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

Tofik Ali

February 14, 2026

Contents

1 Lecture Overview

Regular expressions (regex) are a powerful way to describe text patterns. They are widely used in:

- input validation (phone numbers, emails, dates),
- information extraction (finding all numbers, IDs, links),
- and cleaning data (replacing unwanted characters).

This lecture focuses on practical regex features that you can use in beginner programs.

2 Core Concepts

2.1 Use Raw Strings for Patterns

Regex patterns contain backslashes like `\d`. Python strings also use backslashes for escapes. So it is recommended to use raw strings:

```
pattern = r"\d{4}-\d{2}-\d{2}"
```

2.2 Common Meta Characters (Quick Reference)

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

2.3 Quantifiers

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

Examples:

```
r"\d+" # one or more digits
r"\d{10}" # exactly 10 digits
r"[A-Z]{2}\d{4}" # like AB1234
```

2.4 Core `re` Functions

- `re.search(p, text)`: match anywhere
- `re.match(p, text)`: match only from the start
- `re.fullmatch(p, text)`: entire string must match (great for validation)
- `re.findall(p, text)`: all matches
- `re.sub(p, repl, text)`: replace all matches

```
import re
text = "Marks: 75, 88, 92"
nums = re.findall(r"\d+", text) # ["75", "88", "92"]
```

2.5 Validation vs Extraction

Validation: use `fullmatch`.

```
import re
pin = input("PIN: ").strip()
if re.fullmatch(r"\d{6}", pin):
    print("Valid PIN")
```

Extraction: use `findall` or `finditer`.

```
import re
text = "Call 9876543210 or 9123456789"
phones = re.findall(r"\b\d{10}\b", text)
```

3 Demo Walkthrough

File: `demo/regex_extractor_demo.py`

What it does

- Finds all email addresses in a text block.
- Finds all 10-digit phone numbers.
- Cleans multiple spaces into a single space using `re.sub`.
- Validates a user-provided phone number using `fullmatch`.

4 Interactive Checkpoints (with Solutions)

Checkpoint 1 Solution

Question: difference between `search` and `fullmatch`?

Answer:

- `search` finds a match anywhere inside the text.
- `fullmatch` requires the entire string to match the pattern (best for validation).

Checkpoint 2 Solution

Question: regex for 4-digit PIN? Should it allow leading zeros?

Answer: A 4-digit PIN pattern is `r"\d{4}"`. If leading zeros are allowed, this is fine. If you want to disallow leading zeros, use `r"[1-9]\d{3}"`.

5 Practice Exercises (with Solutions)

Exercise 1: Extract All Integers

Task: Extract all integer numbers from a string.

Solution:

```python
import re
text = input("Text: ")
nums = re.findall(r"-?\d+", text)
print(nums)
```

Exercise 2: Replace Multiple Spaces

Task: Replace multiple spaces with a single space.

Solution:

```python
import re
s = input("Enter text: ")
clean = re.sub(r"\s+", " ", s).strip()
print(clean)
```

Exercise 3: Validate Date (YYYY-MM-DD)

Task: Validate format `YYYY-MM-DD`. (Format only, not calendar correctness.)

Solution:

```python
import re
date = input("Date: ").strip()
if re.fullmatch(r"\d{4}-\d{2}-\d{2}", date):
    print("Valid format")
else:
    print("Invalid format")
```

6 Exit Question (with Solution)

Question: regex pattern for `YYYY-MM-DD`?

Answer: `r"\d{4}-\d{2}-\d{2}"`