Python Programming

Unit 03 – Lecture 04: Regular Expressions (Regex) in Python

Tofik Ali

School of Computer Science, UPES Dehradun

February 14, 2026

Repository: `https://github.com/tali7c/Python-Programming`

Quick Links

Core Concepts Demo Interactive Summary

Agenda

Learning Outcomes

- Explain what regular expressions are and where they are useful

Learning Outcomes

- Explain what regular expressions are and where they are useful
- Use `re.search`, `re.match`, `re.fullmatch`, `re.findall`, `re.sub`

Learning Outcomes

- Explain what regular expressions are and where they are useful
- Use `re.search`, `re.match`, `re.fullmatch`, `re.findall`, `re.sub`
- Understand common meta characters and quantifiers

Learning Outcomes

- Explain what regular expressions are and where they are useful
- Use `re.search`, `re.match`, `re.fullmatch`, `re.findall`, `re.sub`
- Understand common meta characters and quantifiers
- Build simple patterns for validation and extraction

What is Regex?

- A **pattern language** to match text

What is Regex?

- A **pattern language** to match text
- Useful for:

What is Regex?

- A **pattern language** to match text
- Useful for:
  - validation (email, phone, ID formats)

What is Regex?

- A **pattern language** to match text
- Useful for:
  - validation (email, phone, ID formats)
  - extraction (find all numbers, dates, names)

What is Regex?

- A **pattern language** to match text
- Useful for:
  - validation (email, phone, ID formats)
  - extraction (find all numbers, dates, names)
  - cleaning (replace multiple spaces, remove symbols)

Important: Raw Strings for Patterns

- Python uses backslashes in strings (escape sequences)

```
import re
pattern = r"\d{4}-\d{2}-\d{2}"   # YYYY-MM-DD
```

Important: Raw Strings for Patterns

- Python uses backslashes in strings (escape sequences)
- Regex also uses backslashes (e.g., \d)

```
import re
pattern = r"\d{4}-\d{2}-\d{2}"   # YYYY-MM-DD
```

Important: Raw Strings for Patterns

- Python uses backslashes in strings (escape sequences)
- Regex also uses backslashes (e.g., \d)
- Use raw strings: `r"..."`

```
import re
pattern = r"\d{4}-\d{2}-\d{2}"   # YYYY-MM-DD
```

Common Meta Characters

- . any character (except newline)

Common Meta Characters

- `.` any character (except newline)
- `[abc]` one of a,b,c

Common Meta Characters

- `.` any character (except newline)
- `[abc]` one of a,b,c
- `[a-z]` range

Common Meta Characters

- `.` any character (except newline)
- `[abc]` one of a,b,c
- `[a-z]` range
- `\d` digit, `\w` word char, `\s` whitespace

Common Meta Characters

- `.` any character (except newline)
- `[abc]` one of a,b,c
- `[a-z]` range
- `\d` digit, `\w` word char, `\s` whitespace
- Anchors: `^` start, `$` end

Quantifiers

- * 0 or more

```
r"\d+"          # one or more digits
r"\d{10}"       # exactly 10 digits
```

Quantifiers

- * 0 or more
- + 1 or more

```
r"\d+"          # one or more digits
r"\d{10}"       # exactly 10 digits
```

Quantifiers

- * 0 or more
- + 1 or more
- ? 0 or 1

```
r"\d+"          # one or more digits
r"\d{10}"       # exactly 10 digits
```

Quantifiers

- * 0 or more
- + 1 or more
- ? 0 or 1
- {m} exactly m times

```
r"\d+"          # one or more digits
r"\d{10}"       # exactly 10 digits
```

Quantifiers

- * 0 or more
- + 1 or more
- ? 0 or 1
- {m} exactly m times
- {m,n} between m and n times

```
r"\d+"          # one or more digits
r"\d{10}"       # exactly 10 digits
```

Core `re` Functions

- `re.search(p, text)`: match anywhere

Core `re` Functions

- `re.search(p, text)`: match anywhere
- `re.match(p, text)`: match from start

Core `re` Functions

- `re.search(p, text)`: match anywhere
- `re.match(p, text)`: match from start
- `re.fullmatch(p, text)`: entire string must match

Core `re` Functions

- `re.search(p, text)`: match anywhere
- `re.match(p, text)`: match from start
- `re.fullmatch(p, text)`: entire string must match
- `re.findall(p, text)`: all matches as a list

Core `re` Functions

- `re.search(p, text)`: match anywhere
- `re.match(p, text)`: match from start
- `re.fullmatch(p, text)`: entire string must match
- `re.findall(p, text)`: all matches as a list
- `re.sub(p, repl, text)`: replace matches

Example: Validate a Phone Number (10 digits)

```
import re
phone = input("Phone: ").strip()
if re.fullmatch(r"\d{10}", phone):
    print("Valid")
else:
    print("Invalid")
```

Example: Extract Emails

```
import re
text = "Mail us at a@b.com or help@upes.ac.in"
emails = re.findall(r"[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]
print(emails)
```

Demo: Extract + Validate Patterns

- File: `demo/regex_extractor_demo.py`

Demo: Extract + Validate Patterns

- File: `demo/regex_extractor_demo.py`
- Extracts:

Demo: Extract + Validate Patterns

- File: `demo/regex_extractor_demo.py`
- Extracts:
  - email addresses

Demo: Extract + Validate Patterns

- File: `demo/regex_extractor_demo.py`
- Extracts:
  - email addresses
  - Indian-style 10-digit phone numbers

Demo: Extract + Validate Patterns

- File: `demo/regex_extractor_demo.py`
- Extracts:
  - email addresses
  - Indian-style 10-digit phone numbers
- Demonstrates: `findall`, `fullmatch`, `sub`

Checkpoint 1

Question: What is the difference between `search` and `fullmatch`?

Checkpoint 2

Question: Write a regex to match a 4-digit PIN code (e.g., 248001). Should it allow leading zeros?

Think-Pair-Share

Discuss:

- When is regex a good tool?
- When is plain string logic better than regex?

Key Takeaways

- Regex is for pattern matching, validation, and extraction

Key Takeaways

- Regex is for pattern matching, validation, and extraction
- Prefer raw strings `r"..."` for patterns

Key Takeaways

- Regex is for pattern matching, validation, and extraction
- Prefer raw strings `r"..."` for patterns
- Learn meta characters and quantifiers (`.`, `[]`, `+`, `{m,n}`)

Key Takeaways

- Regex is for pattern matching, validation, and extraction
- Prefer raw strings `r"..."` for patterns
- Learn meta characters and quantifiers (`.`, `[]`, `+`, `{m,n}`)
- Use `fullmatch` for validation and `findall` for extraction

Exit Question

Write a one-line regex (pattern only) for a date in format `YYYY-MM-DD`.