

# Regular Expression - REGEX

---

- A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern.
- RegEx can be used to check if a string contains the specified search pattern.

Steps for regEx:

1. Write 'import re'
2. Write the given text.
3. Define the pattern.
4. Use regex function – to extract the values.
5. Print extracted value/result.

## **RegEX Functions:**

1. findall - Returns a list containing all matches
2. search - Returns a Match object if there is a match anywhere in the string
3. Split - Returns a list where the string has been split at each match
4. Sub - Replaces one or many matches with a string

## **RegEX Match characters:**

- [ ] - A set of characters
- \ - Signals a special sequence (can also be used to escape special characters)
- . - Any character (except newline character)
- ^ - Starts with
- \$ - Ends with
- \* - Zero or more occurrences
- + - One or more occurrences
- ? - Zero or one occurrences
- { } - Exactly the specified number of occurrences
- | - Either or
- ( ) - Capture and group

## ***RegEX Special sequence:***

---

`\A` - Returns a match if the specified characters are at the beginning of the string

`\b` - Returns a match where the specified characters are at the beginning or at the end of a word. (Use "r")

`\B` - Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word. (Use "r")

`\d` - Returns a match where the string contains digits (numbers from 0-9)

`\D` - Returns a match where the string DOES NOT contain digits

`\s` - Returns a match where the string contains a white space character

`\S` - Returns a match where the string DOES NOT contain a white space character

`\w` - Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore `_` character)

`\W` - Returns a match where the string DOES NOT contain any word characters

`\Z` - Returns a match if the specified characters are at the end of the string

## ***RegEX Flag:***

---

1. `re.ASCII` or `re.A` - The `re.ASCII` is relevant to the byte patterns only. It makes the `\w`, `\W`, `\b`, `\B`, `\d`, `\D`, and `\S` perform ASCII-only matching instead of full Unicode matching.
2. `re.DEBUG` - The `re.DEBUG` shows the debug information of compiled pattern
3. `re.IGNORECASE` or `re.I` - Perform case-insensitive matching. It means that the `[A-Z]` will also match lowercase letters.
4. `re.LOCALE` or `re.L` - The `re.LOCALE` is relevant only to the byte pattern. It makes the `\w`, `\W`, `\b`, `\B` and case-sensitive matching dependent on the current locale. The `re.LOCALE` is not compatible with the `re.ASCII` flag.
5. `re.MULTILINE` or `re.M` - The `re.MULTILINE` makes the `^` matches at the beginning of a string and at the beginning of each line and `$` matches at the end of a string and at the end of each line.
6. `re.DOTALL` or `re.S` - By default, the dot `.` matches any characters except a newline. The `re.DOTALL` makes the dot `.` matches all characters including a newline.
7. `re.VERBOSE` or `re.X` - The `re.VERBOSE` flag allows you to organize a pattern into logical sections visually and add comments.

### ### Assignment 1: Extracting Phone Numbers

Raw Text: Extract all valid Pakistani phone numbers from a given text.

Example: Text: Please contact me at 0301-1234567 or 042-35678901 for further details.

---

```
# Step1: Write import re
import re
```

```
# Step2: Sample text with phone numbers
text = "Please contact me at 0301-1234567 or 042-35678901 for
further details."
```

```
# Step3: Define the regex pattern for phone numbers
pattern = r"\b\d{3,4}-\d{7,8}\b"
```

```
# Step4: Find all matches using re.findall()
phone_numbers = re.findall(pattern, text)
phone_numbers
```

Concept:

"Please contact me at 0301-1234567 or 042-35678901 for further details." (We have to extract numbers)

0301-1234567

# 1234 -1234567 = \d{4} - \d{7}

042-35678901

# 123 -12345678 = \d{3} - \d{8}

#here \b = boundary

# here \d = for digits

#in curly brackets { } = we write number of count

Combine both

pattern = r"\b\d{3,4}-\d{7,8}\b"

## ### Assignment 2: Validating Email Addresses

Raw Text: Validate email addresses according to Pakistani domain extensions (.pk).

Example: Text: Contact us at info@example.com or support@domain.pk for assistance.

---

```
# Step1: Write import re
import re
```

```
# Step2: Sample text with email address
text = " Contact us at info@example.com or support@domain.pk
for assistance."
```

```
# Step3: Define the regex pattern for email address
pattern = r"\b[\w.-] + @[\w.-] + \.pk\b'
```

```
# Step4: Find all matches using re.findall()
email_address = re.findall(pattern, text)
email_address
```

Concept:

" Contact us at info@example.com or support@domain.pk for assistance." (We have to extract email address that ends with (.pk))

support@domain.pk  
# [\w.-] + @[\w.-] + \.pk

#here \b = boundary

#here \w = string that contain any word or number or underscore

#here backslash use with .pk – to convert dot into normal character.

pattern = r"\b[\w.-] + @[\w.-] + \.pk\b"

### ### Assignment 2: Validating Email Addresses

Raw Text: Validate email addresses according to Pakistani domain extensions (.pk).

Example: Text: Contact us at info@example.com or support@domain.pk for assistance.

---

```
# Step5: Validate email address
if email_address:
    print(f " Valid email address = {email_address}")
else:
    print("Invalid email address")
```

### ### Assignment 3: Extracting CNIC Numbers

Raw Text: Extract all Pakistani CNIC (Computerized National Identity Card) numbers from a given text.

Example: Text: My CNIC is 12345-6789012-3 and another one is 34567-8901234-5.

```
# Step1: Write import re
import re
```

```
# Step2: Sample text with phone numbers
text = "My CNIC is 12345-6789012-3 and another one is
34567-8901234-5."
```

```
# Step3: Define the regex pattern for cnic numbers
pattern = r"\b\d{5}-\d{7}-\d\b"
```

```
# Step4: Find all matches using re.findall()
cnic = re.findall(pattern, text)
cnic
```

Concept:

" My CNIC is 12345-6789012-3 and another one is 34567-8901234-5." (We have to cnic numbers)

12345-6789012-3

# 12345-1234567-1 = \d{5} - \d{7} - \d

34567-8901234-5

# 12345-1234567-1 = \d{5} - \d{7} - \d

- Both patterns are same, so we can write it once:

#here \b = boundary

# here \d = for digits

#in curly brackets {} = we write number of count

pattern = r"\b\d{5}-\d{7}-\d\b"



### ### Assignment 4: Identifying Urdu Words

Raw Text: Identify and extract Urdu words from a mixed English-Urdu text.

Example: Text: بھی ہیں۔ English words میں کچھ sentence یہ

---

There are two ways to extract urdu words from the given English-Urdu text.

1. Using Unicode concept.
  2. Using regex special sequence and match characters concept.
- 

# Using Unicode Concept:

# Step1: Write import re

import re

# Step2: Sample text with urdu words

text = " بھی ہیں۔ English words میں کچھ sentence یہ "

# Step3: Define the regex pattern for urdu words

pattern = r"\b [\u0600-\u06FF]+ \b "

# Step4: Find all matches using re.findall()

urdu\_text = re.findall (pattern, text)

urdu\_text

Concept:

" بھی ہیں۔ English words میں کچھ sentence یہ " (We have to extract urdu text)

# 0600 - 06FF = Unicode for Arabic / Urdu letters

#here \b = boundary

#here \u = unicode

So;

pattern = r"\b [\u0600-\u06FF]+ \b "

### ### Assignment 4: Identifying Urdu Words

Raw Text: Identify and extract Urdu words from a mixed English-Urdu text.

Example: Text: بھی ہیں۔ English words میں کچھ sentence یہ

---

# using regex special sequence and match characters  
concept:

# Step1: Write import re  
`import re`

# Step2: Sample text with urdu words  
`text = " بھی ہیں۔ English words میں کچھ sentence یہ "`

# Step3: Define the regex pattern for urdu words  
`pattern = r"\b [^ \s A-z ]+ \b "`

# Step4: Find all matches using re.findall()  
`urdu_text = re.findall (pattern, text)`  
`urdu_text`

Concept:

" بھی ہیں۔ English words میں کچھ sentence یہ " (We have to extract urdu text)

#here \b = boundary

#here \s = returns a match where the string contains a white space character

- when we use \s with A-z – it returns urdu words from the text.
- + shows one or more occurrences

So;

`pattern = r"\b [^ \s A-z ]+ \b "`

### ### Assignment 5: Finding Dates

Raw Text: Find and extract dates in the format DD-MM-YYYY from a given text.

Example: Text: The event will take place on 15-08-2023 and 23-09-2023.

---

```
# Step1: Write import re
import re
```

```
# Step2: Sample text with dates
text = " The event will take place on 15-08-2023 and
23-09-2023."
```

```
# Step3: Define the regex pattern for dates
pattern = r"\b\d{1,2}[-?]-\d{1,2}[-?]-\d{2,4}\b"
```

```
# Step4: Find all matches using re.findall()
dates = re.findall(pattern, text)
dates
```

Concept: " The event will take place on 15-08-2023 and 23-09-2023." (We have to extract dates)

15 - 08 - 2023

# 12 - 12 - 1234 = \d{1,2} - \d{1,2} - \d{2,4}  
#here year can be written as: 23 or 2023 that's why \d{2,4}

23-09-2023

# 12 - 12 - 1234 = \d{1,2} - \d{1,2} - \d{2,4}  
#here year can be written as: 23 or 2023 that's why \d{2,4}

- Both patterns are same, so we can write it once:

#here \b = boundary

# here \d = for digits

#in curly brackets { } = we write number of count

```
pattern = r"\b\d{5}-\d{7}-\d\b"
```

### ### Assignment 6: Extracting URLs

Raw Text: Extract all URLs from a text that belong to Pakistani domains.

Example: Text: Visit <http://www.example.pk> or <https://website.com.pk> for more information.

---

```
# Step1: Write import re
import re
```

```
# Step2: Sample text with Pakistani domains
text = " Visit http://www.example.pk or
https://website.com.pk for more information."
```

```
# Step3: Define the regex pattern for url
pattern = r"[https?://] + \S + \.pk "
```

```
# Step4: Find all matches using re.findall()
url = re.findall(pattern, text)
url
```

Concept: " Visit <http://www.example.pk> or <https://website.com.pk> for more information." (We have to extract pakistani url from domain i.e. that contain .pk)

```
http://www.example.pk
# [https?://] + \S + \.pk
```

```
https://website.com.pk
# [https?://] + \S + \.pk
```

- Both patterns are same, so we can write it once:

# here ?: = means colon is optional

# here \S = returns a match where the string does not contain a white space character

# here \.pk = backslash use to convert dot into normal character.

```
pattern = r"[https?://] + \S + \.pk "
```

### ### Assignment 7: Analyzing Currency

Raw Text: Extract and analyze currency amounts in Pakistani Rupees (PKR) from a given text.

Example: Text: The product costs PKR 1500, while the deluxe version is priced at Rs. 2500.

---

```
# Step1: Write import re
import re
```

```
# Step2: Sample text with dates
text = " The product costs PKR 1500, while the deluxe version
is priced at Rs. 2500. “
```

```
#here we add Rs and PKR string in text using re.sub function
new_text = re.sub ("Rs. ", " PKR ", text)
```

```
# Step3: Define the regex pattern for dates
pattern = r"\b\w{3} \d{4}\b"
```

```
# Step4: Find all matches using re.findall()
rupees = re.findall(pattern, new_text)
rupees
```

Concept: " The product costs PKR 1500, while the deluxe version is priced at Rs. 2500."(We have to extract pakistani rupees from text)

PKR 1500

```
# 123 1234 = \w{3} \d{4}
```

Rs. 2500

```
# 123 1234 = \w{3} \d{4}
```

- Both patterns are same, so we can write it once:

# here \d = for digits

# in curly brackets { } = we write number of count

# \w = use for PKR and Rs.

So;

```
pattern = r"\b\w{3} \d{4}\b"
```

### ### Assignment 8: Removing Punctuation

Raw Text: Remove all punctuation marks from a text while preserving Urdu characters.

Example: Text: کیا! آپ, یہاں؟

---

There are two ways to extract urdu words from the given English-Urdu text.

1. Using Unicode concept.
  2. Using regex special sequence and match characters concept.
- 

# using Unicode concept:

# Step1: Write import re  
import re

# Step2: Sample text with urdu words

text = "کیا! آپ, یہاں؟"

# Step3: Define the regex pattern for urdu words

pattern = r"\b [\u0600-\u06FF]+ \b "

# Step4: Find all matches using re.findall()

urdu\_text = re.findall (pattern, text)

urdu\_text

Concept:

"کیا! آپ, یہاں؟" (We have to remove punctuation from the text and extract urdu words)

# 0600 - 06FF = Unicode for Arabic / Urdu letters

#here \b = boundary

#here \u = unicode

So;

pattern = r"\b [\u0600-\u06FF]+ \b "

### ### Assignment 8: Removing Punctuation

Raw Text: Remove all punctuation marks from a text while preserving Urdu characters.

Example: Text: کیا! آپ, یہاں؟

# using regex special sequence and match characters  
concept:

# Step1: Write import re  
import re

# Step2: Sample text with urdu words  
text = "کیا! آپ, یہاں؟"

# Step3: Define the regex pattern for urdu words  
pattern = r"\b [^ \s A-z ]+ \b "

# Step4: Find all matches using re.findall()  
urdu\_text = re.findall(pattern, text)  
urdu\_text

Concept:

"کیا! آپ, یہاں؟" (We have to remove punctuation from the text and extract urdu words)

#here \b = boundary

#here \s = returns a match where the string contains a white space character

- when we use \s with A-z – it returns urdu words from the text.
- + shows one or more occurrences

So;

pattern = r"\b [^ \s A-z ]+ \b "

### ### Assignment 9: Extracting City Names

Raw Text: Extract names of Pakistani cities from a given text.

Example: Text: Lahore, Karachi, Islamabad, and Peshawar are major cities of Pakistan.

---

```
# Step1: Write import re
import re
```

```
# Step2: Sample text with city names
text = " Lahore, Karachi, Islamabad, and Peshawar
are major cities of Pakistan."
```

```
# Step3: Define the regex pattern for city names
pattern = r"[A-Z] [a-z] + \b[^\.]"
```

```
# Step4: Find all matches using re.findall()
cities= re.findall(pattern, text)
cities
```

Concept: " Lahore, Karachi, Islamabad, and Peshawar are major cities of Pakistan." (We have to extract names of cities from text)

Lahore, Karachi, Islamabad, and Peshawar

# [A-Z][a-z] – means word start with capital alphabet and ends with small letters.

# . - (dot) means any character (except newline character)

#because Pakistan also starts with capital alphabet and ends with small letters in the given text so: use \b[^\.] = so that Pakistan will not include in the final return list.

So;

```
pattern = r"[A-Z] [a-z] + \b[^\.]"
```



### ### Assignment 10: Analyzing Vehicle Numbers

Raw Text: Identify and extract Pakistani vehicle registration numbers (e.g., ABC-123) from a text.

Example: Text: I saw a car with the number plate LEA-567 near the market.

---

```
# Step1: Write import re
import re
```

```
# Step2: Sample text with city names
text = " I saw a car with the number plate LEA-567
near the market."
```

```
# Step3: Define the regex pattern for city names
pattern = r "\b [A-Z]{3} - \d{3}\b"
```

```
# Step4: Find all matches using re.findall()
vehicle_no= re. findall(pattern, text)
vehicle_no
```

Concept: " I saw a car with the number plate LEA-567 near the market." (We have to extract names of vehicle number from text)

LEA - 567

# 1 2 3 - 123 = # [A-Z]{3} - \d {3}

# here [A-Z] = alphabets

# here \d = for digits

# in curly brackets { } = we write number of count

So,

pattern = r "\b [A-Z]{3} - \d{3}\b"