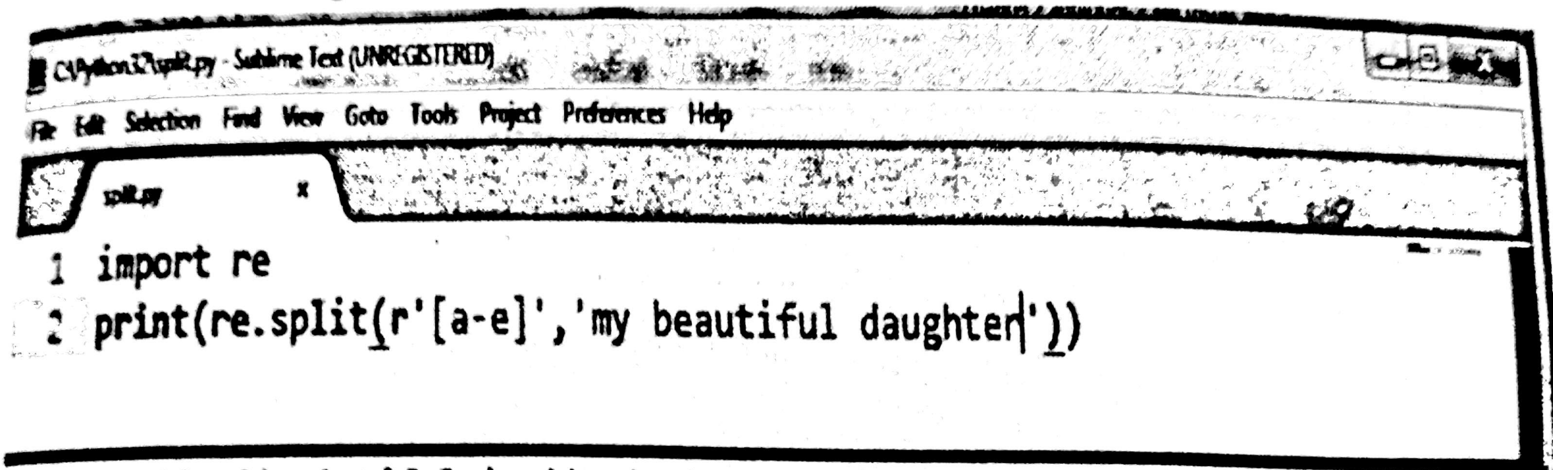
Pattern	Description
^	Matches expression at the beginning of string.(e.g. "^.at" would match cat, hat, sat if present at the start of the string)
$	Matches expression at the end of string.(e.g. ".at$" would match sat,cat ,hat if present at the end)
.	Matches any single character except newline. Using m option allows it to match newline as well.(E.g "at" would match rat, cat etc.)
[...]	Matches any single character in brackets.(e.g. "[ch]at" would match hat, cat but not sat or mat)
[^...]	Matches any single character which is not in the bracket.(e.g. "[^c]at" would match mat, hat but not cat.
()	It contains the subexpression.
*	It matches the previous element 0 or more times.(e.g. "c.*" would match any word begin with c like cat, crow, crowd etc)
re*	Matches 0 or more occurrences of preceding expression.
re+	Matches 1 or more occurrence of preceding expression.
re?	Matches 0 or 1 occurrence of preceding expression.
re{ n}	Matches exactly n number of occurrences of preceding expression.
re{ n,}	Matches n or more occurrences of preceding expression.
re{ n, m}	Matches at least n and at most m occurrences of preceding expression.
a| b	Matches either a or b.
(re)	Groups regular expressions and remembers matched text.
(?imx)	Temporarily toggles on i, m, or x options within a regular expression. If in parentheses, only that area is affected.
(?-imx)	Temporarily toggles off i, m, or x options within a regular expression. If in parentheses, only that area is affected.
(?: re)	Groups regular expressions without remembering matched text.

Pattern	Description
(?imx: re)	Temporarily toggles on i, m, or x options within parentheses.
(?-imx: re)	Temporarily toggles off i, m, or x options within parentheses.
(?#...)	Comment.
(?= re)	Specifies position using a pattern. Doesn't have a range.
(?! re)	Specifies position using pattern negation. Doesn't have a range.
(?> re)	Matches independent pattern without backtracking.
\w	Matches word characters.
\W	Matches nonword characters.
\s	Matches whitespace. Equivalent to [\t\n\r\f].
\S	Matches nonwhitespace.
\d	Matches digits. Equivalent to [0-9].
\D	Matches nondigits.
\A	Matches beginning of string.
\Z	Matches end of string. If a newline exists, it matches just before newline.
\z	Matches end of string.
\G	Matches point where last match finished.
\b	Matches word boundaries when outside brackets. Matches backspace (0x08) when inside brackets.
\B	Matches nonword boundaries.
\n, \t, \r etc.	Matches newlines, carriage returns, return tabs, etc.
\1...\9	Matches nth grouped subexpression.
\10	Matches nth grouped subexpression if it matched already. Otherwise refers to the octal representation of a character code.

Now we will have a look of few examples of the regular expression patterns.

Example 2 : In the following example it eliminates the letters from a to e from the given string.

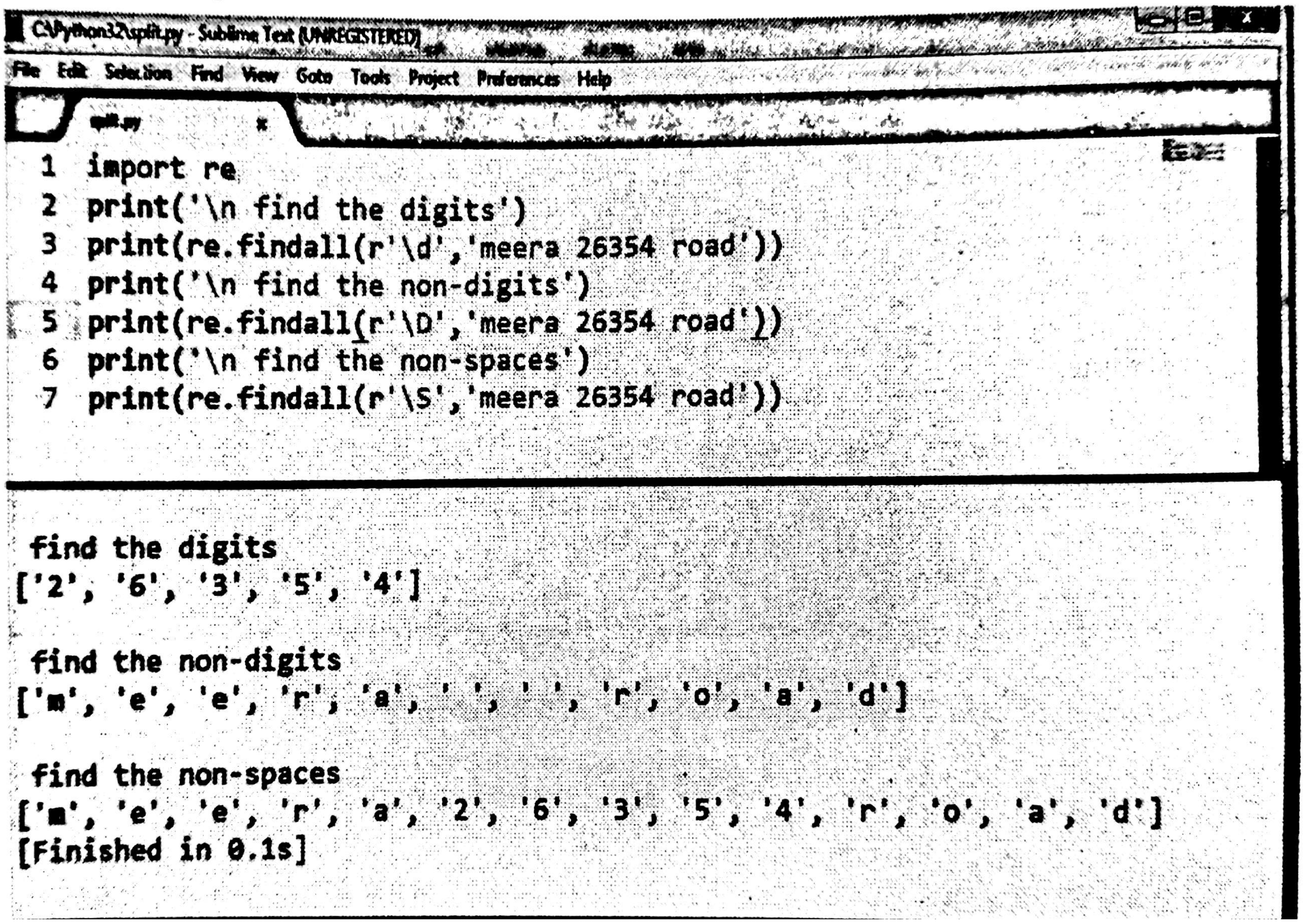

C:\Python32\split.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
split.py

```
import re
print(re.split(r'[a-e]','my beautiful daughter'))
```

```
['my ', '', '', 'utiful ', '', 'ught', 'r']
[Finished in 0.1s]
```

Scanned by CamScanner

Example 4 : In the following example it find out the digits, non digits ans the spaces.


C:\Python32\split.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
split.py

```
import re
print('\n find the digits')
print(re.findall(r'\d','meera 26354 road'))
print('\n find the non-digits')
print(re.findall(r'\D','meera 26354 road'))
print('\n find the non-spaces')
print(re.findall(r'\S','meera 26354 road'))
```

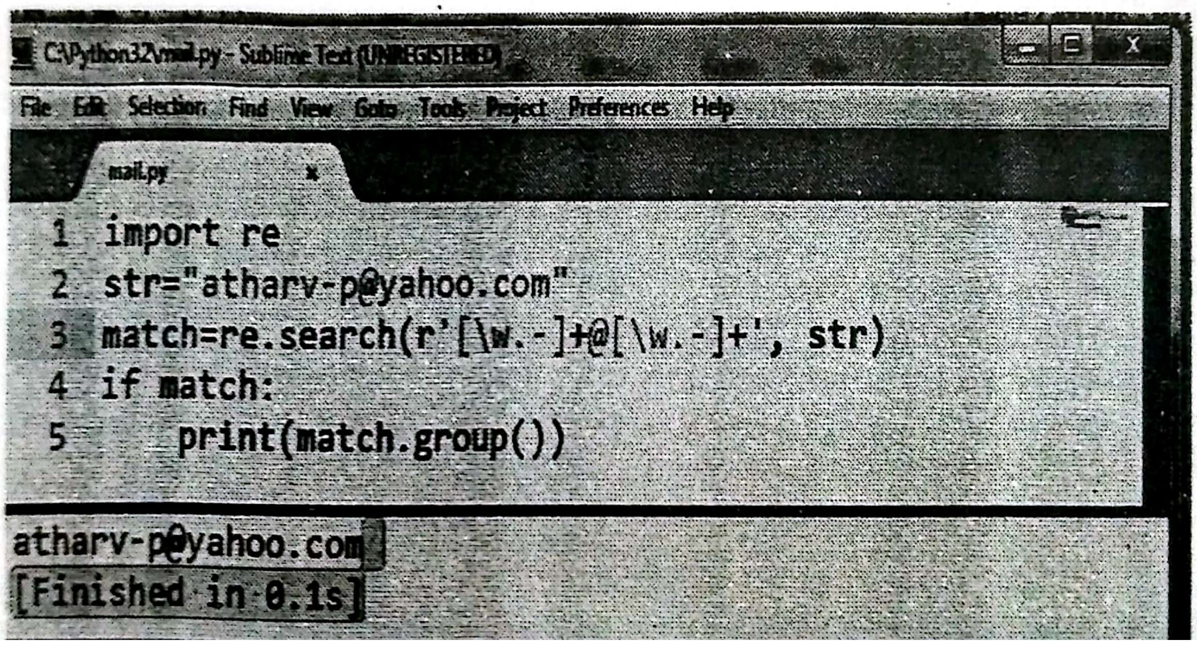
```

 find the digits
['2', '6', '3', '5', '4']

 find the non-digits
['m', 'e', 'e', 'r', 'a', ' ', ' ', 'r', 'o', 'a', 'd']

 find the non-spaces
['m', 'e', 'e', 'r', 'a', '2', '6', '3', '5', '4', 'r', 'o', 'a', 'd']
[Finished in 0.1s]
```

Example 6 : In the following example the code is written to get the mail id.


C:\Python32\mail.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
mail.py

```
import re
str="atharv-p@yahoo.com"
match=re.search(r'[\w.-]+@[\w.-]+', str)
if match:
    print(match.group())
```

```
atharv-p@yahoo.com
[Finished in 0.1s]
```

Flag	Meaning
ASCII, A	The ASCII creates various escapes such as \w, \s, \b and \d . it matches only an ASCII characters with the respective property.
DOTALL, S	The dot(.) match any character, it also includes newlines
IGNORECASE, I	This flag is used to perform case-insensitive matches
LOCALE, L	This flag is used to perform the locale-aware match
MULTILINE, M	It performs the Multi-line matching. It also affects the ^ and $
VERBOSE, X (for 'extended')	The VERBOSE enables the verbose Res. This RE's are organized more cleanly and understandably

Modifier	Description
re.I	The re.I is used to for case-insensitive matching.
re.M	This modifier uses $ to match the end of a line and ^ to match start of any line.
re.U	The letters are interpreted as per the Unicode character set. It affects the behavior of \w, \W, \b, \B.
re.L	The words are interpreted as per current locale. The interpretation is affected by the alphabetic group (\w and \W) and word boundary behavior (\b and \B).
re.S	It makes a period or dot to match any character, including a newline.
re.X	The whitespaces (except inside a set [] or when escaped by a backslash)are ignored by this modifier. It treats the # as comment marker.