CISB5123 Text Analytics
Lab 2
Regular Expressions

A **Regular Expression (RE)** in a programming language is a special text string used for describing a search pattern. It is extremely useful for extracting information from text such as code, files, log, spreadsheets or even documents.

"re" module included with Python primarily used for string searching and manipulation. It is also used frequently for web page "Scraping" (extract large amount of data from websites)

Regular expressions use **TWO (2)** types of characters:

a) Literals

```
a, b, c, A, B, C, 0, 1, 2, 3…
```

b) Metacharacters

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

```
#importing regular expression library
 import re
```

a) re.match()

- re.match( ) is used to check for a **match** at the **beginning of a string**.
- If a match is _found_ in the first line, it *returns the match object*
- If a match is not found in the first line (even though the pattern exists) in some other line), it returns null

```
#using re.match
#to find the first occurrence of the letter 'I' in the string

sentence1 = re.match (r'I', 'I am learning text analytics')
print (sentence1)
```

```
<re.Match object; span = (0, 1), match='I'>
```

The output shows a match because the letter 'I' is at the beginning of the string

```
#using re.match
#to find the first occurrence of the letter 'v' in the string

sentence2 = re.match (r'v', 'I am learning text analytics')
print (sentence2)
```

```
None
```

The output shows None because the letter 'v' does not exist at the beginning of the string

```
#using re.match
#to find the first occurrence of the letter 'am' in the
string

sentence3 = re.match (r'am', 'I am learning text analytics')
print (sentence3)
```

```
None
```

The output shows None because the word 'am' does not exist at the beginning of the string

b) re.search()

- re.search( ) is used to find the **first occurrence of a pattern** in a string **regardless of the location**
- Both re.match( ) and re.search( ) return the first match of a substring found in the string, but re.match( ) checks for a match only at the beginning of the string while re.search( ) checks for a match anywhere in the string

```
sentence4 = re.search(r'am', 'I am learning text analytics')
print (sentence4)
```

```
<re.Match object; span = (2, 4), match='am'>
```

The output shows a match because the word 'am' exists in the string

```
sentence5 = re.search(r'am', 'I am learning text analytics
and am enjoying it')
print (sentence5)
```

```
<re.Match object; span = (2, 4), match='am'>
```

The output is similar with previous example even though there are two (2) 'am' in the string.

However, re.search( ) only returns the first occurrence of 'am' in the string

c) re.findall()

- re.findall( ) is used to find **all** of the **occurrences** of a pattern in a string

```
sentence6 = re.findall(r'am', 'I am learning text analytics
and am enjoying it')
print (sentence6)
```

```
['am', 'am']
```

The output shows ['am', 'am'] because there are two (2) 'am' in the string

d) re.split

- re.split( ) is used to **split** a string by the occurrence of a **given pattern**

```
sentence7 = re.split(r'and', 'I am learning text analytics and
am enjoying it')
print (sentence7)
```

```
['I am learning text analytics ', ' am enjoying it']
```

The string is split into two (2) by the pattern 'and'

```
sentence8 = re.split(r'am', 'I am learning text analytics and
am enjoying it')
print (sentence8)
```

```
['I ', ' learning text analytics and ', ' enjoying it']
```

The string is split into three (3) by the pattern 'am'

```
sentence9 = re.split(r'am', 'I am learning text analytics and
am enjoying it', maxsplit=1)
print (sentence9)
```

```
['I ', ' learning text analytics and am enjoying it']
```

By including the argument maxsplit = 1 (default value is zero), the string is split into two (2) by the first occurrence of 'am'

```
sentence9 = re.split(r'am', 'I am learning text analytics and am
enjoying it', maxsplit=2)
print (sentence9)
```

```
['I ', ' learning text analytics and ', ' enjoying it']
```

By including the argument maxsplit = 2 (default value is zero), the string is split into three (3) by the first two occurrences of 'am'

```
sentence10 = re.split(r'am', 'I am learning text analytics, I
am enjoying it and I am going to ace it', maxsplit=3)
print (sentence10)
```

```
['I ', ' learning text analytics, I ', ' enjoying it and I ', '
going to ace it']
```

By including the argument maxsplit = 3 (default value is zero), the string is split into four (4) by the first three occurrences of 'am'

e) re.sub

- re.sub( ) is used to find the **occurrence** of a given pattern in a string and **replace** with a new value

```
sentence11 = re.sub(r'I', 'we', 'I like text analytics and I enjoy
learning it')
print (sentence11)
```

```
we like text analytics and we enjoy learning it
```

The word ' I ' is changed to 'we' and are replaced in two (2) occurrences

Using metacharacters

- Used for specifying a set of characters to be matched.
- Characters can be listed individually, or a range of characters can be indicated by giving two characters and separating them by a '-'.

a) Metacharacter .

- Find **any characters** in a string (including spaces) *except new line*

```
sentence1 = re.findall (r'.', 'I am learning text analytics')
print (sentence1)

# Each letter is selected including spaces
```

```
['I', ' ', 'a', 'm', ' ', 'l', 'e', 'a', 'r', 'n', 'i', 'n',
'g', ' ', 't', 'e', 'x', 't', ' ', 'a', 'n', 'a', 'l', 'y',
't', 'i', 'c', 's']
```

Return 28 characters (including spaces)

b) Metacharacter \w

- Find **any single character** in a string (excluding spaces) except new line and spaces

```
sentence2 = re.findall (r'\w', 'I am learning
text analytics') print (sentence2)

# Each letter is selected excluding spaces
```

```
['I', 'a', 'm', 'l', 'e', 'a', 'r', 'n', 'i', 'n', 'g', 't',
'e', 'x', 't', 'a', 'n', 'a', 'l', 'y', 't', 'i', 'c', 's']
```

Return 24 characters (including spaces)

c) Metacharacter \w*

- Matches **any characters** with **zero (0) or more characters** in a string including spaces

```
sentence3 = re.findall (r'\w*', 'I am learning text analytics')
print (sentence3)

# Each word is selected including spaces
```

```
['I', '', 'am', '', 'learning', '', 'text', '', 'analytics', '']
```

Return the five (5) words plus spaces

d) Metacharacter \w+

- Matches **one (1) or more characters** in a string excluding spaces

```
sentence4 = re.findall (r'\w+', 'I am learning text analytics')
print (sentence4)

# Each word is selected excluding spaces
```

```
['I', 'am', 'learning', 'text', 'analytics']
```

Return the five (5) words plus spaces

e) Metacharacter ^\w+

- Find the **first word** in a string

```
sentence5 = re.findall (r'^\w+', 'I am learning text analytics')
print (sentence5)

# First word is selected
```

```
['I']
```

f) Metacharacter \w+$

- Matches the **last word** in a string

```
sentence6 = re.findall (r'\w+$', 'I am learning text analytics')
print (sentence6)

# Last word is selected
```

```
['analytics']
```

g) Metacharacter \w\w

- Find **two (2) consecutive characters**

```
sentence7 = re.findall (r'\w\w', 'I am learning text analytics')
print (sentence7)

# 2 consecutive characters are selected
```

```
['am', 'le', 'ar', 'ni', 'ng', 'te', 'xt', 'an', 'al', 'yt',
'ic']
```

'I' is not selected because it only contains one character

h) Metacharacter \b\w\w

- Find **two (2) consecutive characters** in a string

```
sentence8 = re.findall (r'\b\w\w', 'I am learning text analytics')
print (sentence8)

# 2 consecutive characters in a string are selected
```

```
['am', 'le', 'te', 'an']
```

Extracting the domain type of email address

```
sentence9    =    re.findall    (r'@\w+',    'user@text.com.my,
user@analytics.gov.my, user@textanalytics.edu.my')
print (sentence9)

# Only the first word in the domain name is selected
```

```
['@text', '@analytics', '@textanalytics']
```

```
sentence10      =      re.findall      (r'@\w+.\w+','user@text.com.my,
user@analytics.gov.my, user@textanalytics.edu.my')
print (sentence10)
```

```
['@text.com', '@analytics.gov', '@textanalytics.edu']
```

In [20]:

```
sentence11  =  re.findall  (r'@\w+.\w+.\w+',  'user@text.com.my,
user@analytics.gov.my, user@textanalytics.edu.my')
print (sentence11)

# The full domain name is selected
```

```
['@text.com.my', '@analytics.gov.my', '@textanalytics.edu.my']
```

In [21]:

```
# Solution

sentence12 = re.findall (r'@\w+.(\w+.\w+)', 'user@text.com.my,
user@analytics.gov.my, u
print (sentence12)

# To display the type of domain
```

```
['com.my', 'gov.my', 'edu.my']
```

Extracting date

In [22]:

```
sentence13 = re.findall (r'\d{2}-\d{2}-\d{4}', 'Ahmad BIT(IS)
15-05-2001, Johnny BCS(SE) 20-08-2000')
print (sentence13)

# To display the date in the format of dd-mm-yyyy
```

```
['15-05-2001', '20-08-2000']
```

In [23]:

```
sentence14 = re.findall (r'\d{2}-\d{2}-(\d{4})', 'Ahmad BIT(IS)
15-05-2001, Johnny BCS(SE) 20-08-2000')
print (sentence14)

# Only the year will be displayed
```

```
['2001', '2000']
```

In [24]:

```
sentence15 = re.findall(
    r'(\d{2}-\d{2}-)\d{2}(\d{2})',
    'Ahmad BIT(IS) 15-05-2001, Johnny BCS(SE) 20-08-2000'
)

dates = [d + y for d, y in sentence15]
print(dates)
```

```
['15-05-01', '20-08-00']
```

Selecting words that start with vowels in a string

```
In [27]:
sentence16  =  re.findall  (r'[aeiouAEIOU]\w+',  'I  have  eight
story books. I often read them in afternoon')
 print (sentence16)

 #  A  sequence  that  starts  with  a  vowel  followed  by  one  o  rmore
 characters are selected
```

```
['ave',  'eight',  'ory',  'ooks',  'often',  'ead',  'em',  'in',
'afternoon']
```

```
sentence17  =  re.findall  (r'\b[aeiouAEIOU]\w+',  'I  have  eight
story books. I often read them in afternoon')
print (sentence17)

# Only words that start with vowels are selected
```

```
['eight', 'often', 'in', 'afternoon']
```

```
sentence18 = re.findall (r'\b[^aeiouAEIOU\s]\w+', 'I have eight
story books. I often read them in afternoon')
print (sentence18)

# Only words that start with non-vowels are selected
```

```
['have', 'story', 'books', 'read', 'them', 'the']
```

Splitting a string with multiple delimiters

```
sentence19 = re.split (r'[;,]', 'I have many story books, colouring books; I often read them in the afternoon.')
print (sentence19)

# split the words based on the delimiters semi colon and comma
```

```
['I have many story books', ' colouring books', ' I often read them in the afternoon.']
```

```
sentence20 = re.split (r'[;,\s]', 'I have many story books, colouring books; I often read them in the afternoon.')
print (sentence20)

# split the words based on the delimiters semi colon, comma and space
```

```
['I', 'have', 'many', 'story', 'books', '', 'colouring', 'books', '', 'I', 'often', 'read', 'them', 'in', 'the', 'afternoon.']
```

Substituting delimiters

```
sentence21 = re.sub (r'[;,]', '.', 'I have many story books, colouring books; I often read them in the afternoon.')
print (sentence21)

# Substitute the delimiters semi colon and comma with fullstop
```

```
I have many story books. I have many colouring books
```

EXERCISE

1. Write a Python program that extracts all digits from the following text.

```
import re

text = "Order number: 24567, Tracking ID: 8934A12, Quantity: 30"
# Write your regular expression here
pattern = _______

matches = re.findall(pattern, text)
print("Digits found:", matches)
```

Expected output:

```
Digits found: ['2', '4', '5', '6', '7', '8', '9', '3', '4', '1', '2', '3', '0']
```

2. Extract all non-alphanumeric characters (spaces, punctuation, etc.) from the text.

```
text = "Welcome! How's your day going? Call me @ 3pm."
pattern = _______

matches = re.findall(pattern, text)
print("Non-alphanumeric characters:", matches)
```

Expected output:

```
Non-alphanumeric characters: ['!', ' ', "'", ' ', ' ', ' ', '?', ' ', ' ', ' ', '@', ' ', '.']
```

3. Split the text at commas and periods.

```
text = "Data Science, AI, and Machine Learning. These fields are growing fast."
pattern =  _______

split_text = re.split(pattern, text)
print("Split text:", split_text)
```

Expected output:

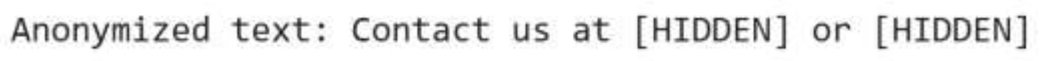
```
Split text: ['Data Science', ' AI', ' and Machine Learning', ' These fields are growing fast',
'']
```

4. Replace email addresses with [HIDDEN].

```
text = "Contact us at support@example.com or info@company.org"
pattern = _______

new_text = re.sub(pattern, "[HIDDEN]", text)
print("Anonymized text:", new_text)
```

Expected output:



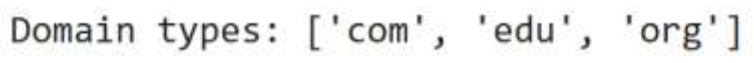
```
Anonymized text: Contact us at [HIDDEN] or [HIDDEN]
```

5. Extract only the domain types (e.g., com, org, edu) from email addresses.

```
text = "Emails: user@gmail.com, admin@university.edu, info@company.org"
pattern = _______

matches = re.findall(pattern, text)
print("Domain types:", matches)
```

Expected output:



```
Domain types: ['com', 'edu', 'org']
```