MATCH WORDS WITH A PARTICULAR PATTERN

Example

```
import re

txt = "sat,hat,mat,pat"
x = re.findall("[shmp]at", txt)

for i in fp:
      print(i)
```

output

sat

hat

mat

pat

MATCH WORDS WITH A PARTICULAR PATTERN WITH SERIES

Example

```
import re

txt = "sat,hat,mat,pat"
x = re.findall("[h-z]at", txt)

for i in fp:
      print(i)

this pattern will gives a series of words that between that start
with h and z
```

output

hat

mat

pat

sat

EXCLUDING THE DATA USING CARET SIGN

Example

```
import re

txt = "sat,hat,mat,pat"
x = re.findall("^[h-z]at", txt)

for i in fp:
      print(i)
```

carrot symbol will helps to exclude the pattern given in the bracket

r -RAW STRING

Example

```
import re

txt = "its \\kochi"
x = re.search(r"\\kochi", txt)


print(x)
```

In normal python program ,for double backslashes python will automatically remove one backslash.so we wont get the correct data to solve this problem we can use "r"(raw string). Raw strings are used so that backslashes do not have to be escaped. ... The group() method returns strings of the matched text.

\d- MATCHES DIGITS

Example

```
import re

txt = "1234"

print(“data:”,len(re.findall(“/d”,data)))
```

output

data:[4]

\D- MATCHES NON DIGITS

Example

```
import re

txt = "1234"

print(“data:”,len(re.findall(“/D”,data)))
```

output

data:[0]

\s- MATCHES space

Example

```
import re

txt = "THE LINE 1"
```

```
x= re.search(r”\s”,text)

print(“space position”,x.start())

start()  -gives starting index position
```

output

```
space position:3
```

$- MATCHES END OF LINE

Example

```
import re

txt = "THE LINE 1"

x= re.findall(r”\d$”,text)

print(x)
```

output

```
[‘1’]
```

^ - MATCHES BEGINNING OF LINE

Example

```
import re

txt = "2 THE LINE 1"

x= re.findall(r”^\d”,text)

print(x)
```

Output

['2']

DIFFERENCE BETWEEN ^ AND[^...]

^

Matches beginning of line.

[^...]

Matches any single character not in brackets

GROUP()

groups() method. This method returns a tuple containing all the subgroups of the match, from 1 up to however many groups are in the pattern. The default argument is used for groups that did not participate in the match;

Example

```
import re

data='MY PHONE NUMBER IS 888-555-1234'

pattern=r'\d\d\d-\d\d\d-\d\d\d\d'

phn=re.search(pattern,data)

p=phn.groups()


print(p)
```

output

```
888-555-1234

pattern=r'\d\d\d-\d\d\d-\d\d\d\d' This line of code can be write as:

pattern=r'\d{3}-\d{3}-\d{4}'
```

subgroup()

```
import re

data=’MY PHONE NUMBER IS 888-555-1234’

pattern=r’(\d{3})-(\(d{3})-(\d{4})’

phn=re.search(pattern,data)

print(phn.groups(1))
print(phn.groups(2))
```

Sr.No.	Match Object Methods & Description
1	group(num=0) This method returns entire match (or specific subgroup num)
2	groups() This method returns all matching subgroups in a tuple (empty if there weren't any)

PROGRAM TO FIND PHONE NUMBER

```
import re

mobile=input("ENTER THE MOBILE NUMBER:")

if (re.search(pattern,mobile)):

     print(f"{pattern} is valid")

else:

     print(f"{pattern} is not valid")
```

EXTRA EXAMPLE PROGRAMS

findall()

```
import re
x= re.findall('[0-9]','ach5fh7bh2jg4y')

print(x)
```

```
split()
Import re

data="all indians are brother and sister"
x= re.split('i','data')

print(x)
```

sub()

Syntax

```
re.sub(pattern, repl, string, max=0)
```

This method replaces all occurrences of the RE *pattern* in *string* with *repl*, substituting all occurrences unless *max* provided. This method returns modified string.

```
Import re

data=”all indians are brother and sister”
x= re.split(‘\s’,’@’,data,3)
print(x)
```