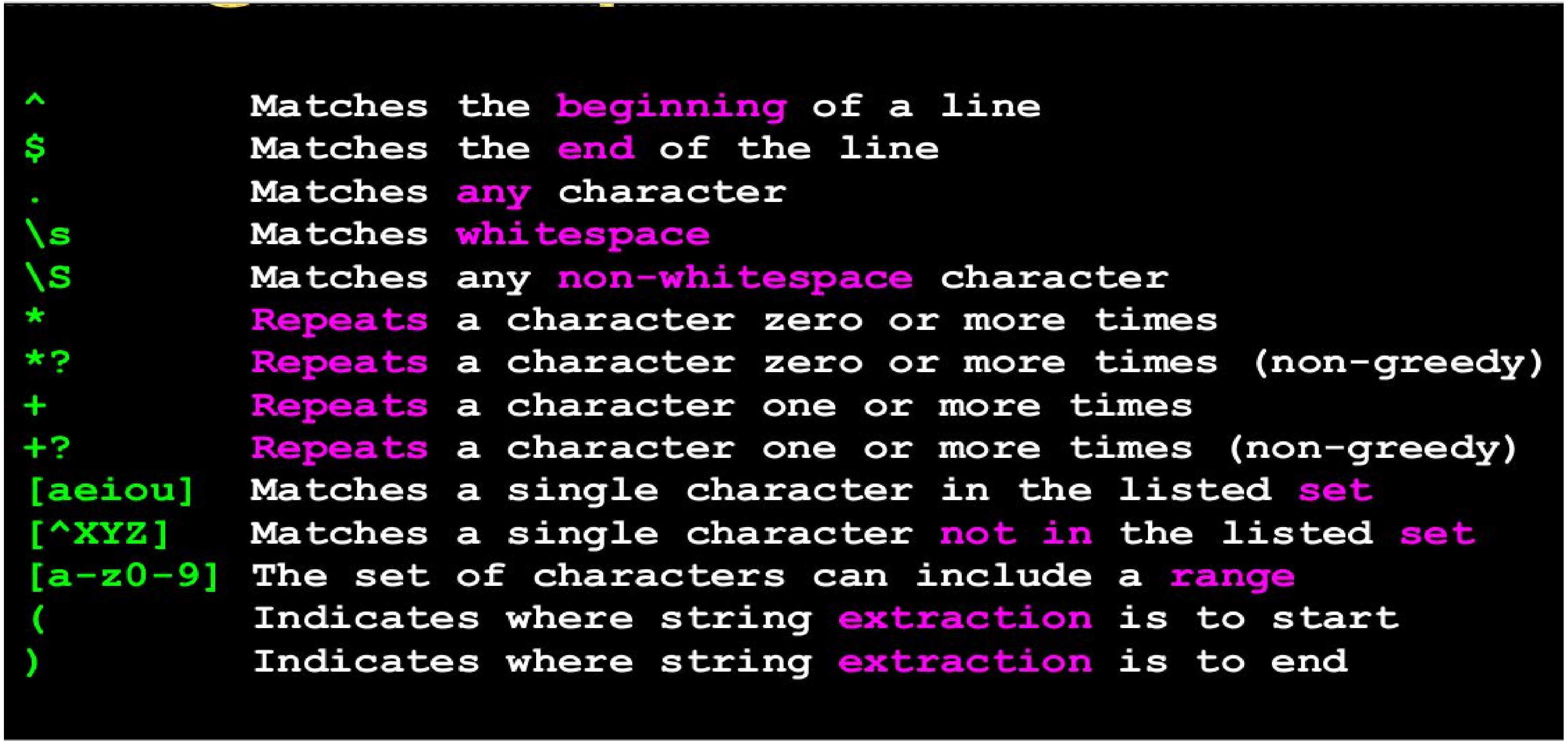
Regular Expressions Using Python

By: Rahul Bajaj

What exactly are we going to talk about today?

What is a Regular Expression?

Regular Expression Quick Guide



^	Matches the beginning of a line
$	Matches the end of the line
.	Matches any character
\s	Matches whitespace
\S	Matches any non-whitespace character
*	Repeats a character zero or more times
*?	Repeats a character zero or more times (non-greedy)
+	Repeats a character one or more times
+?	Repeats a character one or more times (non-greedy)
[aeiou]	Matches a single character in the listed set
[^XYZ]	Matches a single character not in the listed set
[a-z0-9]	The set of characters can include a range
(	Indicates where string extraction is to start
)	Indicates where string extraction is to end

REGULAR EXPRESSION MODULE

1. Before you use regular expressions in your program, you must import the library using “ **import re** ”
2. You can use **re.search()** to see if a string matches a regular expression, similar to using the find() method for strings.
3. You can use **re.findall()** to extract portions of strings that match your regular expressions similar to a combination of find() and slicing in python.

Using re.search() like startswith()

```
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if line.startswith('From:'):
        print line
```

```
import re

hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if re.search('^From:', line):
        print line
```

Matching and Extracting Data

- The re.search() returns a True/False depending on whether the string matches the regular expression

- If we actually want the matching strings to be extracted, we use re.findall()

[0-9]+

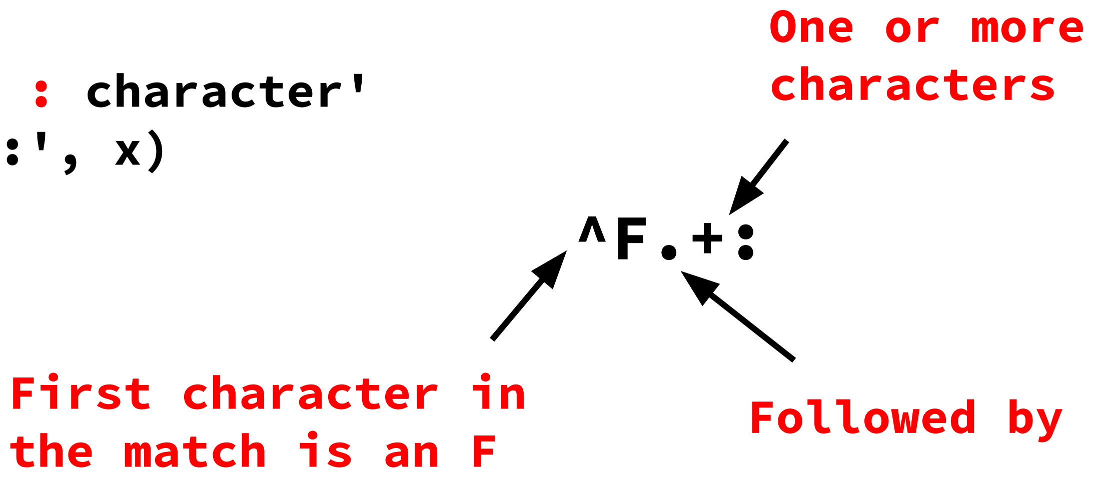
One or more digits

```
>>> import re
>>> x = 'My 2 favorite numbers are 19 and 42'
>>> y = re.findall('[0-9]+',x)
>>> print y
['2', '19', '42']
```

Warning: Greedy Matching

The repeat characters (* and +) push outward in both directions (greedy) to match the largest possible string

```
>>> import re
>>> x = 'From: Using the : character'
>>> y = re.findall('^F.+:', x)
>>> print y
['From: Using the :']
```



Why not 'From:' ?

Non-Greedy Matching

Not all regular expression repeat codes are greedy! If you add a ? character, the + and * chill out a bit...

```
>>> import re
>>> x = 'From: Using the : character'
>>> y = re.findall('^F.+?:', x)
>>> print y
['From:']
```

One or more characters but Not greedy

^F.+?:

First character in the match is an F

Last character in the match is a :

More Examples!!

Extracting a Host Name - Using find() and String Slicing

```
                    21        31
                    ↓         ↓
From stephen.marquard@uct.ac.za Sat Jan5 09:14:16 2008
```

```
>>> data = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
>>> atpos = data.find('@')
>>> print atpos
21
>>> sppos = data.find(' ',atpos)
>>> print sppos
31
>>> host = data[atpos+1 : sppos]
>>> print host
uct.ac.za
```

The Double Split Pattern

Sometimes we split a line one way, and then grab one of the pieces of the line and split that piece again

```
From stephen.marquard@uct.ac.za Sat Jan5 09:14:16 2008
```

```
words = line.split()
email = words[1]
pieces = email.split('@')
print pieces[1]
```

```
stephen.marquard@uct.ac.za
['stephen.marquard', 'uct.ac.za']
' uct.ac.za'
```

Even Cooler Regex Version

From stephen.marquard@uct.ac.za Sat Jan5 09:14:16 2008

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan5 09:14:16 2008'
y = re.findall('^From .*@([^ ]*)',lin)
print y
['uct.ac.za']
```

Summary

1. Regular expressions are a cryptic but powerful language for matching strings and extracting elements from those strings.
2. Regular expressions have special characters that indicate intent.

THANKS!