Department of Computer Science and Engineering PES University, Bangalore, India

Lecture Notes
Python for Computational Problem Solving UE25CS151A

Lecture 74

Introduction to re Module

Prepared By,
Prof. Kundhavai K R,
Assistant Professor
Dept. of CSE, PESU

Intro to the re Module (Regular Expressions):

The re module in Python is for Regular Expressions (often shortened to "RegEx").

Think of it as a super-powerful "Find" tool. Instead of searching for *exact* text (like find('hello')), it lets you search for patterns.

What's a "Pattern"?

- "Find *any* 10-digit phone number."
- "Find *all* email addresses in a document."
- "Find *any* word that starts with 'c' and ends with 't'."
- “Find any 5-letter word that starts with 'h' and ends with 'o'."

You can't do this with basic string methods like .find(). The re module is the special tool for these advanced search jobs.

What is the re Module?

- It's a built-in Python module **(no pip install needed)** that lets us use Regular Expressions.
- Our main goal is to **import re** and use its functions.

It gives us tools to search, find, split, and replace text using patterns.

The Basic Language of Regular Expressions:

A Regular Expression is made up of:

- Normal characters — letters and numbers (a, b, 1, etc.)
- Special characters — called metacharacters

Here are the *most important* ones you need to know:

Meta character	**What It Means**	**Simple Example**
. (dot)	Any single character (except newline)	h.t matches "hat", "hot", "h8t"
\d	Any digit (0-9)	\d\d matches "12", "99", "05"
\w	Any word character (a-z, A-Z, 0-9, _)	\w\w matches "hi", "A5", "b_"
\s	Any whitespace (space,	hello\sworld matches "hello world"

Meta character	What It Means	Simple Example
	tab, newline)	
+	One or more of the thing before it	\d+ matches "1", "123", "99999"
*	Zero or more of the thing before it	ab*c matches "ac", "abc", "abbbc"
[]	A character set (any *one* of these)	[aeiou] matches any single vowel
^	Start of the string	^Hello matches a string *only if* it starts with "Hello"
$	End of the string	world$ matches a string *only if* it ends with "world"

Note:

- Whenever you write a RegEx pattern, use a raw string with **r"..."** in Python. This tells Python not to treat \ as a special escape.
- **Example:**

  pattern = r"\d+" # correct

  pattern = "\\d+" # avoid

The Two Most Useful Functions in re:

Function	Description
re.search()	Finds the first match
re.findall()	Finds all matches and returns a list

1. **re.search()** - scans the entire string from left to right and stops at the first place where the pattern matches.

- This function searches the entire string for the first place the pattern exists.
- It returns a special "Match Object" if it finds one.
- It returns None (Python's "nothing") if it doesn't find a match.

Example 1:

```
import re
text = "My car brand is Audi.I care for it"
pattern = r"c\w\w"  # a 3-letter word starting with 'c'
match = re.search(pattern, text)
if match:
   print(f"Found a match: {match.group()}")
  #Extracts the actual text from the Match object
else:
   print("No match found.")
print(match)
print(match.group())
print(match.start())
print(match.end())
```

Output:

```
Found a match: car
<re.Match object; span=(3, 6), match='car'>
car
3
6
```

2. **re.findall()** - This is often more useful. It finds *all* the matches in the string and returns them as a list.

   If no matches are found, it just returns an empty list.

Example2:

```
import re
text = "My numbers are 123 and 456, and my postcode is 789."
pattern = r"\d"   # one digit
all_numbers = re.findall(pattern, text)
print(all_numbers)
```

Output:

['1', '2', '3', '4', '5', '6', '7', '8', '9']

Example 3:

```
import re
text = "My numbers are 123 and 456, and my postcode is 789."
pattern = r"\d+"   # one or more digit
all_numbers = re.findall(pattern, text)
print(all_numbers)
```

Output:

['123', '456', '789']

Example 4:

```
import re
text = "hello all."
pattern = r"\d+"   # one or more digit
all_numbers = re.findall(pattern, text)
print(all_numbers)
```

Output: #empty list

[]

Practice Examples:

Example 4: Find all words in text

```
import re
text = "Python is easy and fun!"
words = re.findall(r"\w+", text)
print(words)
```

Output:

['Python', 'is', 'easy', 'and', 'fun']

Explanation:

So, \w+ means "find one or more-word characters in a row." This is the perfect pattern for matching a whole word.

Example 5 — Replace all spaces with underscores:

```
import re
text = "Welcome to Python class"
new_text = re.sub(r"\s", "_", text)
print(new_text)
```

Output:

Welcome_to_Python_class

Explanation:

re.sub(pattern to *find*, *replacement* string, original string): This is the "substitute" function.

Example 6 - Find all words starting with “p” or “P”

```
import re
text = "Python programming is pretty popular."
pattern = r"\b[Pp]\w+"
```

```
# \b means word boundary — ensures we only match full words
print(re.findall(pattern, text))
```

Output:

```
['Python', 'programming', 'pretty', 'popular']
```

Example 7 — Find all 3-letter words

```
import re
text = "The fox ran and hid in the den."
pattern = r"\b\w{3}\b"
print(re.findall(pattern, text))
```

Output:

```
['The', 'fox', 'ran', 'and', 'hid', 'the', 'den']
```

Explanation:

\w{3} → exactly 3 characters
\b → start and end of a word

Example 8 — Match all words that end with “ing”

```
import re
text = "I am learning coding and swimming."
pattern = r"\w+ing"
print(re.findall(pattern, text))
```

Output:

```
['learning', 'coding', 'swimming']
```

Example 9— Split text by spaces or commas

```
import re
text = "apple, banana orange,grapes"
pattern = r"[\s,]+"
```

```
print(re.split(pattern, text))
```

Output:

['apple', 'banana', 'orange', 'grapes']

Explanation:

[\s,]+ → split where there are spaces or commas (one or more times).

Example 10 — Replace digits with a symbol

```
import re
text = "My password is 12345"
pattern = r"\d"
new_text = re.sub(pattern, "*", text)
print(new_text)
```

Output:

My password is *****

Example 11 — Validate if text starts with "Hello"

```
import re
text = "Hello world!"
pattern = r"^Hello" #case sensitive
if re.search(pattern, text):
   print("Starts with Hello")
else:
   print("Does not start with Hello.")
```

Output:

Starts with Hello

Example 12 — Find all email addresses

```
import re
text = "Contact us at info@pes.edu or support@pes.com"
pattern = r"\S+@\S+"
```

```
print(re.findall(pattern, text))
# \S = non-space characters → captures emails cleanly.
```

Output:

```
['info@pes.edu', 'support@pes.com']
```

Example 13 — Find all phone numbers with 10 digits

```
import re
text = "Call 9876543210 or 9123456789 or 777900 for details."
pattern = r"\d{10}"
print(re.findall(pattern, text))
```

Output:

```
['9876543210', '9123456789']
```

Example 14 — Find all phone numbers that start with digit 9 and have only 10 digits total

```
import re
text = "Call 9000000000 or 91234567892 or 8123456789 for details."
pattern = r"\b9\d{9}\b"
matches = re.findall(pattern, text)
print("Phone numbers found:", matches)
```

Output:

```
Phone numbers found: ['9000000000']
```

Try this:

r"\b[6-9]\d{9}\b" → matches all mobile numbers starting with 6, 7, 8, or 9

Example 15 - Extract all capitalized words

```
import re
text = "Python is created by Guido van Rossum from Netherlands."
pattern = r"\b[A-Z][a-z]+"
print(re.findall(pattern, text))
# [A-Z] → first letter uppercase, [a-z]+ → remaining lowercase letters.
```

Output:

```
['Python', 'Guido', 'Rossum', 'Netherlands']
```

Example 16 — Check if a string ends with ".edu"

```
import re
text = "www.pes.edu"
pattern = r".edu$"
if re.search(pattern, text):
   print("It's a .edu website")
else:
   print("Not a .edu website")
```

Output:

```
It's a .edu website
```

Example 17 — Find words containing numbers

```
import re
text = "user1 user2 admin boss3 team"
pattern = r"\w*\d\w*"
print(re.findall(pattern, text))
```

Output:

```
['user1', 'user2', 'boss3']
```

Practice Questions:

- Finding all words ending with **'e'**
- Finding all **two-digit** numbers
- Replacing all **vowels** with *
- Extracting all **hashtags** from:
  "Loving #Python #Coding #DataScience"

"Patterns aren't just found — they're *designed* "