

Python - Regex

Palani Karthikeyan
+91 9902084004

Regular expressions

- Regular expressions, also called regex is implemented in pretty much every computer language.
- Regex is used to find a pattern in specific pattern search.
- Globally regex supports two operations
 1. Match operation
 2. Substitute operation
- Regex patterns are generic text,numbers or specialcharacter,space etc.,

- It is widely used in natural language processing, web applications that require validating string input (like email address) and pretty much more data science projects that involve text mining.

import re

- In python, it is implemented in the standard module 're'.
- **>>> import re**
- To get information about regex use help()
- **>>> help(re)**
- Before going to discuss about the regex pattern,1st understand about following metacharacters.
- Here's a complete list of the metacharacters.
- **. ^ \$ * + ? { } [] \ | ()**

. (dot) Matches any single character except newline character.

^ Matches the pattern at the beginning of line.

\$ Matches the pattern at the end of the line.

[...] Matches any single character in brackets.

[^...] Matches any single character not in brackets

***** Matches 0 or more occurrences of preceding expression.

+ Matches 1 or more occurrence of preceding expression.

? Matches 0 or 1 occurrence of preceding expression

{n} Matches exactly n number of occurrences of preceding expression.

{ n, } Matches n or more occurrences of preceding expression.

{ n, m } Matches at least n and at most m occurrences of preceding expression.

pattern1|pattern2 Matches either pattern1 or pattern2

(pattern) Groups regular expressions and remembers matched text.

Special Character Classes

\d Match a digit same as **[0-9]**

\D Match a nondigit same as **[^0-9]**

\s Match a whitespace character same as **[\t\r\n\f]**

\S Match nonwhitespace same as **[^\t\r\n\f]**

\w Match a single word character **[A-Za-z0-9_]**

\W Match a nonword character **[^A-Za-z0-9_]**

\A**Pattern** Match "Pattern" at the start of a string

Pattern\Z Match "Pattern" at the end of a string

re module

- The **re** module provides an interface to the regular expression engine, allowing you to compile REs into objects and then perform matches with them.

Regex Functions

- The **re** module offers a set of functions that allows us to search a string for a match

Function	Description
compile	Regular expressions are compiled into pattern objects
search	Returns a Match object if there is a match anywhere in the string
findall	Returns a list containing all matches
split	Returns a list where the string has been split at each match
sub	Replaces one or many matches with a string
finditer	Find all substrings where the RE matches, and returns them as an iterator.
group	Return the string matched by the RE
start	Return the starting position of the match
end	Return the ending position of the match
span	Return a tuple containing the (start, end) positions of the match

Compiling Regular Expressions

- Regular expressions are compiled into pattern objects.
- Which have methods for various operations such as searching for pattern matches or performing string substitutions.

```
>>> import re
>>> pobj = re.compile('sales')
>>> pobj
<_sre.SRE_Pattern object at 0x...>
```

`re.compile()` also accepts an optional flags argument (re.I) `re.IGNORECASE`

```
>>> pobj=re.compile('sales',re.IGNORECASE)
```

raw string notation

- To use Python's raw string notation for regular expressions; backslashes are not handled.
- In any special way in a string literal prefixed with 'r', so r"\n" is a two-character string containing '\' and 'n'.
- while "\n" is a one-character string containing a newline.
- Regular expressions will often be written in Python code using this raw string notation.
- **search()** searches for the pattern in a given input string.
- returns a particular match object that contains the starting and ending positions of the first occurrence of the pattern.

```
>>> pobj=re.compile("sales")
```

```
>>> pobj.search("ram,sales,pune")
```

```
<re.Match object; span=(4, 9), match='sales'>
```

```
>>>
```

```
>>> pobj.search("ram,prod,bangalore")
```

```
# Pattern is not matched from input string
```

```
>>> # search() return will be None
```

```
...
```

```
>>> pobj.search("ram,prod,bangalore") == None
```

```
True
```

We can write it in another way

Syntax :-

re.search("pattern","inputstring",flag=re.I)

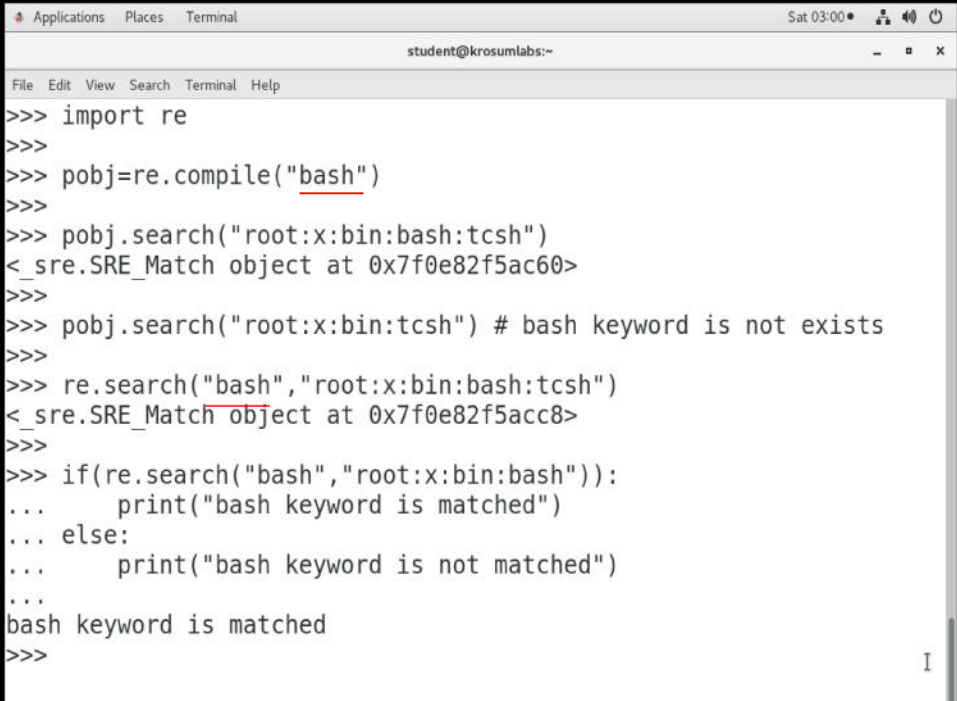
- **>>> re.search("sales","ram,sales,pune")**
- **<re.Match object; span=(4, 9), match='sales'>**
- **>>> re.search("sales","ram,HR,pune")**

using conditional statement

```
>>> if(re.search("sales","ram,sales,pune")):  
...     print("Pattern is matched")  
... else:  
...     print("Pattern is NOT Matched")
```

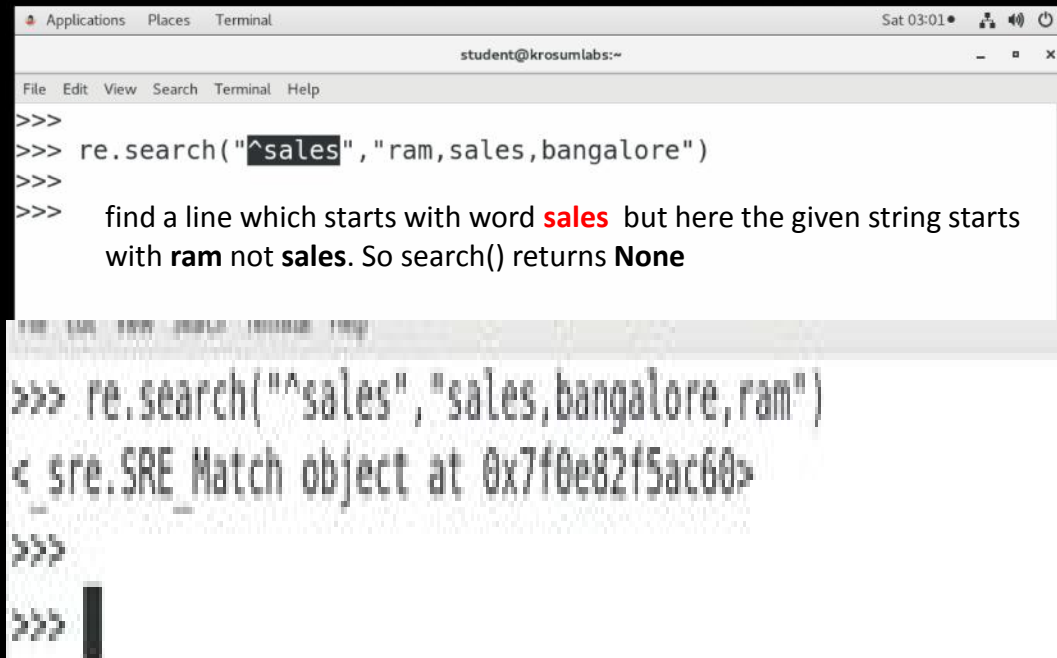
Pattern is matched

Search a **bash** word from input string



```
Applications Places Terminal Sat 03:00 student@krosumlabs:~
File Edit View Search Terminal Help
>>> import re
>>>
>>> pobj=re.compile("bash")
>>>
>>> pobj.search("root:x:bin:bash:tcsh")
<_sre.SRE_Match object at 0x7f0e82f5ac60>
>>>
>>> pobj.search("root:x:bin:tcsh") # bash keyword is not exists
>>>
>>> re.search("bash","root:x:bin:bash:tcsh")
<_sre.SRE_Match object at 0x7f0e82f5acc8>
>>>
>>> if(re.search("bash","root:x:bin:bash")):
...     print("bash keyword is matched")
... else:
...     print("bash keyword is not matched")
...
bash keyword is matched
>>>
```

Find a line starts with a pattern



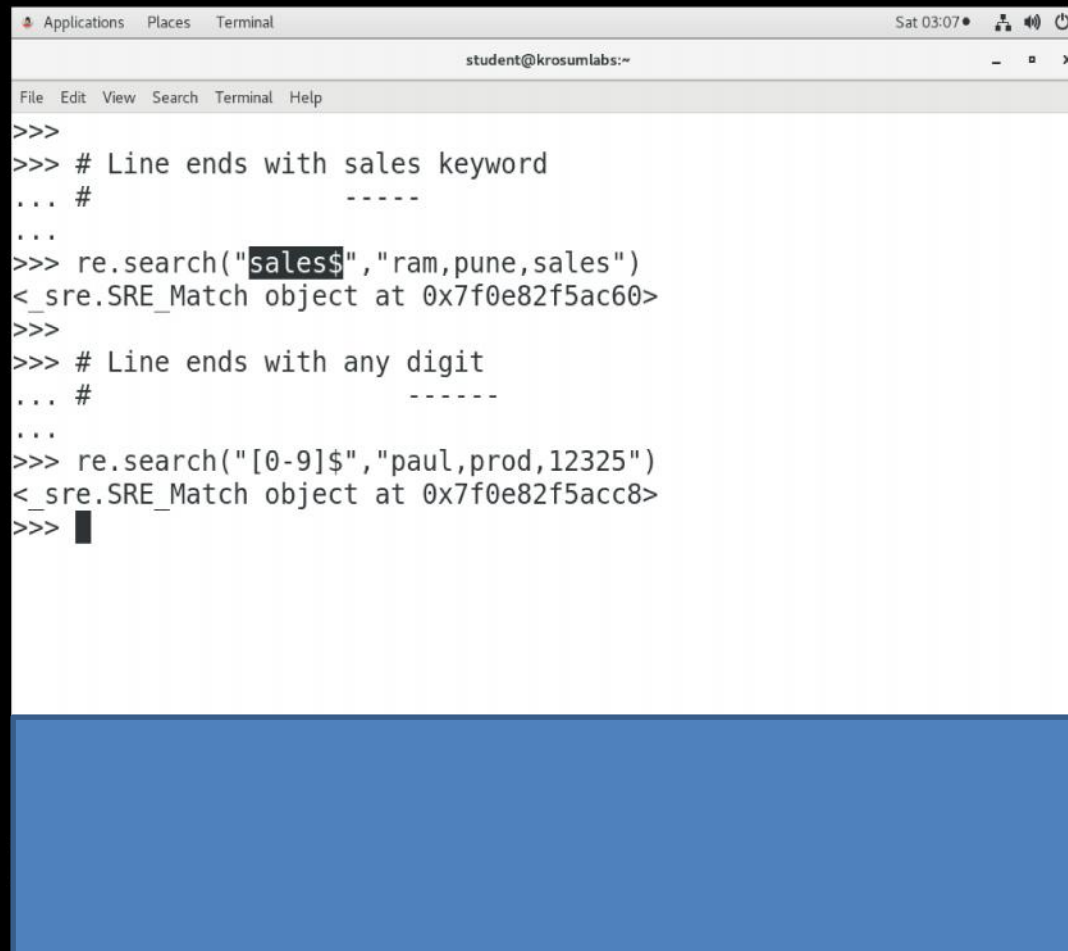
The screenshot shows a terminal window with a menu bar (Applications, Places, Terminal) and a title bar (student@krosumlabs:~). The terminal contains the following text:

```
>>>
>>> re.search("^sales", "ram,sales,bangalore")
>>>
>>> find a line which starts with word sales but here the given string starts
with ram not sales. So search() returns None
```

Below this, there is a blurred section of the terminal showing another code execution:

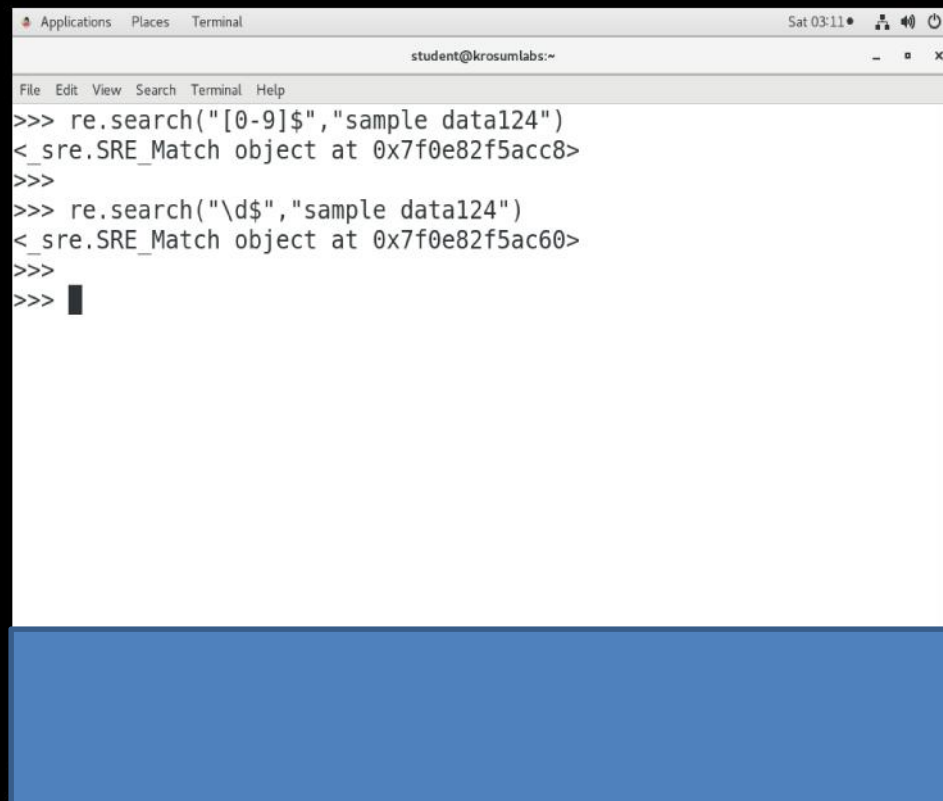
```
>>> re.search("^sales", "sales,bangalore,ram")
<_sre.SRE_Match object at 0x7f0e82f5ac60>
>>>
>>>
```

Find a line that ends with a pattern



```
Applications Places Terminal Sat 03:07 • student@krosumlabs:~
File Edit View Search Terminal Help
>>>
>>> # Line ends with sales keyword
... # -----
...
>>> re.search("sales$","ram,pune,sales")
<_sre.SRE_Match object at 0x7f0e82f5ac60>
>>>
>>> # Line ends with any digit
... # -----
...
>>> re.search("[0-9]$", "paul,prod,12325")
<_sre.SRE_Match object at 0x7f0e82f5acc8>
>>>
```


Find a line that ends with a digit

A terminal window titled 'student@krosumlabs:~' with a menu bar (File, Edit, View, Search, Terminal, Help) and a status bar (Sat 03:11). The terminal shows two successful regex searches. The first search uses '[0-9]\$' and the second uses '\\d\$'. Both return a match object. The input area at the bottom is highlighted with a blue bar.

```
>>> re.search("[0-9]$", "sample data124")
<_sre.SRE_Match object at 0x7f0e82f5acc8>
>>>
>>> re.search("\\d$", "sample data124")
<_sre.SRE_Match object at 0x7f0e82f5ac60>
>>>
>>> █
```


```
Applications Places Terminal Sat 04:16 ● [system icons]
student@krosumlabs:~
File Edit View Search Terminal Help
>>> # Matches pattern line starts with any uppercase chars followed
    # any lower case chars, ends with any digit.
...
>>> re.search("[A-Z][a-z].*[0-9]$", "Text123")
<_sre.SRE_Match object at 0x7f0e82f5acc8>
>>>
>>> re.search("[A-Z][a-z].*[0-9]$", "TEXT123")
>>>                                     #there is no lowercase chars
...
>>> re.search("[A-Z][a-z].*[0-9]$", "abc123")
>>>                                     # Line starts with lowerchars
...
>>> re.search("[A-Z][a-z].*\\d$", "ORACLE\\INUX7")
>>>                                     #Notmatched
```

Applications Places Terminal Sat 04:27

student@krosumlabs:~

File Edit View Search Terminal Help

```
>>> # Linux Commandline
... # -----
... # df -Th|grep "^/dev/sda[1-3]" <== Line starts with any storage
... #                               devices
...
>>> # in python
... # -----
...
>>> import os
>>> for v in os.popen("df -Th").readlines():
...     v=v.strip()
...     if(re.search("^/dev/sda[1-3]",v)):
...         print(v)
...
/dev/sda2      xfs      47G   3.8G   43G   9% /
/dev/sda3      xfs      19G  474M   19G   3% /home
/dev/sda1      xfs      485M  187M  298M  39% /boot
>>>
```



```
Applications Places Terminal Sat 04:30
student@krosumlabs:~
File Edit View Search Terminal Help
>>> # Filter list of process name ends with character 'd'
... # -----
... # ps -e|grep d$ <== Linux Commandline
...
>>> for v in os.popen("ps -e").readlines():
...     v=v.strip()
...     if(re.search("d$",v)):
...         print(v)
```

```
Applications Places Terminal Sat 04:30
student@krosumlabs:~
File Edit View Search Terminal Help
>>> # Filter list of process name ends with character 'd'
... # -----
... # ps -e|grep d$ <== Linux Commandline
...
>>> for v in os.popen("ps -e").readlines():
...     v=v.strip()
...     if(re.search("d$",v)):
...         print(v)
...
1 ?      00:00:06 systemd
2 ?      00:00:00 kthreadd
7 ?      00:00:00 rcu_sched
17 ?     00:00:00 khungtaskd
19 ?     00:00:00 ksmd
20 ?     00:00:00 khugepaged
22 ?     00:00:00 kintegrityd
24 ?     00:00:00 kblockd
25 ?     00:00:00 md
42 ?     00:00:00 kthrotld
45 ?     00:00:00 kmpath_rdacd
46 ?     00:00:00 kpsmoused
103 ?    00:00:00 kauditd
```

```
Applications Places Terminal Sat 04:32
student@krosumlabs:~
File Edit View Search Terminal Help
>>>
>>>
>>>
>>> daemon_process=os.popen("ps -e|grep d$").read()
>>>
>>> print(daemon_process)
1 ?      00:00:06 systemd
2 ?      00:00:00 kthreadd
7 ?      00:00:00 rcu_sched
17 ?     00:00:00 khungtaskd
19 ?     00:00:00 ksmd
20 ?     00:00:00 khugepaged
22 ?     00:00:00 kintegrityd
24 ?     00:00:00 kblockd
25 ?     00:00:00 md
42 ?     00:00:00 kthrotld
45 ?     00:00:00 kmpath_rdacd
46 ?     00:00:00 kpsmoused
103 ?    00:00:00 kauditd
381 ?    00:00:00 rpciod
382 ?    00:00:00 lvmetad
393 ?    00:00:00 systemd-udevd
```

```
Applications Places Terminal Sat 04:38 student@krosumlabs:~  
File Edit View Search Terminal Help  
data1  
data2  
data3  
data4  
data5  
data6  
data7  
data8  
0  
>>> # Filters the line which ends with space(char)  
... # -----  
... # Linux commandline --> cat IP.txt|grep "[[:space:]]$"   
... #  
...  
>>> with open("IP.txt") as fobj:  
...     for v in fobj.readlines():  
...         v=v.strip()  
...         if(re.search("\s$",v)):  
...             print(v)  
...  
>>>  
>>>
```

```
Applications Places Terminal Sat 04:55 student@krosumlabs:~
File Edit View Search Terminal Help
>>> # write a python program:
... # -----
... # read a emp ID and emp name from STDIN
... # emp ID should begin with uppercase chars between A to E,
... # followed by any 3 digits.
... #
... # emp name should begin with any uppercase char followed by
... # any number of lowercase chars.
... # Example:-
... # -----
... # ArunKumar ARUN arun V.Arun Arun<space> Arun5 - invalidformat
... #
... # Arunkumar Vishnuseravana Johnpaul Arun Varun -Validformat
...
>>>
```

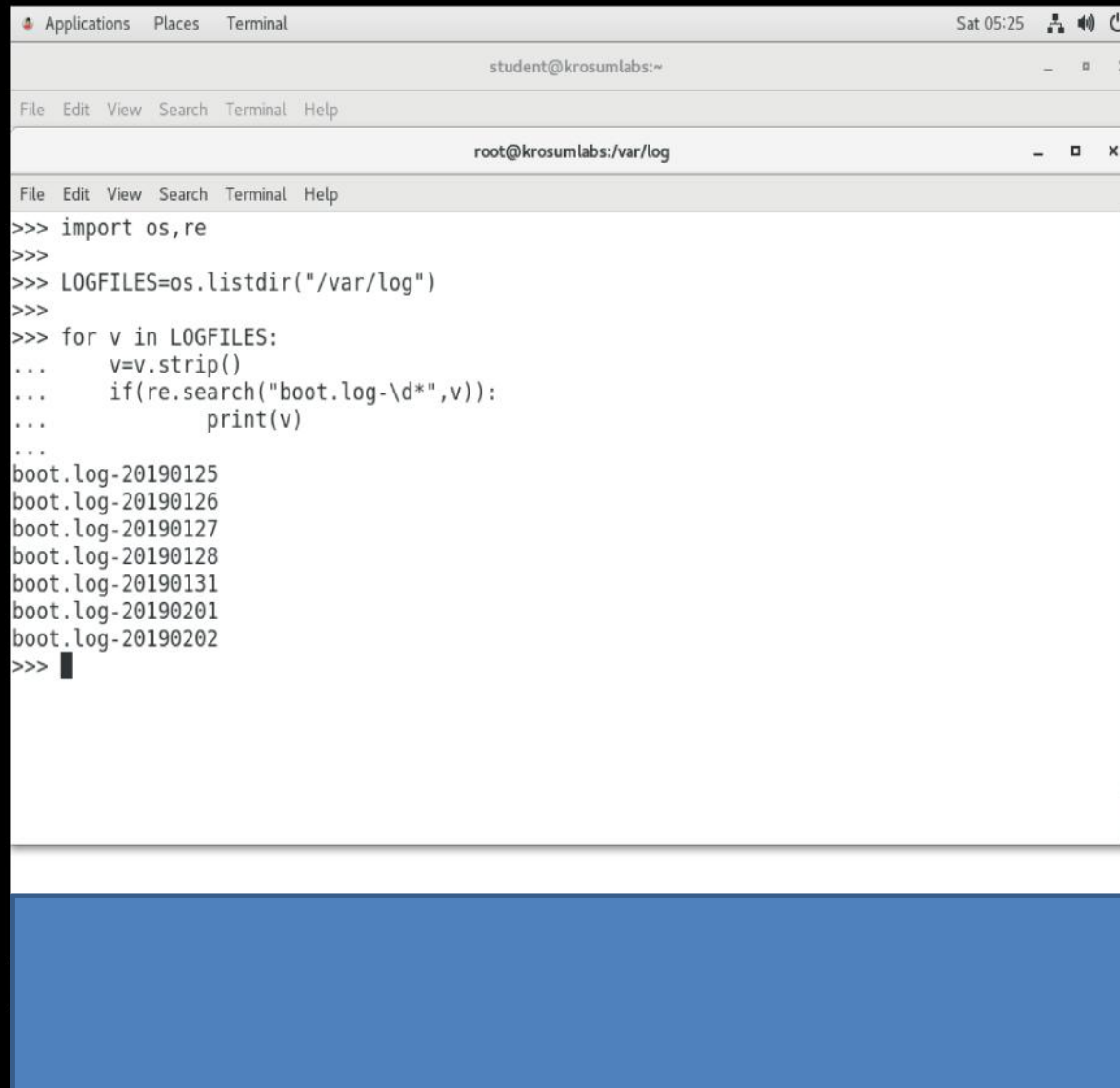


```
Applications Places Terminal Sat 05:05
student@krosumlabs:~
File Edit View Search Terminal Help
[student@krosumlabs ~]$ cat -n e1.py
 1 import re,sys
 2
 3 eid=raw_input("Enter your emp id:")
 4
 5 if(not re.search("^[A-E]\d{3}$",eid)):
 6     print("Invalid format")
 7     sys.exit(1)
 8
 9
10 ename=raw_input("Enter emp name:")
11
12 if(not re.search("^[A-Z][a-z]+$",ename)):
13     print("Invalid format")
14     sys.exit(1)
15
16 print("Emp name:{}\tEmp ID:{}".format(ename,eid))
[student@krosumlabs ~]$
```

```
Applications Places Terminal Sat 05:05
student@krosumlabs:~
File Edit View Search Terminal Help
[student@krosumlabs ~]$ python e1.py
Enter your emp id:E123
Enter emp name:Ashokkumar
Emp name:Ashokkumar Emp ID:E123
[student@krosumlabs ~]$
[student@krosumlabs ~]$ python e1.py
Enter your emp id:E1234
Invalid format
[student@krosumlabs ~]$ python e1.py
Enter your emp id:F123
Invalid format
[student@krosumlabs ~]$ python e1.py
Enter your emp id:E345
Enter emp name:KARTHIK
Invalid format
[student@krosumlabs ~]$ python e1.py
Enter your emp id:E456
Enter emp name:Xerox.paul
Invalid format
[student@krosumlabs ~]$
```

```
Applications Places Terminal Sat 05:17
student@krosumlabs:~
File Edit View Search Terminal Help
root@krosumlabs:/var/log
File Edit View Search Terminal Help
[root@krosumlabs log]# ls
anaconda      cron-20190125  messages      spooler
audit         cron-20190127  messages-20181224  spooler-20181224
boot.log      cups          messages-20190101  spooler-20190101
boot.log-20190125  dmesg        messages-20190125  spooler-20190125
boot.log-20190126  dmesg.old    messages-20190127  spooler-20190127
boot.log-20190127  firewallld   pluto            tallylog
boot.log-20190128  gdm          ppp              tuned
boot.log-20190131  glusterfs    qemu-ga          wpa_supplicant.log
boot.log-20190201  grubby_prune_debug  sa              wtmp
boot.log-20190202  lastlog      samba            Xorg.0.log
btmtp         libvirt      secure           Xorg.0.log.old
btmtp-20190201  maillog      secure-20181224  Xorg.9.log
chrony        maillog-20181224  secure-20190101  yum.log
cron          maillog-20190101  secure-20190125  yum.log-20190101
cron-20181224  maillog-20190125  secure-20190127
cron-20190101  maillog-20190127  speech-dispatcher
[root@krosumlabs log]# ls boot.log-[0-9]*
boot.log-20190125  boot.log-20190127  boot.log-20190131  boot.log-20190202
boot.log-20190126  boot.log-20190128  boot.log-20190201
[root@krosumlabs log]#
```

Linux ls /var/boot/boot.log-[0-9]*



The screenshot shows a Linux terminal window with a menu bar (Applications, Places, Terminal) and a status bar (Sat 05:25). The terminal title is 'student@krosumlabs:~'. Below it is a sub-window titled 'root@krosumlabs:/var/log' with its own menu bar (File, Edit, View, Search, Terminal, Help). The terminal displays a Python script that lists files in /var/log and filters for those matching the pattern 'boot.log-[0-9]*'. The output shows seven files: boot.log-20190125, boot.log-20190126, boot.log-20190127, boot.log-20190128, boot.log-20190131, boot.log-20190201, and boot.log-20190202.

```
>>> import os,re
>>>
>>> LOGFILES=os.listdir("/var/log")
>>>
>>> for v in LOGFILES:
...     v=v.strip()
...     if(re.search("boot.log-\d*",v)):
...         print(v)
...
boot.log-20190125
boot.log-20190126
boot.log-20190127
boot.log-20190128
boot.log-20190131
boot.log-20190201
boot.log-20190202
>>>
```

Extract **year,month** and **date** from given url string.

```
>>> import re
>>>
>>> url="https://www.krosum.com/news/latest-beckhams/2018/02/09/ode
ll-fame-author"
>>>
>>> # Extract year,month and date from an URL
... # .....
...
>>> re.findall("\d{4}/\d{2}/\d{2}",url)
['2018/02/09']
>>>
>>>
```

re.split()

```
>>>
>>> import re
>>>
>>> "root:x:bin:bash:tcsh".split(":")
['root', 'x', 'bin', 'bash', 'tcsh']
>>>
>>> "root-x:bin,bash-tcsh".split(":")
['root-x', 'bin,bash-tcsh']
>>>
>>> "root-x:bin,bash-tcsh".split("-")
['root', 'x:bin,bash-tcsh']
>>>
>>> "root-x:bin,bash-tcsh".split("~")
['root-x:bin,bash', 'tcsh']
>>>
>>> # re.split("[:|--]", "root-x:bin,bash-tcsh")
... # -----
... #         character class
...
>>>
>>> re.split("[:|--]", "root-x:bin,bash-tcsh")
['root', 'x', 'bin', 'bash', 'tcsh']
>>>
>>>
```

Split the text around 1 or more space chars

```
>>> Given="""
... This is sample
... text line
... multiple line text
... data from line by line"""
>>>
>>> # re.split("regex",Inputstring)  <== Syntax
...
>>> # split the text around 1 or more space characters
... #           .. .....
...

>>>
>>> re.split("\s+",Given)
['', 'This', 'is', 'sample', 'text', 'line', 'multiple', 'line', 'text', 'dat
a', 'from', 'line', 'by', 'line']
>>> █
```

re.match()

```
>>> re.search("^root", "root:x:bin:bash")
<re.Match object; span=(0, 4), match='root'>
>>>
>>>
>>> help(re.match)
Help on function match in module re:

match(pattern, string, flags=0)
    Try to apply the pattern at the start of the string, returning
    a Match object, or None if no match was found.

>>>
>>> # re.match(pattern,inputstring)
... # -----
... # |__pattern matches at the beginning of the inputstring
... #
... # re.search("^pattern",input) same as re.match("pattern",input)
... # -----
... #
... #
>>> re.match("root", "root:x:bin:bash")
<re.Match object; span=(0, 4), match='root'>
>>>
```


group() vs groups()



The screenshot shows a terminal window titled "OL7.4 [Running] - Oracle VM VirtualBox". The terminal interface includes a menu bar with "File", "Machine", "View", "Input", "Devices", and "Help". Below the menu bar is a toolbar with icons for Applications, Places, and Terminal. The terminal window has a title bar with "student@krosumlabs:~" and standard window controls. The terminal content shows a Python interactive session with the following commands and output:

```
>>> import re
>>>
>>> re.search("(\\w\\d+).*(\\w\\d+)", "Sample log file size:54KB and 65MB")
<_sre.SRE_Match object at 0x7f5f5e24f2d8>
>>>
>>> obj=re.search("(\\w\\d+).*(\\w\\d+)", "Sample log file size:54KB and 65MB")
>>>
>>> obj.group()
'54KB and 65'
>>>
>>> obj.groups()
('54', '65')
>>>
>>> █
```

re.sub()

re.sub() - replace old pattern with a new pattern in the given string.

```
>>> help(re.sub)
```

Help on function sub in module re:

sub(pattern, repl, string, count=0, flags=0)

Return the string obtained by replacing the leftmost non-overlapping occurrences of the pattern in string by the replacement repl. repl can be either a string or a callable; if a string, backslash escapes in it are processed. If it is a callable, it's passed the Match object and must return a replacement string to be used.

>>>

>>> re.sub("sales","PRODUCTION","ram,sales,pune")

'ram,PRODUCTION,pune'

>>>

>>> re.sub("sales","PRODUCTION","ram,SALES,pune")

'ram,SALES,pune'

>>> # old pattern is not matched from inputstring,so there is no replacement

... # ---

...

>>> re.sub("sales","PRODUCTION","ram,SALES,pune",0,re.I) # ignore case

'ram,PRODUCTION,pune'

>>>

>>>

substitute all the digits with *

```
>>> re.sub("\d+", "*", "DATE:6th-March-2013 file INFO details:459KB")
```

```
'DATE:*th-March-* file INFO details:*KB'
```

```
>>> # delete all the digits[0-9]
```

```
... # -----
```

```
>>> re.sub("\d+", "", "DATE:6th-March-2013 file INFO details:459KB")
```

```
'DATE:th-March- file INFO details:KB'
```

```
>>> re.sub("sales","PROD","data1,sales,sales,sales,SALES,Sales")  
'data1,PROD,PROD,PROD,SALES,Sales'
```

ignore the case



```
>>> re.sub("sales","PROD","data1,sales,sales,sales,SALES,Sales",0,re.I) # ignore case  
'data1,PROD,PROD,PROD,PROD,PROD'
```

```
>>> re.sub("sales","PROD","data1,sales,sales,sales,SALES,Sales",1,re.I)  
'data1,PROD,sales,sales,SALES,Sales'
```

replace 1st matched occurrence



```
[student@krosumlabs ~]$ cat -n IP # <== this is sample inputfile
```

```
1 root:x:0:0:root:/root:/bin/bash
2 bin:x:1:1:bin:/bin:/sbin/bash
3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
4 adm:x:3:4:adm:/var/adm:/sbin/nologin
5 bash:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

```
[student@krosumlabs ~]$
```

```
[student@krosumlabs ~]$ sed 's/bash/KSH/' IP #<== replace bash to ksh
```

```
root:x:0:0:root:/root:/bin/KSH
```

```
bin:x:1:1:bin:/bin:/sbin/KSH
```

```
daemon:x:2:2:daemon:/sbin:/sbin/nologin
```

```
adm:x:3:4:adm:/var/adm:/sbin/nologin
```

```
KSH:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

```
[student@krosumlabs ~]$ cat -n p1
```

```
1 import re
2
3 with open("IP") as fobj:
4     for v in fobj.readlines():
5         v=v.strip()
6         s=re.sub("bash","KSH",v)
7         print(s)
```

```
[student@krosumlabs ~]$
```

OL7.4 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Applications Places Terminal Sat 07:02

student@krosumlabs:~

File Edit View Search Terminal Help

```
3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
4 adm:x:3:4:adm:/var/adm:/sbin/nologin
5 bash:x:4:7:lp:/var/spool/lpd:/sbin/nologin
[student@krosumlabs ~]$
[student@krosumlabs ~]$ sed 's/bash/KSH/' IP #<== replace bash to ksh
root:x:0:0:root:/root:/bin/KSH
bin:x:1:1:bin:/bin:/sbin/KSH
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
KSH:x:4:7:lp:/var/spool/lpd:/sbin/nologin
[student@krosumlabs ~]$ cat -n p1
1 import re
2
3 with open("IP") as fobj:
4     for v in fobj.readlines():
5         v=v.strip()
6         s=re.sub("bash", "KSH",v)
7         print(s)
[student@krosumlabs ~]$ python p1
root:x:0:0:root:/root:/bin/KSH
bin:x:1:1:bin:/bin:/sbin/KSH
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
KSH:x:4:7:lp:/var/spool/lpd:/sbin/nologin
[student@krosumlabs ~]$
```

Input String

New pattern

Old pattern

student@krosumlabs:~

File Edit View Search Terminal Help

```
[student@krosumlabs ~]$ sed 's/bash/KSH/' IP >NEWFILE # writing to new file
```

```
[student@krosumlabs ~]$
```

```
[student@krosumlabs ~]$ cat -n p2
```

```
1 import re
```

```
2
```

```
3 with open("IP") as fobj:
```

```
4     with open("NEWFILE", "w") as wobj:
```

```
5         for v in fobj.readlines():
```

```
6             v=v.strip()
```

```
7             s=re.sub("bash", "KSH", v)
```

```
8             wobj.write(s+"\n") # writing to new file
```

```
[student@krosumlabs ~]$
```

```
[student@krosumlabs ~]$ python p2
```

```
[student@krosumlabs ~]$ cat -n NEWFILE
```

```
1 root:x:0:0:root:/root:/bin/KSH
```

```
2 bin:x:1:1:bin:/bin:/sbin/KSH
```

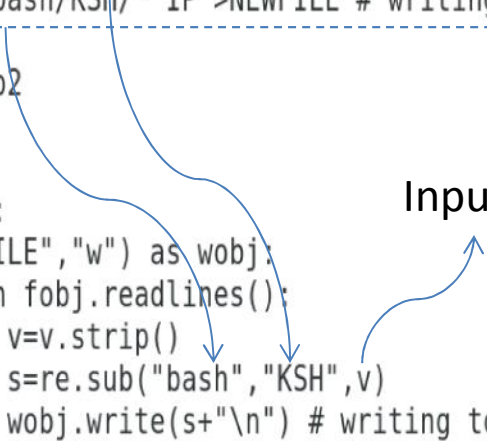
```
3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
```

```
4 adm:x:3:4:adm:/var/adm:/sbin/nologin
```

```
5 KSH:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

```
[student@krosumlabs ~]$
```

Input string



OL7.4 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Applications Places Terminal Sat 07:07

student@krosumlabs:~

```
[student@krosumlabs ~]$ cat -n IP
1 root:x:0:0:root:/root:/bin/bash
2 bin:x:1:1:bin:/bin:/sbin/bash
3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
4 adm:x:3:4:adm:/var/adm:/sbin/nologin
5 bash:x:4:7:lp:/var/spool/lpd:/sbin/nologin
[student@krosumlabs ~]$
[student@krosumlabs ~]$ sed 's/bash/KSH/' IP
root:x:0:0:root:/root:/bin/KSH
bin:x:1:1:bin:/bin:/sbin/KSH
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
KSH:x:4:7:lp:/var/spool/lpd:/sbin/nologin
[student@krosumlabs ~]$
[student@krosumlabs ~]$ sed -n 's/bash/KSH/p' IP # display replaced lines to
# screen.

root:x:0:0:root:/root:/bin/KSH
bin:x:1:1:bin:/bin:/sbin/KSH
KSH:x:4:7:lp:/var/spool/lpd:/sbin/nologin
[student@krosumlabs ~]$
```

```
[student@krosumlabs ~]$ cat -n p3
```

```
1 import re
```

```
2
```

```
3 with open("IP") as fobj:
```

```
4     for v in fobj.readlines():
```

```
5         v=v.strip()
```

```
6         if re.search("bash",v):
```

```
7             s=re.sub("bash","KSH",v)
```

```
8             print(s)
```

```
[student@krosumlabs ~]$ python p3
```

```
root:x:0:0:root:/root:/bin/KSH
```

```
bin:x:1:1:bin:/bin:/sbin/KSH
```

```
KSH:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

```
[student@krosumlabs ~]$
```

Search **bash** keyword from input

Replace bash with **KSH**

Print only the replaced line to the screen

OL7.4 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Applications Places Terminal Sat 07:12

student@krosumlabs:~

```
>>> import re
>>>
>>> # delete all the space from given string
... # -----
...
>>> import os
>>>
>>> var=os.popen("uptime").read()
>>> var
' 07:10:26 up 12:04,  2 users,  load average: 0.17, 0.19, 0.31\n'
>>>
>>> re.sub("\s+", "", var)
'07:10:26up12:04,2users,loadaverage:0.17,0.19,0.31'
>>>
>>>
>>> re.sub("\s+", "", os.popen("ls -l /etc/passwd").read())
'-rw-r--r--.1rootroot2074Jan2521:23/etc/passwd'
>>>
>>> re.sub("\s+", "\t", os.popen("ls -l /etc/passwd").read())
'-rw-r--r--.\t1\troot\troot\t2074\tJan\t25\t21:23\t/etc/passwd\t'
>>>
>>> print(re.sub("\s+", "\t", os.popen("ls -l /etc/passwd").read()))
-rw-r--r--.    1      root    root    2074    Jan    25    21:23    /etc/
passwd
>>>
```

```
>>> os.system("ls -l /etc/passwd")
-rw-r--r--. 1 root root 2074 Jan 25 21:23 /etc/passwd
0
>>> # display single line into multiple lines
... # ----- //based on space split \n
...
>>> re.sub("\s+", "\n", os.popen("ls -l /etc/passwd").read())
'-rw-r--r--.\n1\nroot\nroot\n2074\nJan\n25\n21:23\n/etc/passwd\n'
>>>
>>> print(re.sub("\s+", "\n", os.popen("ls -l /etc/passwd").read()))
-rw-r--r--.
1
root
root
2074
Jan
25
21:23
/etc/passwd
>>>
```

```
>>> # Using re.sub() delete operation
... # -----
...
>>> re.sub("#","", "uptime # display cpu loadbalance") # remove # character
'uptime display cpu loadbalance'
>>>
>>> re.sub("#.*","", "uptime # display cpu loadbalance") # remove # followed by
                                                                # all chars(comment)
'uptime '
>>>
>>> # remove last character
... # -----
...
>>>
>>> var='uptime '
>>>
>>> re.sub(".$","", var)
'uptime'
>>>
>>> # delete bash word from given string
... # -----
...
>>> re.sub("bash.", "", "root:x:bash:/usr/bin/tcsh")
'root:x:/usr/bin/tcsh'
>>> █
```

```
>>> process="""
... python
... sh
... oracle
... bash"""
>>>
>>> re.findall("^\\w",process)
[]
>>>
>>> re.findall("^\\w",process,re.MULTILINE)
['p', 's', 'o', 'b']
```

MULTILINE the code will check each line in the string for object

