In [14]:
```python
f = open("file1.txt","r")
fh = f.read().split("\n")
for i in fh:
    print(i,end= ",")
f.close()
```

```
data1,data2,data3,,
```

In [13]:
```python
f = open("file1.txt")
fh = f.read()
print(fh)
print(len(fh))
f.close()

```

```
data1
data2
data3

18
```

In [8]:
```python
fh
```

Out[8]:
```
['data1', 'data2', 'data3', 'data4data5', 'data5']
```

tasks

- Find the number of letters in your file
- find the words
- find the number of distinct words

In [18]:
```python
f = open("file2.txt","r")
fh = f.read()
c=0
for i in fh:
    if i.isalpha():
        c+=1
print(c)
f.close()

```

```
71
```

In [21]:
```python
f = open("file2.txt","r")
fh = f.read().split()
print(len(fh))
f.close()
```

```
16
```

In [26]:
```python
len(list(set(fh)))
```
Out[26]:
```
14
```

In [29]:
```python
f = open("file2.txt","r")
fh = f.read().split()
l =[]
for i in fh:
    if i not in l:
        l.append(i)
print(len(l))
f.close()
```
```
14
```

In [33]:
```python
# Writing the numbers in to the file from 1 to 50

f = open("file3.txt","w")
for i in range(1,51):
    f.write(str(i)+"\n")
f.close()
```

In [34]:
```python
l = ["Python","Progrmming","2020-21"]
f = open("data.txt","w")
for i in l:
    f.write(i+"\n")
f.close()
```

In [36]:
```python
with open("data.txt","r") as f:
    print(f.read())
```
```
Python
Progrmming
2020-21
```

Type *Markdown* and LaTeX: α^2

In [38]:
```python
36
36**0.333
```
Out[38]:
```
3.2979854306834198
```

In [43]:
```python
import math
math.sqrt(36)
dir(math)
math.log10(2)
```
Out[43]:
```
0.3010299956639812
```

Regular Expressions

- **import regular expression package**

Methods in re

- Search
- match
- findall

re.methodname("pattern","String")

In [49]:
```
import re
print(re.search("SD","APSSDCSDSD"))  #3,4
```

```
<re.Match object; span=(3, 5), match='SD'>
```

In [53]:
```
print(re.match("SD","APSSDC"))
print(re.match("APS","APSSDC"))
```

```
None
<re.Match object; span=(0, 3), match='APS'>
```

In [54]:
```
print(re.findall("SD","APSSDCSDSD"))  #3,4
```

```
['SD', 'SD', 'SD']
```

Symbols

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 [57]:
```python
# "."

print(re.search("..","APSSDC"))
print(re.search("..","AP"))
print(re.search("..","A"))
```
```
<re.Match object; span=(0, 2), match='AP'>
<re.Match object; span=(0, 2), match='AP'>
None
```

In [60]:
```python
# ^
print(re.search("^AP","APSSDC"))
print(re.search("^S","AP"))
print(re.search("^A","A"))

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

In [64]:
```python
# $
print(re.search("DC$","APSSDC"))
print(re.search("DC$","AP"))
print(re.search("3$","A123"))
print(re.match("^AP","APSSDC"))
print(re.match("DC$","APSSDC"))

```
```
<re.Match object; span=(4, 6), match='DC'>
None
<re.Match object; span=(3, 4), match='3'>
<re.Match object; span=(0, 2), match='AP'>
None
```

In [67]:
```python
# *
print(re.search("S*","APSSDC"))
print(re.search("S*","AP"))
print(re.search("S","AP"))
```
```
<re.Match object; span=(0, 0), match=''>
<re.Match object; span=(0, 0), match=''>
None
```

In [68]:
```python
# +
print(re.search("S+","APSSDC"))
print(re.search("S+","AP"))
print(re.search("S*","AP"))
```
```
<re.Match object; span=(2, 4), match='SS'>
None
<re.Match object; span=(0, 0), match=''>
```

In [86]:
```python
# {min,max}

print(re.search("S{1,2}","APSSSSSSSSSSSSDC"))
print(re.search("9{0,1}","AP9"))
print(re.search("S{2,5}","APS"))
```

```
<re.Match object; span=(2, 4), match='SS'>
<re.Match object; span=(0, 0), match=''>
None
```

In [88]:
```python
# []
print(re.search("[ADC]","APSSSSSSSSSSSSDC"))
print(re.search("[DC]","APSSDC"))
print(re.search("[DC]","APSCD"))
```

```
<re.Match object; span=(0, 1), match='A'>
<re.Match object; span=(4, 5), match='D'>
<re.Match object; span=(3, 4), match='C'>
```

In [94]:
```python
# \d,\D,\s,\S
print(re.search("\d","AP123SSSSSSSSSSSSDC"))
print(re.search("\d\d","AP123"))
print(re.search("\D","12APS"))  # otherthan digits
print(re.search("\s","AP SDC"))  # to match spaces
print(re.search("\S","   12AP SDC"))  # to match other than spaces
```

```
<re.Match object; span=(2, 3), match='1'>
<re.Match object; span=(2, 4), match='12'>
<re.Match object; span=(2, 3), match='A'>
<re.Match object; span=(2, 3), match=' '>
<re.Match object; span=(3, 4), match='1'>
```

###### Phone number validator

* starting digit must be 6/7/8/9
* remaining digts 0-9 exactly 9 times

- 8095674321
- 08095674321
- 918095674321
- +918095674321

In [96]:
```python
patt = "[+]{0,1}[9][1][6-9][0-9]{9}|[0]{0,1}[6-9][0-9]{9}"
n = input()
re.match(patt,n)
```

```
+9180956740
```

Name validator

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

Email Validator

- srilalitha.m@apssdc.in (mailto: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

In [ ]:
```

```