list of charecters taken as a group

print(re.search("SD","APSSDC"))
print(re.search("[SD]","APSSDC"))
print(re.match("[SD]","APDC"))
print(re.match("[SD]","SAPSSDC"))
print(re.match("[SD]","DPSSDC"))
print(re.match("SD","DPSSDC"))

```

```
<re.Match object; span=(3, 5), match='SD'>
<re.Match object; span=(2, 3), match='S'>
None
<re.Match object; span=(0, 1), match='S'>
<re.Match object; span=(0, 1), match='D'>
None
```

In [33]:
```
## \d, \D, \s, \S

print(re.search("\d","APSS12DC"))
print(re.search("\d\d","APSS1290DC"))
print(re.search("\D","12APSS12DC"))
print(re.findall("\d\d\d","APSS1290DC"))
print(re.search("\D"," 12APSS12DC"))
print(re.search("\s","APSS 12DC"))
print(re.search("\s","APSS12DC"))
print(re.search("\S","      APSS12DC"))
print(re.search("\S","APSS12DC"))
```

```
<re.Match object; span=(4, 5), match='1'>
<re.Match object; span=(4, 6), match='12'>
<re.Match object; span=(2, 3), match='A'>
['129']
<re.Match object; span=(0, 1), match=' '>
<re.Match object; span=(4, 5), match=' '>
None
<re.Match object; span=(6, 7), match='A'>
<re.Match object; span=(0, 1), match='A'>
```

Name Validator

- M Srilalitha
- M Sri Lalitha
- Mulpuru Srilalitha
- Mulpuru Sri Lalitha
- M. Srilalitha
- Srilalitha M

```
#### Phone number validator

* 9775436643
* 09775436643
* 919775436643
* +919775436643
```

```
In [4]:
import re
p1 ="^[+]{0,1}[9][1][6-9][0-9]{9}|^[0]{0,1}[6-9][0-9]{9}"
n = input()
print(re.match(p1,n))
print(re.search(p1,n))

# APSSDC9775436643teifaif
```

```
APSSDC9775436643teifaif
None
None
```

```
#### Email Validator

* srilalitha.m@apssdc.in

* All letters including starting letter must be lowercase alphabet
* Cotains some special charecters (optional)[-_.]
* contains some numbers(optional)
* contains some digits or alphabets after special charecter
* Must contains @
* Collection of alphabets ===> len 4-8
* must contain .
* Collection of alphabets ===> len 2-4
```