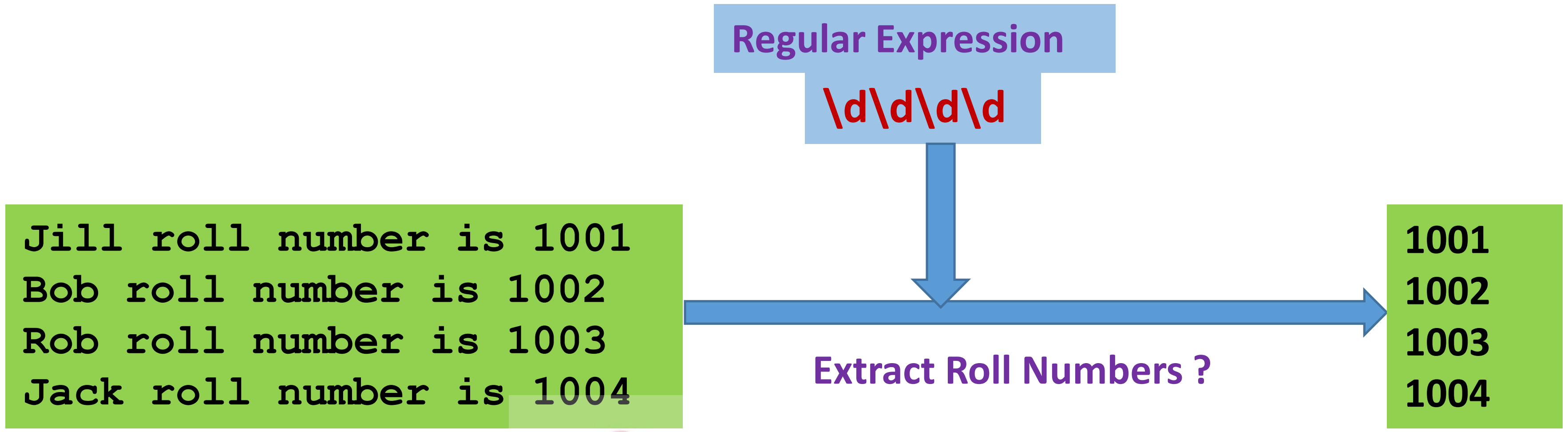
REGULAR EXPRESSIONS IN PYTHON

Dedicated to my sweet friend Jahnavi ☺

Introduction to Regular Expressions

What is a regular expression?

"A string that defines a text matching pattern"

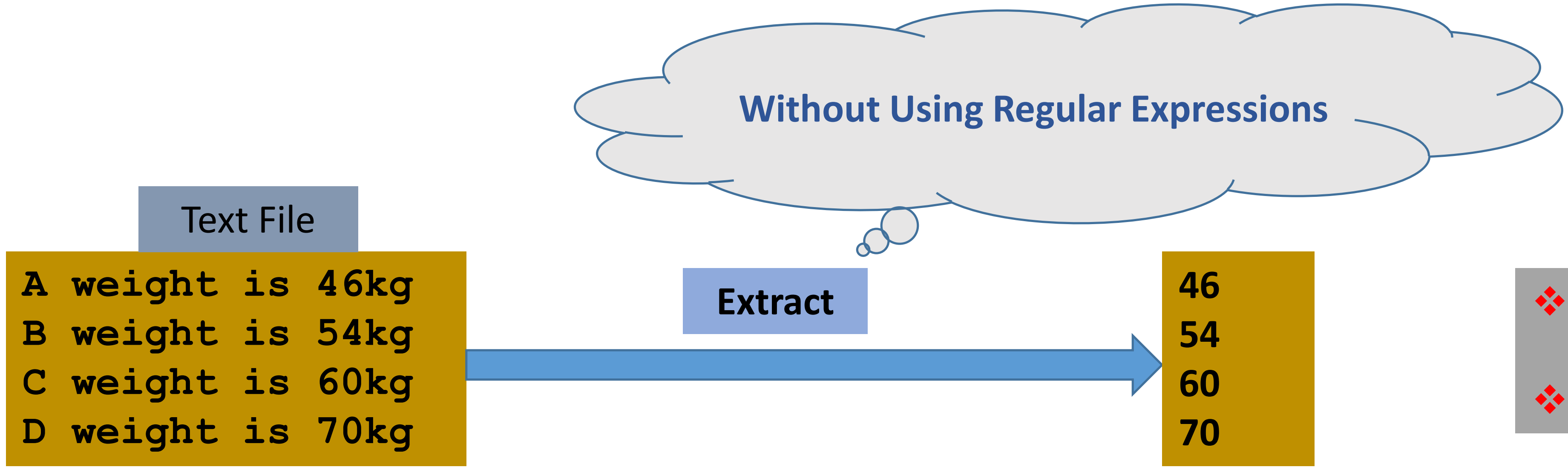


```
Jill roll number is 1001
Bob roll number is 1002
Rob roll number is 1003
Jack roll number is 1004
```

What is the advantage of using regular expressions?

- Using regular expressions, You can extract text which follows a pattern by writing only very few lines of codes

Example

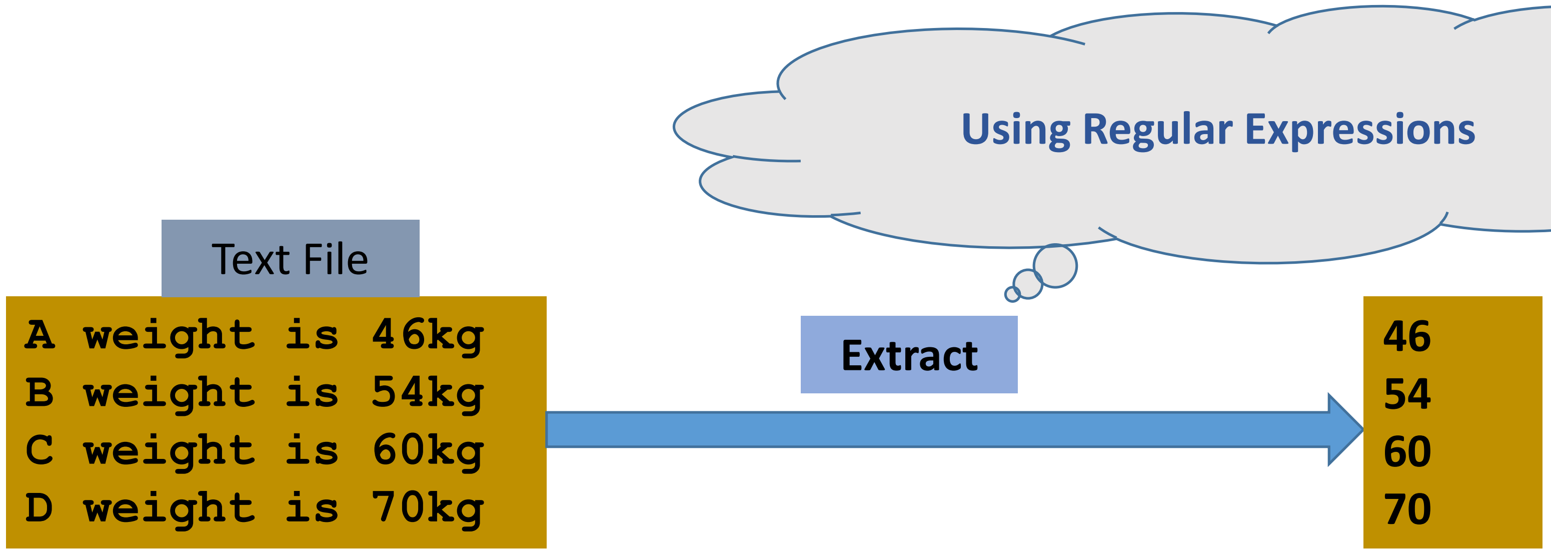


```
A weight is 46kg
B weight is 54kg
C weight is 60kg
D weight is 70kg
```

```
46
54
60
70
```

- **Lengthy Code**
- **Complex**

Example



```
A weight is 46kg
B weight is 54kg
C weight is 60kg
D weight is 70kg
```

```
46
54
60
70
```

- **Less Code**
- **Simple**

re MODULE BASICS

re Module

- Python supports regular expressions through re module
- That is, you have to import re module for using regular expressions

```
import re
```

- No need to explicitly install this module

Steps

```
import re

text='Kalyan roll number is 7004'
pattern=r'\d\d\d\d'

regex=re.compile(pattern)

number=regex.findall(text)
print(number)
```

Import re module

Write regular expression

Create regex object

Call the function using regex object

re MODULE FUNCTIONS

findall()

- Looks for the match any where in the string
- Returns all matched substrings as a list if there is match, otherwise returns empty list

```
regex=re.compile(pattern)
values=regex.findall(text)
```

finditer()

- Looks for the match any where in the string
- Returns objects for all matched substrings as a list if there is a match, otherwise returns empty list

```
regex=re.compile(pattern)
moList=regex.finditer(text)
```

sub()

- replaces all the matched substrings with the given replString and returns the modified string, if there is match
- Returns original string, if there is no match
- Similar to replace option in text editors

```
regex=re.compile(pattern)
regex.sub(replString,text)
```

split()

- Looks for match anywhere in the string
- Splits the string at the matched substrings and returns the splitted string as a list
- Returns original string, if there is no match
- Similar to split() method in strings

```
regex=re.compile(pattern)
regex.split(text)
```

GROUPS

Groups

- You want to match a substring in a string and want to extract a part of matched substring, grouping is used.

Match the roll number CS1004 and extract the last four digits

```
text='My roll number is CS1004'
pattern=r'CS(\d\d\d\d)'
```

Groups - Types

- Numbered Groups
- Named Groups
- Non-capturing Groups

Numbered Groups

```
import re

text='Kalyan roll number is CS1004'

pattern='(CS)(\d\d\d\d)'
regex=re.compile(pattern)

mo=regex.search(text)
print(mo.group())    #prints CS1004
print(mo.group(0))   #prints CS1004
print(mo.group(1))   #prints CS
print(mo.group(2))   #prints 1004
print(mo.groups())   #prints (CS,1004)
```

Named Groups

- When groups are large in number, it is difficult to remember the group numbers

- In such a case, we use named groups

```
import re

text='Kalyan roll number is CS1004'

pattern=r'(?P<branch>CS)(?P<roll>\d\d\d\d)'
regex=re.compile(pattern)

mo=regex.search(text)
print(mo.group())              #prints CS1004
print(mo.group(0))             #prints CS1004
print(mo.group('branch'))      #prints CS
print(mo.group('roll'))        #prints 1004
print(mo.groups())             #prints (CS,1004)
```

NonCapturing Group(?:)

```
import re
text='My  personal number is 043-225431  and my office number is 043-225143'

pattern1='\d\d\d-\d\d\d\d\d'
regex=re.compile(pattern1)
numbers=regex.findall(text)
print(numbers)
```

```
['043-22543', '043-22514']
```

```
pattern2='(\d\d\d)-(\d\d\d\d\d)'
regex=re.compile(pattern2)
numbers=regex.findall(text)
print(numbers)
```

```
[('043', '22543'), ('043', '22514')]
```

```
pattern3='(?:\d\d\d)-(?:\d\d\d\d\d)'
regex=re.compile(pattern3)
numbers=regex.findall(text)
print(numbers)
```

```
['043-22543', '043-22514']
```

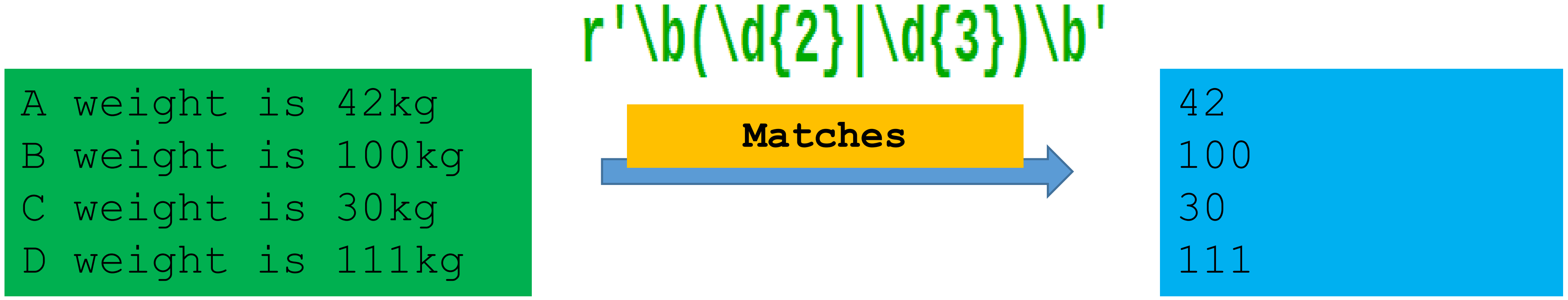
META CHARACTERS

Meta Characters

| (pipe)
? (question mark)
* (asterisk)
+ (plus symbol)
. (dot symbol)

| (pipe)

Matches one of the many characters



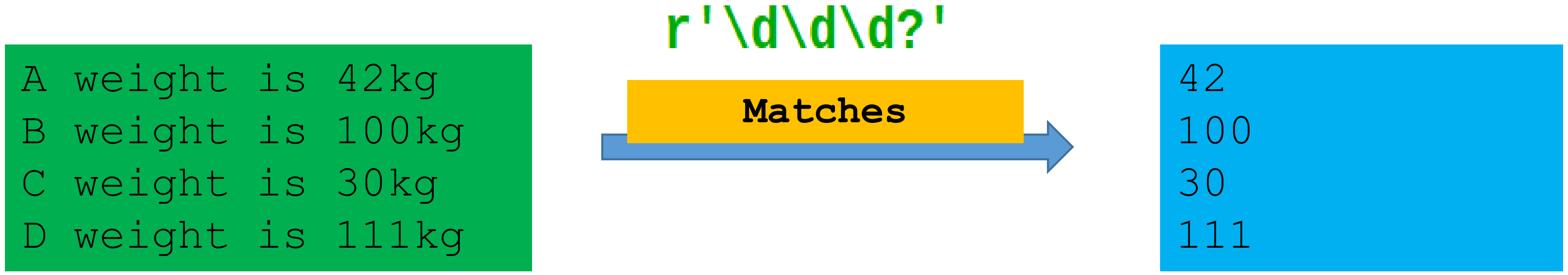
```
r'\b(\d{2}|\d{3})\b'
```

```
A weight is 42kg
B weight is 100kg
C weight is 30kg
D weight is 111kg
```

```
42
100
30
111
```

?(question mark)

Matches zero or one occurrence



```
A weight is 42kg
B weight is 100kg
C weight is 30kg
D weight is 111kg
```

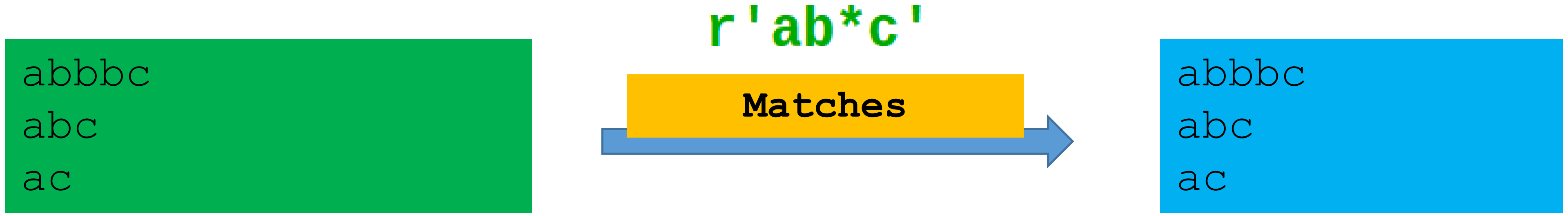
`r'\d\d\d?'`

Matches

```
42
100
30
111
```

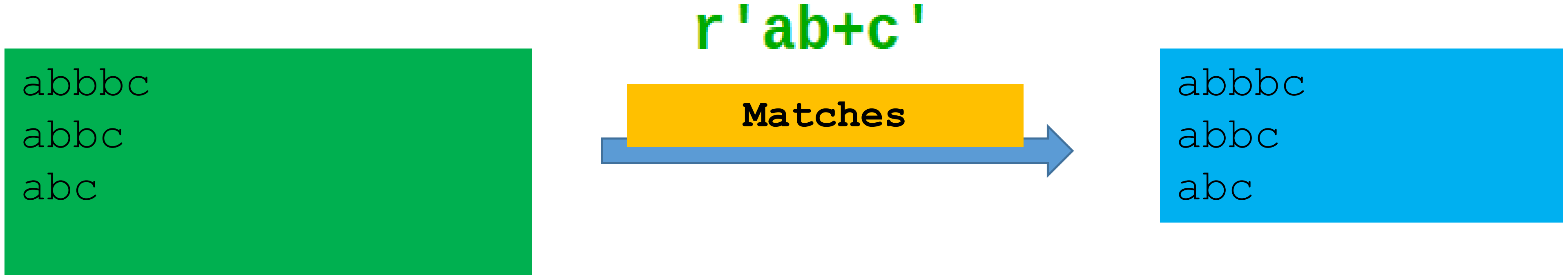
*(asterisk)

Matches zero or more occurrence



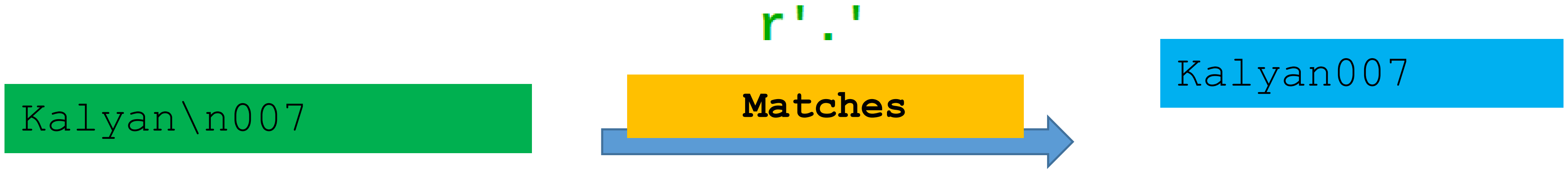
+(plus symbol)

Matches one or more occurrence



. (dot symbol)

Matches any character except '\n'



MATCHING REPETITIONS

pattern{m}

Matches exactly m repetitions

`r'\d{3}'`

equivalent to `r'\d\d\d'`

Matches exactly 3 digits

pattern{m,n}

Matches minimum of m repetitions & maximum of n repetitions

pattern{m,}

Matches a minimum of m repetitions

```
r'\d{3,}'
```

Matches exactly 3 digits

Matches exactly 4 digits

Matches exactly 5 digits

Matches exactly 6 digits

.

.

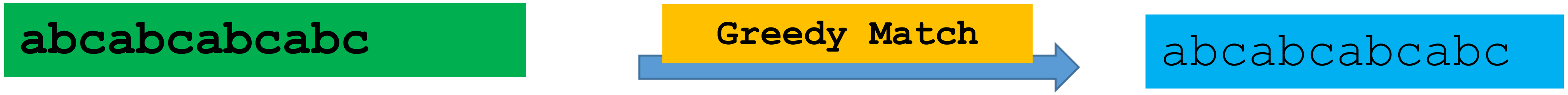
.

GREEDY & NON GREEDY MATCHING

Greedy Matching

Looks for the maximum possible match

```
pattern=r'a[a-z]+c'
regex=re.compile(pattern)
mo=regex.search(text)
```



NonGreedy Matching(?)

Looks for the minimum possible match

```
pattern=r'a[a-z]+?c'
regex=re.compile(pattern)
mo=regex.search(text)
```

abcabcabcabc $\xrightarrow{\text{NonGreedy Match}}$ abc

CHARACTER CLASSES

Character Classes

Matches one of the many characters

Types

- Positive Character Class
- Negative Character Class
- Shorthand Character Class

Positive Character Class

Matches one of the characters specified in []

[abc]	Matches a or b or c
[aeiou]	Matches a ,e,i,o,u
[0123456789]	Matches numbers 0 to 9
[a-c0-9]	Matches a,b,c or 0 to 9

Negative Character Class

Matches any character other than the characters specified in [^]

[^aeiou] → Matches other an aeiou

```
b1001
c1002
d1003
f1004
h1005
```

r'[^aeiou]\d{4}' →

```
b1001
c1002
d1003
f1004
h1005
```

Shorthand Character Class

\d	Matches	any decimal digit	Equivalent to	[0-9]
\D	Matches	Any non- digit	Equivalent to	[^0-9]
\w	Matches	any word character	Equivalent to	[a-zA-z_0-9]
\W	Matches	any non word char	Equivalent to	[^a-zA-Z_0-9]
\s	Matches	any space character	Equivalent to	[\n\t\r\f\v]
\S	Matches	Any non space char	Equivalent to	[^\n\t\v\r\f]

BACK REFERENCES

Numbered Back references

The numbers are 1116,1414,2020,4035

↓

1414, 2020

Numbered Back references

Office Land Line number is 043405117

Office Land Line number is 043-405117

Named Back references

The numbers are 1116,1414,2020,4035

↓

1414, 2020

Named Back references

Office Land Line number is 043405117

$\downarrow$

Office Land Line number is 043-405117

ASSERTIONS

Assertions

Look Ahead Assertions

Positive Look Ahead Assertions
Negative Look Ahead Assertions

Look Behind Assertions

Positive Look Behind Assertions
Negative Look Behind Assertions

Positive Look Ahead Assertion

```
Kalyan_cs,Meghana_cs,John,Jack
```

$\downarrow$

```
Kalyan,Meghana
```

Negative Look Ahead Assertion

```
Values are 12,13,14a,15b
```

```
12,13
```

Positive Look Behind Assertion

```
CS1001,CS1002,CS1003,1989
```

$\downarrow$

```
1001,1002,1003
```

Negative Look Behind Assertion

```
CS1001,CS1002,CS1003,1989
```

```
1989
```