In [ ]:

```
```

In [1]:

```
import re
```

Question 1- Write a Python program to replace all occurrences of a space, comma, or dot with a colon.

Sample Text- 'Python Exercises, PHP exercises.'

Output: Python:Exercises::PHP:exercises:

In [2]:

```
text = 'Python Exercises, PHP exercises.'
print(re.sub("[ ,.]", ":", text))
```

```
Python:Exercises::PHP:exercises:
```

In [ ]:

```
```

Question 2- Write a Python program to find all words starting with 'a' or 'e' in a given string.

In [3]:

```
# Solution

text = "The following example creates an ArrayList with a capacity of 50 elements. Four

list1 = re.findall("[ae]\w+", text)
print(list1)
```

In [4]:

```
#solution 2

text = "The following example creates an ArrayList with a capacity of 50 elements. Four

pattern = "[ae][\w]+"
for match in re.finditer(pattern, text):
    s = match.start()
    e = match.end()
    print((text[s:e]))
```

```
example
eates
an
ayList
apacity
elements
elements
are
en
added
ayList
and
ayList
ed
accordingly
```

In [5]:

```
#Solution 3

text = "The following example creates an ArrayList with a capacity of 50 elements. Four

pattern = "[ae][\w]+"
for match in re.finditer(pattern, text):
    print(match.group(0))
```

```
example
```

```
eates
an
ayList
apacity
elements
elements
are
en
added
ayList
and
ayList
ed
accordingly
```

[ae][\w]+ - In this pattern [ae] is used to return the one or more occurence of character alphabetically between a and z, lower case or upper case and [\w]+ is used to return zero or more occurrence of any word characters (characters from a to Z, digits from 0-9, and the underscore _ character). So this pattern will return a string starting with 'a' or 'e' followed by any word.

In [ ]:

Question 3- Create a function in python to find all words that are at least 4 characters long in a string. The use of the re.compile() method is mandatory.

In [6]:

```
def specific_char(string):
    pattern = re.compile(r"\b\w{4,}\b")
    string = pattern.findall(string)
    return string

print(specific_char('Create a function in python to find all words that are at least 4 c
```

```
['Create', 'function', 'python', 'find', 'words', 'that', 'least', 'characters', 'long',
'string', '____']
```

In [7]:

```
def specific_char(string):
    pattern = re.compile(r"\w{4,}")
    string = pattern.findall(string)
    return string

print(specific_char('Create a function in python to find all words that are at least 4 c
```

```
['Create', 'function', 'python', 'find', 'words', 'that', 'least', 'characters', 'long',
'string', '____']
```

\b\w{4,}\b - In this pattern \w{4,} is used to extract at least 4 character words.

\b\w{4,} or **\w{4,}\b** can also be used as a pattern for this question.

*,@,#,%,$, etc. are symbols, not characters but underscore is a character.

In [ ]:

Question 4- Create a function in python to find all three, four, and five character words in a string. The use of the re.compile() method is mandatory.

In [8]:

```
def specific_characters(string):
    pattern = re.compile(r"\b\w{3,5}\b")
    string = pattern.findall(string)
    return string
```

```
print(specific_characters('Create a function in, python to$ find all three, four, and fi
```

```
['find', 'all', 'three', 'four', 'and', 'five', 'words', 'in_', 'The', 'use', 'the']
```

\b\w{3,5}\b - In this pattern \w{3,5} is used to extract all three, four, and five character words.

\b is used at the beginning and end of the pattern to extract words with three, four and five characters and white spaces should not be counted. We used \b to ensure that there were three, four and five letters at the beginning and end of a word.

*,@,#,%,$, etc. are symbols, not characters but underscore is a character.

In [ ]:

Question 5- Create a function in Python to remove the parenthesis in a list of strings. The use of the re.compile() method is mandatory.

In [9]:

```
def remove_parenthesis(text):
  for item in text:
    re_parenthesis=re.compile(r" ?\(| ?\)")
    print(re_parenthesis.sub("", item))

items = ["example (.com)", "hr@fliprobo (.com)", "github (.com)", "Hello ( Data Science
remove_parenthesis(items)
```

```
example.com
hr@fliprobo.com
github.com
Hello Data Science World
Data Scientist
```

In [10]:

```
import re
def Remove_parenthesis(string):
    pattern = re.compile(r' ?\(| ?\)')
    word = [pattern.sub('', s) for s in string]
    return word
sample_text = ['example (.com)", "hr@fliprobo (.com)", "github (.com)", "Hello ( Data Sc
list = Remove_parenthesis(sample_text)
for strings in list:
  print(list)
```

```
['example.com", "hr@fliprobo.com", "github.com", "Hello Data Science World", "Data Scien
tist']
```

In the used pattern,

- space or white space followed by ? is used to match zero or one occurence of spaces.
- \(is used to match occurrence of (in the string.
- | is pipe operator which is used to give multiple patterns
- \) is used to match occurence of (in the string.

The given pattern is used to search for spaces, (and). If these are found then it will be removed using sub() method.

In [ ]:

Question 6- Write a python program to remove the parenthesis area from the text stored in the text file using Regular Expression.

In [11]:

```
with open("C:/Users/LENOVO/Documents/RegEx/parenthesis_.txt") as file:
    for item in file:
        print(re.sub(r" ?\([^)]+\)", "", item), end= " ")
```

```
["example", "hr@fliprobo", "github", "Hello", "Data"]
```

Here is a brief explanation of the regular expression used in this question:

- ` ?` matches zero or one occurence if spaces(white spaces).
- `\(` is used to match occurrence of (in the string.
- `[^)]` matches one or more occurence of anything except) in the string.
- `\)` is used to match occurrence of) in the string.

In [ ]:

Question 7- Write a regular expression in Python to split a string into uppercase letters.

Sample text: “ImportanceOfRegularExpressionsInPython”

Output: [‘Importance’, ‘Of’, ‘Regular’, ‘Expression’, ‘In’, ‘Python’]

In [12]:

```
text= "ImportanceOfRegular@ExpressionsInPython"
upper_case = [s for s in re.split("([A-Z][^A-Z]*)", text) if s]

print(upper_case)
```

```
['Importance', 'Of', 'Regular@', 'Expressions', 'In', 'Python']
```

[A-Z][^A-Z]* means any string starting with capital character followed by zero or more occurence of anything except uppercase letters

In [ ]:

In [13]:

```
pattern=r'[A-Z][a-z]*'
string=input("Enter the string : ")
result = re.findall(pattern, string)
print(result)
```

```
Enter the string : ImportanceOfRegularExpressionsInPython
['Importance', 'Of', 'Regular', 'Expressions', 'In', 'Python']
```

[A-Z][a-z]* means any string starting with capital character followed by zero or more occurence of lowercase letters

In [ ]:

Question 8- Create a function in python to insert spaces between words starting with numbers.

Sample Text: “RegularExpression1IsAn2ImportantTopic3InPython"

Expected Output: RegularExpression 1IsAn 2ImportantTopic 3InPython

In [14]:

```
def numbers_spaces(str1):
    return re.sub(r"(\w)([0-9])", r"\1 \2", str1)

print(numbers_spaces("RegularExpression1IsAn2ImportantTopic3InPython"))
```

```
RegularExpression 1IsAn 2ImportantTopic 3InPython
```

In [15]:

```
def putSpace(input):
    words = re.findall('\w[a-zA-Z]*', input)
    print(' '.join(words))

input = 'regularExpression4isan2ImportantTopic3InPython'
putSpace(input)
```

```
regularExpression 4isan 2ImportantTopic 3InPython
```

\w[a-zA-Z]*- means any string starting with any character followed by only many lowercase and uppercase letters.

In this question we have used join() method to join all items in a list into a string, using a space character as separator.

In [ ]:

Question 9- Create a function in python to insert spaces between words starting with capital letters or with numbers.

In [16]:

```
def Insert_space(string):
    words = re.findall(r"[0-9]|[A-Z][a-z]*", string)
    print(' '.join(words))

Insert_space("RegularExpression1IsAn2ImportantTopic3InPython")
```

```
Regular Expression 1 Is An 2 Important Topic 3 In Python
```

In [17]:

```
def putSpace(input):
    words = re.findall('[A-Z0-9][a-z]*', input)
    print(' '.join(words))

input = "RegularExpression1IsAn2ImportantTopic3InPython"
putSpace(input)
```

```
Regular Expression 1 Is An 2 Important Topic 3 In Python
```

In [ ]:

Question 10- Write a python program to extract email address from the text stored in the text file using Regular Expression.

In [18]:

```
import re
with open("C:/Users/LENOVO/Documents/RegEx/Extract Email Addresses from a Text File_New.
        for line in file:
            email = re.findall(r'\b[\w\.]+@[\w\.]+\b\.[\w]{3,5}', line)
            print(email)
```

```
['xyz@domain.com', 'xyz.abc@sdomain.domain.com']
['hr@fliprobo.com']
```

Here is a brief explanation of the regular expression used in this question:

- `\b[\w\.]` matches one or more characters that can be lowercase or uppercase letters, digits, or any of the characters `._%+-`
- `@` matches the `@` symbol
- `\.` matches a literal period (the backslash escapes the period so that it is treated as a literal character, rather than a special character in the regular expression)

- `[\w]{3,5}` matches three or four or five characters.

In [19]:

```
with open("C:/Users/LENOVO/Documents/RegEx/Extract Email Addresses from a Text File_New.
        for line in file:
            pattern= '[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}'
            urls = re.findall(pattern, line)
            print(urls)
```

```
['xyz@domain.com', 'xyz.abc@sdomain.domain.com']
['hr@fliprobo.com']
```

Here is a brief explanation of the regular expression used in this question:

- `[a-zA-Z0-9._%+-]+` matches one or more characters that can be lowercase or uppercase letters, digits, or any of the characters `._%+-`
- `@` matches the `@` symbol
- `[a-zA-Z0-9.-]+` matches one or more characters that can be lowercase or uppercase letters, digits, or any of the characters `.-`
- `\.` matches a literal period (the backslash escapes the period so that it is treated as a literal character, rather than a special character in the regular expression)
- `[a-zA-Z]{2,}` matches two or more characters that can be lowercase or uppercase letters.

In [ ]:

Question 11- Write a Python program to match a string that contains only upper and lowercase letters, numbers, and underscores.

In [20]:

```
# Solution
def text_match(text):
        patterns = '[a-zA-Z]+[0-9_]+'
        x=re.findall(patterns,  text)
        if x:
                print(x)
                return 'Found a match!\n'
        else:
                return('Not matched!')

print(text_match("The quick brown fox jumps over the lazy dog named xYz_123."))
print(text_match("Python_Exercises_1"))
```

```
['xYz_123']
Found a match!

['Python_', 'Exercises_1']
Found a match!
```

In [21]:

```
def text_match(text):
        patterns = '[a-zA-Z]+[0-9_]+|[0-9_]+[a-zA-Z]+'
        x=re.search(patterns,  text)
        if x:
                print(x)
                print(x.group())
                return 'Found a match!\n'
        else:
                return('Not matched!')

print(text_match("The quick brown fox jumps over the lazy dog named XyZ_123."))
print(text_match("The quick brown fox jumps over the lazy dog named 123_XyZ."))
print(text_match("Python_Exercises_1"))
```

```
<re.Match object; span=(50, 57), match='XyZ_123'>
XyZ_123
Found a match!

<re.Match object; span=(50, 57), match='123_XyZ'>
123_XyZ
Found a match!

<re.Match object; span=(0, 7), match='Python_'>
Python_
Found a match!
```

In [ ]:

Question 12- Write a Python program where a string will start with a specific number.

In [22]:

```python
# Solution
def match_num(string):
    text = re.compile(r"^5")
    if text.match(string):
        return True
    else:
        return False
print(match_num('5-2345861'))
print(match_num('6-2345861'))
```

```
True
False
```

In [23]:

```python
def match_num(string):
    x=re.findall(r'\b5[\w]*', string)
    if x:
        return True
    else:
        return False
print(match_num('5-dataScience123'))
print(match_num('6_dataScience123'))
```

```
True
False
```

\b5[\w]asterisk- In this pattern \b5 is used to return a string starting with 5 and [\w]* is used to return zero or more occurrence of any word characters (characters from a to Z, digits from 0-9, and the underscore _ character). So this pattern will return a string starting with 5 followed by any word.

In [ ]:

Question 13- Write a Python program to remove leading zeros from an IP address

In [24]:

```python
ip = "216.08.094.196"
string = re.sub('\.[0]*', '.', ip)
print(string)
```

```
216.8.94.196
```

In [25]:

```python
ip = "216.008.094.196"
string = re.sub('\.[0]*', '.', ip)
print(string)
```

```
216.8.94.196
```

\.[0]asterisk- In this pattern '\.' is used to find the string starting with dot and [0]* is used to find zero or more occurrence of zero. So this code will replace the dot followed by any number of zeroes with a dot.

In [ ]:

```
 
```

Question 14- Write a regular expression in python to match a date string in the form of Month name followed by day number and year stored in a text file.

Sample text : On August 15th 1947 that India was declared independent from British colonialism, and the reins of control were handed over to the leaders of the Country’.

Output- August 15th 1947

In [26]:

```
with open("C:/Users/LENOVO/Documents/RegEx/date_String_in_form_of_month.txt") as file:
        for line in file:
            pattern = "([a-zA-Z]+) (\d+[a-z]+) (\d+)"

            matched = re.search(pattern, line)
            print ("Output: %s" % (matched.group()))
```

```
Output: August 15th 1947
```

([a-zA-Z]+)- It is used to return the one or more occurence of character alphabetically between a and z, lower case or upper case.

(\d+[a-z]+)- In this pattern \d+ is used to find one or more occurrences of digits and [a-z]+ is used to find one or more occurrences of lowercase letters between a to z. Therefore this pattern will return the digits followed by lowercase alphabetic characters.

(\d+)- In this pattern \d+ is used to find one or more occurence of digits.

This pattern will return the date string in the form of Month name followed by day number and year

In [ ]:

```
 
```

Question 15- Write a Python program to search some literals strings in a string.

Sample text : 'The quick brown fox jumps over the lazy dog.'

Searched words : 'fox', 'dog', 'horse'

In [27]:

```
patterns = [ 'fox', 'dog', 'horse' ]
text = 'The quick brown fox jumps over the lazy dog.'
for pattern in patterns:
    print('Searching for "%s" in "%s" ->' % (pattern, text))
    if re.search(pattern,  text):
        print('Matched!\n')
    else:
        print('Not Matched!')
```

```
Searching for "fox" in "The quick brown fox jumps over the lazy dog." ->
Matched!

Searching for "dog" in "The quick brown fox jumps over the lazy dog." ->
Matched!

Searching for "horse" in "The quick brown fox jumps over the lazy dog." ->
Not Matched!
```

In [28]:

```
patterns = [ 'fox', 'dog', 'horse' ]
```

```
text = 'The quick brown fox jumps over the lazy dog.'

for pattern in patterns:
    print('Searching for "%s" in "%s" ' % (pattern, text))
    x=re.findall(pattern,  text)
    if x:
        print(x)
        print('Literal Found\n')
    else:
      print('Literal Not Found')
```

```
Searching for "fox" in "The quick brown fox jumps over the lazy dog."
['fox']
Literal Found

Searching for "dog" in "The quick brown fox jumps over the lazy dog."
['dog']
Literal Found

Searching for "horse" in "The quick brown fox jumps over the lazy dog."
Literal Not Found
```

In [ ]:

Question 16- Write a Python program to search a literals string in a string and also find the location within the original string where the pattern occurs

Sample text : 'The quick brown fox jumps over the lazy dog.'

Searched words : 'fox'

In [29]:

```
pattern = 'fox'
text = 'The quick brown fox jumps over the lazy dog.'
match = re.search(pattern, text)
s = match.start()
e = match.end()
print('Found "%s" in "%s" from %d to %d ' % (match.re.pattern, match.string, s, e))
```

```
Found "fox" in "The quick brown fox jumps over the lazy dog." from 16 to 19
```

Here start() returns the index of the start of the substring matched by regex.search() and end() returns the index of the end of the substring matched by regex.search().

In [ ]:

Question 17- Write a Python program to find the substrings within a string.

Sample text : 'Python exercises, PHP exercises, C# exercises'

Pattern : 'exercises'

Note: There are two instances of exercises in the input string.

In [30]:

```
text = 'Python exercises, PHP exercises, C# exercises'
pattern = 'exercises'
for match in re.findall(pattern, text):
    print('Found "%s"' % match)
```

```
Found "exercises"
Found "exercises"
Found "exercises"
```

In [ ]:

Question 18- Write a Python program to find the occurrence and position of the substrings within a string.

In [31]:

```
# Solution
text = 'Python exercises, PHP exercises, C# exercises'
pattern = 'exercises'
for match in re.finditer(pattern, text):
    s = match.start()
    e = match.end()
    print('Found "%s" at %d:%d' % (text[s:e], s, e))
```

```
Found "exercises" at 7:16
Found "exercises" at 22:31
Found "exercises" at 36:45
```

In [ ]:

Question 19- Write a Python program to convert a date of yyyy-mm-dd format to dd-mm-yyyy format.

In [32]:

```
# Solution
def change_date_format(dt):
        return re.sub(r'(\d{4})-(\d{1,2})-(\d{1,2})', '\\3-\\2-\\1', dt)
dt1 = "2026-01-02"
print("Original date in YYY-MM-DD Format: ",dt1)
print("New date in DD-MM-YYYY Format: ",change_date_format(dt1))
```

```
Original date in YYY-MM-DD Format:  2026-01-02
New date in DD-MM-YYYY Format:  02-01-2026
```

(\d{4})- To find the numbers (0-9) of length between 4 in a given string.

(\d{1,2})-To find the numbers (0-9) of length between 1 to 2 in a given string.

(\d{1,2})- To find the numbers (0-9) of length between 1 to 2 in a given string.

\\3- Used to replace year to day. This means that group reference 3 will be placed at position 1 and similarly \\2 and \\1 will be used.

In [ ]:

Question 20- Create a function in python to find all decimal numbers with a precision of 1 or 2 in a string. The use of the re.compile() method is mandatory.

Sample Text: "01.12 0132.123 2.31875 145.8 3.01 27.25 0.25"

Expected Output: ['01.12', '145.8', '3.01', '27.25', '0.25']

In [33]:

```
def decimal_with_precision(string):

  deci_num = re.compile(r"\b([0-9]+\.[0-9]{1,2})\b")
  result = deci_num.findall(text)
  print(result)

text= "01.12 0132.123 2.31875 145.8 3.01 27.25 0.25"
decimal_with_precision(text)
```

```
['01.12', '145.8', '3.01', '27.25', '0.25']
```

"\b([0-9]+.[0-9]{1,2})\b" - Here we have created a group using () and inside the group we have used-

- [0-9]+ represents one or more repetitions of numbers.
- . matches a dot
- [0-9]{1,2} represents 1 or 2 numbers
- \b is used at the beginning and end of the pattern to extract decimal numbers with a precision of 1 or 2.

In [ ]:

Question 21- Write a Python program to separate and print the numbers and their position of a given string.

In [34]:

```
text = "The following example creates an ArrayList with a capacity of 50 elements. 4 ele

for m in re.finditer("\d+", text):
    print(m.group(0))
    print("Index position:", m.start())
```

```
50
Index position: 62
4
Index position: 75
```

In [ ]:

Question 22- Extract maximum numeric value from a string

In [35]:

```
def extractMax(input):

    numbers = re.findall('\d+',input)
    print(numbers)

    numbers = map(int,numbers)        #converting each number from string into integer.

    print ("Maximum Numeric value is ",max(numbers))

input = 'My marks in each semester are: 947, 896, 926, 524, 734, 950, 642'
extractMax(input)
```

```
['947', '896', '926', '524', '734', '950', '642']
Maximum Numeric value is  950
```

In [1]:

```
import re
def extractMax(input):
    numbers = re.findall('\d+',input)
    print(numbers)
    numbers = list(map(int,numbers))

    max = numbers[0]            #Assume first number in list is largest and initially we h
    for x in numbers:           #Traverse through the list
        if x > max:             #compare each number of the list with the value stored in
            max = x             #Whichever is largest assign that value to variable "max".
    return "Maximum Numeric value is", max                  #It will return the "max" valu

input = 'My marks in each semester are: 947, 896, 926, 524, 734, 950, 642'
extractMax(input)
```

```
['947', '896', '926', '524', '734', '950', '642']
```

Out[1]:

```
('Maximum Numeric value is', 950)
```

In [ ]:

Question 23- Create a function in python to insert spaces between words starting with capital letters.

In [37]:

```
def putSpace(input):
    words = re.findall('[A-Z][a-z]*', input)
    print(' '.join(words))

input = 'BruceWayneIsBatman'
putSpace(input)
```

```
Bruce Wayne Is Batman
```

In this question we have used join() method to join all items in a list into a string, using a space character as separator.

In [38]:

```
#Soultion 2
def putspaces1(str1):
  return re.sub(r"(\w)([A-Z])", r"\1 \2", str1)

print(putspaces1("RegularExpressionIsAnImportantTopicInPython"))
```

```
Regular Expression Is An Important Topic In Python
```

In [ ]:

Question 24- Python regex to find sequences of one upper case letter followed by lower case letters

In [39]:

```
def match(text):
        pattern = '[A-Z][a-z]*$'

        if re.search(pattern, text):
                return('Yes')
        else:
                return('No')


print(match("Welcome"))
print(match("WelcomeHomewelcome Hello Welcome"))
print(match("Welcomes you"))
```

```
Yes
Yes
No
```

In [ ]:

Question 25- Remove consecutive duplicate words from Sentence using Regular Expression

In [40]:

```
def removeDuplicateWords(input):

    regex = r'\b(\w+)(?:\W+\1\b)+'
    return re.sub(regex, r'\1', input)

# Test Case: 1
str1 = "Good bye bye world world"
print(removeDuplicateWords(str1))

# Test Case: 2
str2 = "Ram went went to to his home"
print(removeDuplicateWords(str2))

# Test Case: 3
```

```
str3 = "Hello hello world world"
print(removeDuplicateWords(str3))

# Test Case: 4
str4 = "Hello Hello world World"
print(removeDuplicateWords(str4))
```

```
Good bye world
Ram went to his home
Hello hello world
Hello world World
```

The details of the above regular expression can be understood as:

- “\b” - A word boundary. Boundaries are needed for special cases. For example, in “My thesis is great”, “is” wont be matched twice.
- “\w+” - A word character: [a-zA-Z_0-9]
- “\W+” - A non-word character: [^\w]
- “\1” - Matches whatever was matched in the 1st group of parentheses, which in this case is the (\w+)
- “+” - Match whatever it’s placed after 1 or more times

In [ ]:

Question 26- Program to accept string ending with alphanumeric character

In [41]:

```
regex = '[a-zA-z0-9]$'

def check_alpha_numeric(string):
    if(re.search(regex, string)):
        print("The string is ending with an alphanumeric character. \n")

    else:
        print("The string is not ending with an alphanumeric character. \n")


check_alpha_numeric("pitchumca@")
check_alpha_numeric("pitchumca123")
check_alpha_numeric("pitchum.")
check_alpha_numeric("staysafeindistancestay")
```

```
The string is not ending with an alphanumeric character.

The string is ending with an alphanumeric character.

The string is not ending with an alphanumeric character.

The string is ending with an alphanumeric character.
```

In [ ]:

Question 27- Write a python program using RegEx to extract the hashtags.

Sample Text: text = """RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonetization as the same has rendered USELESS <ed><U+00A0><U+00BD><ed><U+00B1><U+0089> "acquired funds" No wo"""

Output: ['#Doltiwal', '#xyzabc', '#Demonetization']

In [42]:

```
text = """RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonetization as the

hashtags = re.findall(r"#\w+", text)

print("Tweet:\n", text)
print("\n Hashtag:\n", hashtags)
```

```
Tweet:
 RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonetization as the same has
rendered USELESS <ed><U+00A0><U+00BD><ed><U+00B1><U+0089> "acquired funds" No wo

 Hashtag:
 ['#Doltiwal', '#xyzabc', '#Demonetization']
```

In [43]:

```
#solution 2

text = """RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonetization as the

hashtags = re.findall(r"#[a-zA-Z0-9_]+", text)     #

print("Tweet:\n", text)
print("\n Hashtag:\n", hashtags)
```

```
Tweet:
 RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonetization as the same has
rendered USELESS <ed><U+00A0><U+00BD><ed><U+00B1><U+0089> "acquired funds" No wo

 Hashtag:
 ['#Doltiwal', '#xyzabc', '#Demonetization']
```

hash[a-zA-Z0-9_]+-This pattern is used to match one or more characters that begins with a hash (#) and is followed by a lowercase letter or capital letter or a number or an underscore. A plus sign is used after square brackets to match one or more repetitions of the given pattern.

In [ ]:

Question 28- Write a python program using RegEx to remove <U+..> like symbols Check the below sample text, there are strange symbols something of the sort <U+..> all over the place. You need to come up with a general Regex expression that will cover/remove all such symbols.

Sample Text: "@Jags123456 Bharat band on 28??<ed><U+00A0><U+00BD><ed><U+00B8><U+0082>Those who are protesting #demonetization are all different party leaders"

Output: @Jags123456 Bharat band on 28??<ed><ed>Those who are protesting #demonetization are all different party leaders

In [44]:

```
text= "@Jags123456 Bharat band on 28??<ed><U+00A0><U+00BD><ed><U+00B8><U+0082>Those who
clean_text = re.sub(r"<U\+[A-Z0-9]+>", "", text)

print("Text before:\n", text)
print("\n Text after:\n", clean_text)
```

```
Text before:
 @Jags123456 Bharat band on 28??<ed><U+00A0><U+00BD><ed><U+00B8><U+0082>Those who are pr
otesting #demonetization are all different party leaders

 Text after:
 @Jags123456 Bharat band on 28??<ed><ed>Those who are protesting #demonetization are all
different party leaders
```

<U+[A-Z0-9]+>-This pattern is used to match one or more characters enclosed within <> that begin with 'U' followed by a plus sign and then a capital letter (upper case letters) or a number. A plus sign is used after square brackets to match one or more repetitions of a given pattern.

```
In [ ]:
```

Question 29- Write a python program to extract dates from the text stored in the text file.

Sample Text: Ron was born on 12-09-1992 and he was admitted to school 15-12-1999.

Store this sample text in the file and then extract dates.

```
In [45]: import re
         with open("C:/Users/LENOVO/Documents/RegEx/sample_text1.txt") as file:
                 for line in file:
                     emails = re.findall(r"\d{2}-\d{2}-\d{4}", line)
                     print(emails)
```

```
['12-09-1992', '15-12-1999']
```

```
In [ ]:
```

Question 30- Create a function in python to remove all words from a string of length between 2 and 4.

The use of the re.compile() method is mandatory.

Sample Text: "The following example creates an ArrayList with a capacity of 50 elements. 4 elements are then added to the ArrayList and the ArrayList is trimmed accordingly."

Expected Output: following example creates ArrayList a capacity elements. 4 elements added ArrayList ArrayList trimmed accordingly.

```
In [46]: def remove_words(str1):
           shortword = re.compile(r'\b\w{2,4}\b')
           print(shortword.sub('', text))

         text = "The following example creates an ArrayList with a capacity of 50 elements. 4 ele
         remove_words(text)
```

```
 following example creates  ArrayList  a capacity   elements. 4 elements   added   Array
List   ArrayList  trimmed accordingly.
```