ASIAN INSTITUTE OF MANAGEMENT

Data Mining and Wrangling

Regular Expressions

Session 5 and 6

BSDSBA 2028

04 February 2026

ASIAN INSTITUTE OF MANAGEMENT

Session 5 and 6 – Regular Expressions

Gameplan

Part 1 – Regular Expressions: Fundamentals

11:00 AM to 11:10 AM	Class Administrative Matters
11:10 AM to 12:30 NN	Lecture Discussion + Hands-On Coding

Part 2 – Regular Expressions: Practical Example

1:30 PM to 2:00 PM	Class Administrative Matters
2:00 PM to 3:00 PM	In-Class Activity

Class Administrative Matters

Deliverables

ICA02 – Data Formats (due 04 Feb 2026 EOD)

E1 – Data Formats (due 06 Feb 2026 EOD)

E1S2 – Data Formats (optional, due 13 Feb 2026 EOD)

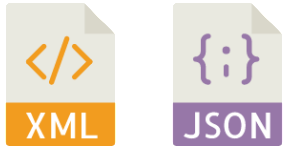
Other

Jojie access

Data Types for Data Mining

Structured

Semi-structured



Unstructured

AIM

Icons made by Smashicons from www.flaticon.com

Regular Expressions

What are regular expressions?

A **Regular Expression**, or **RegEx** for short, is simply a sequence of characters that specifies a certain search pattern.

RegEx can be used to check if a string contains the specified pattern, and provides various operations for string manipulations such as verifying an email or password or even extracting some data.

```
^([a-z0-9_.-]+)@([a-z0-9.-]+)\.([a-z.]{2,6})$
```

Regular Expressions

What are regular expressions?

A **Regular Expression**, or **RegEx** for short, is simply a sequence of characters that specifies a certain search pattern.

RegEx can be used to check if a string contains the specified pattern, and provides various operations for string manipulations such as verifying an email or password or even extracting some data.

```
^([a-z0-9_.-]+)@([a-z0-9.-]+)\.([a-z.]{2,6})$
```

username mail server domain

Regular Expressions

Why use Regular Expressions?

- A lot can be achieved with just a few characters making your code **concise**
- When efficiently implemented, can lead to **faster execution times**
- RegEx is **portable** such that the syntax works in a variety of programming languages

Regular Expressions

How to use Regular Expressions in Python?

Python has a built-in package called `re`, which can be used to apply regular expressions to strings.

We implement a regular expression using the following steps:

1. Specify a **regex pattern string**.
2. Compile the pattern string using `re.compile`.
3. Use the regex object to search a string for the pattern.
4. Perform downstream tasks based on the result.

Regular Expressions

How to use Regular Expressions in Python?

A brief summary of the available regular expression methods in Python is shown below:

Method	Description
`findall`	Return all nonoverlapping matching patterns in a string as a list
`finditer`	Like `findall`, but returns an iterator
`match`	Match pattern at start of string and optionally segment pattern components into groups; if the pattern matches, return a match object, and otherwise `None`
`search`	Scan string for match to pattern, returning a match object if so; unlike `match`, the match can be anywhere in the string as opposed to only at the beginning
`split`	Break string into pieces at each occurrence of pattern
`sub, subn`	Replace all (`sub`) or first `n` occurrences (`subn`) of pattern in string with replacement expression; use symbols `\1, \2, ...` to refer to match group elements in the replacement string

AIM

Regular Expressions

Fundamentals of Regular Expressions

Regular expressions depend on the use of certain special characters called, ***metacharacters***, to express patterns. The rest are called ***literals***, which are normal text characters.

Some examples of metacharacters are:

^ $ [ ] . | ( ) ? * + { }

= < !

Regular Expressions

Regular Expressions Metacharacters

Character Classes

Quantifiers

Position

Other

```
^([a-z0-9_.-]+)@([a-z0-9.-]+)\.([a-z.]{2,6})$
```

Regular Expressions

Why NOT use Regular Expressions?

- Regular expressions relies on **reasonable assumptions** about the structure of the data.
- **Readability** is a real challenge, even for the person who wrote them.
- Errors resulting from wrong patterns don’t have **no tracebacks**.

Reminders

Deliverables

R03 – Regular Expressions (due 04 Feb 2026 EOD)

ICA02 – Data Formats (due 04 Feb 2026 EOD)

E1 – Data Formats (due 06 Feb 2026 EOD)

E1S2 – Data Formats (optional, due 13 Feb 2026 EOD)

ICA03 – Regular Expressions (due 11 Feb 2026 EOD)

E2 – Regular Expressions (due 13 Feb 2026 EOD)