Questions on RegEx

Shubham Verma

RegEx Concept https://www.linkedin.com/posts/shubham-verma-3968a5119_regular-expression-regex-activity-6973592766576545792-GI2R?utm_source=share&utm_medium=member_desktop

Linkedin https://www.linkedin.com/in/shubham-verma-3968a5119

Credits W3School for questions

In [ ]:
```
import re
```

1. Write a Python program to check that a string contains only a certain set of characters (in this case a-z, A-Z and 0-9).

In [2]:
```
s1 = "asjdJASDJA123asd123"
s2 = "&%576%^&%7^$%#$$%#)"
```

In [3]:
```
def char_check(test_string):
    pattern = re.compile(r"[^(a-z)+(A-Z)+(0-9)+]")
    check_str = pattern.search(test_string)
    return not bool(check_str)
```

In [4]:
```
char_check(s1)
```

Out[4]:
```
True
```

In [5]:
```
char_check(s2)
```

Out[5]:
```
False
```

2. Write a Python program that matches a string that has an a followed by zero or more b's.

In [6]:
```
def custom_string_match(test_string):
    pattern = re.compile(r"^a(b*)")
    if pattern.search(test_string):
        return "match found"
    else:
        return "match not found"
```

In [7]:
```
s1 = "aaaa"
custom_string_match(s1)
```

Out[7]:
```
'match found'
```

In [8]:
```
s2 = "abbb"
custom_string_match(s2)
```

Out[8]:
```
'match found'
```

In [9]:
```
s3 = "abc"
custom_string_match(s3)
```

Out[9]:
```
'match found'
```

In [10]:
```
s4 = "babbbc"
custom_string_match(s4)
```

Out[10]:
```
'match not found'
```

3. Write a Python program that matches a string that has an a followed by one or more b's.

In [11]:
```
def custom_string_match1(test_string):
    pattern = re.compile(r"^a(b+)")
```

```
    if pattern.search(test_string):
        return "match found"
    else:
        return "match not found"
```

In [12]:
```
s1 = "aaaa"
custom_string_match1(s1)
```

Out[12]:
```
'match not found'
```

In [13]:
```
s2 = "abbb"
custom_string_match1(s2)
```

Out[13]:
```
'match found'
```

In [14]:
```
s3 = "abc"
custom_string_match1(s3)
```

Out[14]:
```
'match found'
```

In [15]:
```
s4 = "babbbc"
custom_string_match1(s4)
```

Out[15]:
```
'match not found'
```

4. Write a Python program that matches a string that has an a followed by zero or one 'b'.

In [16]:
```
def custom_string_match2(test_string):
    pattern = re.compile(r"ab?")
    if pattern.search(test_string):
        return "match found"
    else:
        return "match not found"
```

In [17]:
```
s1 = "aaaa"
custom_string_match2(s1)
```

Out[17]:
```
'match found'
```

In [18]:
```
s2 = "aabbb"
custom_string_match2(s2)
```

Out[18]:
```
'match found'
```

In [19]:
```
s3 = "abc"
custom_string_match2(s3)
```

Out[19]:
```
'match found'
```

In [20]:
```
s4 = "baabbbc"
custom_string_match2(s4)
```

Out[20]:
```
'match found'
```

5. Write a Python program that matches a string that has an a followed by three 'b'.

In [21]:
```
def custom_string_match3(test_string):
    pattern = re.compile(r"ab{3}")
    if pattern.search(test_string):
        return "match found"
    else:
        return "match not found"
```

In [22]:
```
s1 = "abbb"
custom_string_match3(s1)
```

Out[22]:
```
'match found'
```

In [23]:
```
s2 = "ababb"
custom_string_match3(s2)
```

Out[23]:
```
'match not found'
```

6. Write a Python program that matches a string that has an a followed by two to three 'b'.

In [24]:
```
def custom_string_match4(test_string):
    pattern = re.compile(r"ab{2,3}")
    if pattern.search(test_string):
        return "match found"
    else:
        return "match not found"
```

In [25]:
```
s1 = "ababb"
custom_string_match4(s1)
```

Out[25]:
```
'match found'
```

In [26]:
```
s2 = "abbbab"
custom_string_match4(s2)
```

Out[26]:
```
'match found'
```

In [27]:
```
s3 = "abab"
custom_string_match4(s3)
```

Out[27]:
```
'match not found'
```

7. Write a Python program to find sequences of lowercase letters joined with a underscore.

In [28]:
```
def custom_string_match5(test_string):
    pattern = re.compile(r"^[a-z]+_[a-z]+$")
    if pattern.search(test_string):
        return "match found"
    else:
        return "match not found"
```

In [29]:
```
s1 = "krish_naik"
custom_string_match5(s1)
```

Out[29]:
```
'match found'
```

In [30]:
```
s2 = "Krish_naik"
custom_string_match5(s2)
```

Out[30]:
```
'match not found'
```

8. Write a Python program to find the sequences of one upper case letter followed by lower case letters.

In [31]:
```
def custom_string_match6(test_string):
    pattern = re.compile(r"[A-Z]+[a-z]+")
    if pattern.search(test_string):
        return "match found"
    else:
        return "match not found"
```

In [32]:
```
s1 = "Python"
custom_string_match6(s1)
```

Out[32]:
```
'match found'
```

In [33]:
```
s2 = "pyThon"
custom_string_match6(s2)
```

Out[33]:
```
'match found'
```

In [34]:
```
s3 = "PYTHON"
custom_string_match6(s3)
```

Out[34]:
```
'match not found'
```

9. Write a Python program that matches a string that has an 'a' followed by anything, ending in 'b'.

In [35]:
```
def custom_string_match7(test_string):
    pattern = re.compile(r"a.*b$")
    if pattern.search(test_string):
        return "match found"
    else:
        return "match not found"
```

In [36]:
```
s1 = "aabbbbd"
custom_string_match7(s1)
```

Out[36]:
```
'match not found'
```

In [37]:
```
s2 = "aabAbbbc"
custom_string_match7(s2)
```

Out[37]:
```
'match not found'
```

In [38]:
```
s3 = "accddbbjjjb"
custom_string_match7(s3)
```

Out[38]:
```
'match found'
```

In [39]:
```
s4 = "accddbADADbjjjb"
custom_string_match7(s4)
```

Out[39]:
```
'match found'
```

10. Write a Python program that matches a word at the beginning of a string.

In [40]:
```
def custom_string_match8(test_string):
    pattern = re.compile(r"^\w+")
    if pattern.search(test_string):
        return "match found"
    else:
        return "match not found"
```

In [41]:
```
s1 = "@This is new code"
custom_string_match8(s1)
```

Out[41]:
```
'match not found'
```

In [42]:
```
s2 = "This is new code"
custom_string_match8(s2)
```

Out[42]:
```
'match found'
```

11. Write a Python program that matches a word at the end of string, with optional punctuation.

In [43]:
```
def custom_string_match9(test_string):
    pattern = re.compile(r"\w+|[\.\?!]$")
    if pattern.search(test_string):
        return "match found"
    else:
        return "match not found"
```

In [44]:
```
s1 = "This is new code?"
custom_string_match9(s1)
```

Out[44]:
```
'match found'
```

In [45]:
```
s2 = "This is new code"
custom_string_match9(s2)
```

Out[45]:
```
'match found'
```

In [46]:
```
s3 = "This is new code "
custom_string_match9(s3)
```

Out[46]:
```
'match found'
```

12. Write a Python program that matches a word containing 'z'.

In [47]:
```
def custom_string_match10(test_string):
    pattern = re.compile(r"(\w*z.\w*)|(^z\w*)|(\w*z$)")
    if pattern.search(test_string):
        return "match found"
    else:
        return "match not found"
```

In [48]:
```
s1 = "I want to visit a nearby zoo"
custom_string_match10(s1)
```

Out[48]:
```
'match found'
```

In [49]:
```
s2 = "I want to go to a nearby temple"
custom_string_match10(s2)
```

Out[49]:
```
'match not found'
```

13. Write a Python program that matches a word containing 'z', not at the start or end of the word.

In [50]:
```
def custom_string_match11(test_string):
    pattern = re.compile(r"\Bz\B")
    if pattern.search(test_string):
        return "match found"
    else:
        return "match not found"
```

In [51]:
```
s1 = "I want to visit a nearby zoo"
custom_string_match11(s1)
```

Out[51]:
```
'match not found'
```

In [52]:
```
s2 = "I want to visit a nearby ooz"
custom_string_match11(s2)
```

Out[52]:
```
'match not found'
```

In [53]:
```
s3 = "I want to visit a nearby ozso"
custom_string_match11(s3)
```

Out[53]:
```
'match found'
```

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

In [54]:
```
def custom_string_match12(test_string):
    pattern = re.compile(r"^[a-zA-Z0-9_]*$")
    if pattern.search(test_string):
        return "match found"
    else:
        return "match not found"
```

In [55]:
```
s1 = "how_are_you_2022"
custom_string_match12(s1)
```

Out[55]:
```
'match found'
```

In [56]:
```
s2 = "how are you_"
custom_string_match12(s2)
```

Out[56]:
```
'match not found'
```

15. Write a Python program where a string will start with a specific number.

In [57]:
```
def custom_string_match13(test_string, number):
    pattern = re.compile(r"^{}".format(number))
    if pattern.search(test_string):
```

```
        return "String starts with {}".format(number)
    else:
        return "String doesn't start with {}".format(number)
```

In [58]:
```
s1 = "43123891283"
custom_string_match13(s1, 4)
```

Out[58]:
```
'String starts with 4'
```

In [59]:
```
s2 = "93123891283"
custom_string_match13(s2, 4)
```

Out[59]:
```
"String doesn't start with 4"
```

16. Write a Python program to remove leading zeros from an IP address.

In [60]:
```
def custom_string_sub(ip_add):
    pattern = re.compile(r"\.[0]*")
    return pattern.sub(".", ip_add)
```

In [61]:
```
s1 = "192.09.08.0200"
custom_string_sub(s1)
```

Out[61]:
```
'192.9.8.200'
```

17. Write a Python program to check for a number at the end of a string.

In [62]:
```
def custom_string_num_check(test_string):
    pattern = re.compile(r"\d$")
    if pattern.search(test_string):
        return "String ends with a number"
    else:
        return "String doesn't end with a number"
```

In [63]:
```
s1 = "asdJA9"
custom_string_num_check(s1)
```

Out[63]:
```
'String ends with a number'
```

In [64]:
```
s2 = "Krish"
custom_string_num_check(s2)
```

Out[64]:
```
"String doesn't end with a number"
```

18. Write a Python program to search the numbers (0-9) of length between 1 to 3 in a given string.

In [65]:
```
def custom_string_match14(test_string):
    pattern = re.compile(r"[0-9]{1}|[0-9]{2}|[0-9]{3}")
    if pattern.search(test_string):
        return "match found"
    else:
        return "match not found"
```

In [66]:
```
s1 = "D.O.B 1"
custom_string_match14(s1)
```

Out[66]:
```
'match found'
```

In [67]:
```
s2 = "qhasd 12 asd"
custom_string_match14(s2)
```

Out[67]:
```
'match found'
```

In [68]:
```
s3 = "qhasd w345w asd"
custom_string_match14(s3)
```

Out[68]:
```
'match found'
```

In [69]:
```
s4 = "qhasd asd"
custom_string_match14(s4)
```

Out[69]:
```
'match not found'
```

19. Write a Python program to search some literals strings in a string.

In [70]:
```
def custom_string_match15(test_string, input_string, *string_pattern):
    pattern = [i for i in string_pattern]
    for i in pattern:
        if re.search(input_string ,test_string):
            return "match found"
        else:
            return "match not found"
```

In [71]:
```
s1 = "Sudhanshu and Krish are mentors in Ineuron"
custom_string_match15(s1, "Krish", "Krish", "Sudhanshu", "Ineuron")
```

Out[71]:
```
'match found'
```

In [72]:
```
s2 = "Sudhanshu and Krish are mentors in Ineuron"
custom_string_match15(s2, "kamlesh", "Krish", "Sudhanshu", "Ineuron")
```

Out[72]:
```
'match not found'
```

In [73]:
```
s3 = "Sudhanshu and Krish are mentors in Ineuron"
custom_string_match15(s3, "Sudhanshu", "Krish", "Sudhanshu", "Ineuron")
```

Out[73]:
```
'match found'
```

20. 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.

In [74]:
```
def custom_string_match16(test_string, input_string, *string_pattern):
    pattern = [i for i in string_pattern]
    for i in pattern:
        match = re.search(input_string ,test_string)
        if match:
            start_position = match.start()
            end_position = match.end()
            return "match found at position ({} to {})".format(start_position, end_position)
        else:
            return "match not found"
```

In [75]:
```
s1 = "Sudhanshu and Krish are mentors in Ineuron"
custom_string_match16(s1, "Krish", "Krish", "Sudhanshu", "Ineuron")
```

Out[75]:
```
'match found at position (14 to 19)'
```

In [76]:
```
s2 = "Sudhanshu and Krish are mentors in Ineuron"
custom_string_match16(s2, "Sudhanshu", "Krish", "Sudhanshu", "Ineuron")
```

Out[76]:
```
'match found at position (0 to 9)'
```

In [77]:
```
s3 = "Sudhanshu and Krish are mentors in Ineuron"
custom_string_match15(s3, "kamlesh", "Krish", "Sudhanshu", "Ineuron")
```

Out[77]:
```
'match not found'
```

21. Write a Python program to find the substrings within a string.

In [78]:
```
def custom_string_match17(test_string, substring):
    for match in re.findall(substring, test_string):
        if match:
            return "match found"
```

In [79]:
```
s1 = "I have enrolled for FSDS in iNeuron"
custom_string_match17(s1, "FSDS")
```

Out[79]:
```
'match found'
```

In [80]:
```
s2 = "I have enrolled for FSDS in iNeuron"
custom_string_match17(s2, "iNeuron")
```

Out[80]:
```
'match found'
```

22. Write a Python program to find the occurrence and position of the substrings within a string.

In [81]:
```
def custom_string_match18(test_string, substring):
    matches = re.finditer(substring, test_string)
    for match in matches:
        start_string = match.start()
        end_string = match.end()
        if match:
            start_position = match.start()
            end_position = match.end()
            return "match found at position ({} to {})".format(start_position, end_position)
```

In [82]:
```
s1 = "I have enrolled for FSDS in iNeuron"
custom_string_match18(s1, "FSDS")
```

Out[82]:
```
'match found at position (20 to 24)'
```

In [83]:
```
s2 = "I have enrolled for FSDS in iNeuron"
custom_string_match18(s2, "iNeuron")
```

Out[83]:
```
'match found at position (28 to 35)'
```

23. Write a Python program to replace whitespaces with an underscore and vice versa.

In [84]:
```
def replace_custom(test_string):
    if " " in test_string:
        return re.sub(" ","_", test_string)
    else:
        return re.sub("_"," ", test_string)
```

In [85]:
```
s1 = "iNeuron is Tech Company"
replace_custom(s1)
```

Out[85]:
```
'iNeuron_is_Tech_Company'
```

In [86]:
```
s2 = "iNeuron_is_Tech_Company"
replace_custom(s2)
```

Out[86]:
```
'iNeuron is Tech Company'
```

24. Write a Python program to extract year, month and date from a string.

In [87]:
```
def date_finder(test_string):
    pattern = re.compile(r"(\d{4}[-/\.]\d{1,2}[-/\.]\d{1,2})|\d{1,2}[-/\.]\d{1,2}[-/\.]\d{4}")
    matches = pattern.findall(test_string)
    for match in matches:
        print(match)
```

In [88]:
```
s1 = "My DOB is 1999-09-04 and i am 23 years old"
date_finder(s1)
```

```
1999-09-04
```

In [89]:
```
s2 = "My DOB is 1999.09.04 and i am 23 years old"
date_finder(s2)
```

```
1999.09.04
```

In [90]:
```
s3 = "My DOB is 1999/09/04 and i am 23 years old"
date_finder(s3)
```

```
1999/09/04
```

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

In [91]:
```
def date_changer(test_string):
    pattern = re.compile(r"(\d{4})[-/\.](\d{1,2})[-/\.](\d{1,2})")
    return pattern.sub(r"\3-\2-\1", test_string)
```

In [92]:
```
s1 = "My DOB is 1999-09-04 and i am 23 years old"
date_changer(s1)
```

Out[92]:
```
'My DOB is 04-09-1999 and i am 23 years old'
```

In [93]:
```
s2 = "My DOB is 1999.09.04 and i am 23 years old"
date_changer(s2)
```

Out[93]:
```
'My DOB is 04-09-1999 and i am 23 years old'
```

In [94]:
```
s3 = "My DOB is 1999/09/04 and i am 23 years old"
date_changer(s3)
```

Out[94]:
```
'My DOB is 04-09-1999 and i am 23 years old'
```

26. Write a Python program to match if two words from a list of words starting with letter 'P'.

In [95]:
```
def letter_checker(test_string):
    for word in test_string:
            match = re.match("(P\w+)\W(P\w+)", word)
            if match:
                print(match.groups())
```

In [96]:
```
l_word = ["Python PHP", "Java JavaScript", "c c++"]
letter_checker(l_word)
```

```
('Python', 'PHP')
```

27. Write a Python program to separate and print the numbers of a given string.

In [97]:
```
def custom_printer(test_string):
    pattern = re.compile(r"\D+")
    matches = pattern.split(test_string)
    for match in matches:
        print(match)
```

In [98]:
```
s1 = "My 100 favourite 20 course 30 is 40 Python"
custom_printer(s1)
```

```
100
20
30
40
```

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

In [99]:
```
def custom_matcher(test_string):
    pattern = re.compile(r"[ae]\w+")
    matches = pattern.findall(test_string)
    for match in matches:
        print(match)
```

In [100…
```
s1 = "england is engagged in russia-ukraine war"
custom_matcher(s1)
```

```
england
engagged
aine
ar
```

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

In [101…
```
def custom_printer1(test_string):
    pattern = re.compile(r"\d+")
    matches = pattern.finditer(test_string)
    for match in matches:
        start_num = match.start()
        print("The number {} is found at index {}.".format(match.group(), start_num))
```

In [102…
```
s1 = "My 100 favourite 20 course 30 is 40 Python"
```

```
custom_printer1(s1)
```

```
The number 100 is found at index 3.
The number 20 is found at index 17.
The number 30 is found at index 27.
The number 40 is found at index 33.
```

30. Write a Python program to abbreviate 'Road' as 'Rd.' in a given string.

In [103…
```
def abbreviator(test_string, word, rep_str):
    pattern = re.compile(r"{}$|^{}|{}".format(word, word, word))
    return pattern.sub(r"{}".format(rep_str), test_string)
```

In [104…
```
s1 = "road road road"
abbreviator(s1, "road", "rd")
```

Out[104]:
```
'rd rd rd'
```

In [105…
```
s2 = "My name is samiksha"
abbreviator(s2, "samiksha", "sam")
```

Out[105]:
```
'My name is sam'
```

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

In [106…
```
def replace_custom1(test_string):
    pattern = re.compile(r"[ \.,]")
    return pattern.sub(":", test_string)
```

In [107…
```
s1 = "My name is sampoorna, i live in Nagaland."
replace_custom1(s1)
```

Out[107]:
```
'My:name:is:sampoorna::i:live:in:Nagaland:'
```

32. Write a Python program to replace maximum 2 occurrences of space, comma, or dot with a colon.

In [108…
```
def replace_custom1(test_string):
    pattern = re.compile(r"[ \.,]")
    return pattern.sub(":", test_string, 2)
```

In [109…
```
s1 = "My name is sampoorna, i live in Nagaland."
replace_custom1(s1)
```

Out[109]:
```
'My:name:is sampoorna, i live in Nagaland.'
```

33. Write a Python program to find all five characters long word in a string.

In [110…
```
def word_finder(test_string):
    pattern = re.compile(r"\b\w{5}\b")
    matches = pattern.findall(test_string)
    for match in matches:
        print(match)
```

In [111…
```
s1 = "Uinted States basketball team is best among all the teams in the world"
word_finder(s1)
```

```
among
teams
world
```

34. Write a Python program to find all three, four, five characters long words in a string.

In [112…
```
def word_finder1(test_string):
    pattern = re.compile(r"\b\w{3,5}\b")
    matches = pattern.findall(test_string)
    for match in matches:
        print(match)
```

In [113…
```
s1 = "Uinted States basketball team is best in the world"
```

```
word_finder1(s1)
```

```
team
best
the
world
```

35. Write a Python program to find all words which are at least 4 characters long in a string.

In [114…
```
def word_finder2(test_string):
    pattern = re.compile(r"\b\w{4,}\b")
    matches = pattern.findall(test_string)
    for match in matches:
        print(match)
```

In [115…
```
s1 = "Uinted States basketball team is best in the world"
word_finder2(s1)
```

```
Uinted
States
basketball
team
best
world
```

36. Write a python program to convert camel case string to snake case string.

In [116…
```
def camel_to_snake(test_string):
    pattern = re.compile(r"(.)([A-Z][a-z]+)")
    pattern1 = re.compile(r"([a-z0-9])([A-Z])")
    intermediate = re.sub(pattern, r"\1_\2", test_string)
    return re.sub(pattern1, r"\1_\2", intermediate).lower()
```

In [117…
```
s1 = "myClass className"
camel_to_snake(s1)
```

Out[117]:
```
'my_class class_name'
```

37. Write a python program to convert snake case string to camel case string.

In [163…
```
def snake_to_camel(test_string):
    splits_new = []
    test_string = test_string.replace(" ", "_")
    splits = test_string.split("_")
    for i in splits:
        if i != splits[0]:
            splits_new.append(i.capitalize())
    splits_new.insert(0, splits[0])
    camelcase = "".join(i for i in splits_new)

    return camelcase
```

In [164…
```
s2= "my_class class_name"
snake_to_camel(s2)
```

Out[164]:
```
'myClassClassName'
```

38. Write a Python program to extract values between quotation marks of a string.

In [6]:
```
def word_extractor1(test_string):
    pattern = re.compile(r"'(.*?)'")
    matches = pattern.findall(test_string)
    for match in matches:
        print(match)
```

In [7]:
```
s1 = "'ramesh','suresh','kamlesh'"
word_extractor1(s1)
```

```
ramesh
suresh
kamlesh
```

In [11]:
```
def word_extractor2(test_string):
    pattern = re.compile(r'"(.*?)"')
```

```
    matches = pattern.findall(test_string)
    for match in matches:
        print(match)
```

In [12]:
```
s1 = '"ramesh","suresh","kamlesh"'
word_extractor2(s1)
```

```
ramesh
suresh
kamlesh
```

39. Write a Python program to remove multiple spaces in a string.

In [13]:
```
def whitespace_remover(test_string):
    pattern = re.compile(r" +")
    return pattern.sub(" ", test_string)
```

In [14]:
```
s1 = "my      name    is     Smarnika"
whitespace_remover(s1)
```

Out[14]:
```
'my name is Smarnika'
```

40. Write a Python program to remove all whitespaces from a string.

In [2]:
```
def all_space_remover(test_string):
    pattern = re.compile(r" {0,}")
    return pattern.sub("", test_string)
```

In [3]:
```
s1 = "my name is Smarnika"
all_space_remover(s1)
```

Out[3]:
```
'mynameisSmarnika'
```

41. Write a Python program to remove everything except alphanumeric characters from a string.

In [8]:
```
def custom_remover_everything(test_string):
    pattern = re.compile(r"[\W_]+")
    return pattern.sub("", test_string)
```

In [11]:
```
s1 = "**??my name is__ Smarnika___384728397??**"
custom_remover_everything(s1)
```

Out[11]:
```
'mynameisSmarnika384728397'
```

42. Write a Python program to find urls in a string.

In [12]:
```
def url_extractor(test_string):
    pattern = re.compile(r"http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+")
    return pattern.findall(test_string)
```

In [16]:
```
s1 = "https://www.pavithran.net/disha-publication-books-pdf/ https://nicsdtcnagpur.swagatam.gov.in/PublicSite
url_extractor(s1)
```

Out[16]:
```
['https://www.pavithran.net/disha-publication-books-pdf/',
 'https://nicsdtcnagpur.swagatam.gov.in/PublicSite/Login.aspx']
```

43. Write a Python program to split a string at uppercase letters.

In [17]:
```
def upper_split(test_string):
    pattern = re.compile(r"[A-Z][^A-Z]*")
    return pattern.findall(test_string)
```

In [19]:
```
s1 = "MyNameIsMr.T"
upper_split(s1)
```

Out[19]:
```
['My', 'Name', 'Is', 'Mr.', 'T']
```

44. Write a Python program to do a case-insensitive string replacement.

In [24]:
```
def case_insensitive_replacement(test_string, word, replace_word):
```

```
    pattern = re.compile(r"{}".format(word), re.IGNORECASE)
    return pattern.sub(replace_word, test_string)
```

In [25]:
```
s1 = "Hello World"
case_insensitive_replacement(s1, "world", "Planet")
```

Out[25]:
```
'Hello Planet'
```

45. Write a Python program to remove the ANSI escape sequences from a string.

In [26]:
```
def remove_ANSI_escape_sequence(test_string):
    pattern = re.compile(r'\x1b[^m]*m')
    return pattern.sub("", test_string)
```

In [27]:
```
s1 = "\t\u001b[0;35mgoogle.com\u001b[0m \u001b[0;36m216.58.218.206\u001b[0m"
remove_ANSI_escape_sequence(s1)
```

Out[27]:
```
'\tgoogle.com 216.58.218.206'
```

46. Write a Python program to find all adverbs and their positions in a given sentence.

In [37]:
```
def adverb_locator(test_string):
    pattern = re.compile(r"\w+ly")
    matches = pattern.finditer(test_string)
    for match in matches:
        start_position = match.start()
        end_position = match.end()
        return "Adverb: {}, position({}, {})".format(match.group(), start_position, end_position)
```

In [38]:
```
s1 = "My friend arrived early"
adverb_locator(s1)
```

Out[38]:
```
'Adverb: early, position(18, 23)'
```

47. Write a Python program to split a string with multiple delimiters.

Note : A delimiter is a sequence of one or more characters used to specify the boundary between separate, independent regions in plain text or other data streams. An example of a delimiter is the comma character, which acts as a field delimiter in a sequence of comma-separated values.

In [43]:
```
def split_at_delimiter(test_string):
    pattern = re.compile(r"; |,|:|\*|\n")
    return pattern.split(test_string)
```

In [45]:
```
s1 = "Stephen, curry:is \nbest shooter; in NBA"
split_at_delimiter(s1)
```

Out[45]:
```
['Stephen', ' curry', 'is ', 'best shooter', 'in NBA']
```

48. Write a Python program to check a decimal with a precision of 2.

In [47]:
```
def decimal_checker(test_string):
    pattern = re.compile(r"^[0-9]*(\.[0-9]{1,2})$")
    return bool(pattern.search(test_string))
```

In [52]:
```
s1 = "123.23"
decimal_checker(s1)
```

Out[52]:
```
True
```

In [53]:
```
s2 = ".23"
decimal_checker(s2)
```

Out[53]:
```
True
```

In [54]:
```
s3 = "123.2"
decimal_checker(s3)
```

Out[54]:
```
True
```

In [55]:
```
s4 = "123"
decimal_checker(s4)
```

Out[55]:
```
False
```

49. Write a Python program to remove words from a string of length between 1 and a given number.

In [76]:
```
def words_remover_custom(test_string):
    pattern = re.compile(r"\W*\b\w{1,4}\b")
    return pattern.sub("", test_string)
```

In [78]:
```
s1 = "My name is Smarinka Sahoo"
words_remover_custom(s1)
```

Out[78]:
```
' Smarinka Sahoo'
```

50. Write a Python program to remove the parenthesis area in a string.

In [8]:
```
def parenthesis_remover(test_string):
    pattern = re.compile(r" ?\([^)]+\)")
    return pattern.sub("",test_string)
```

In [9]:
```
s1 = "my name is (smarnika)"
parenthesis_remover(s1)
```

Out[9]:
```
'my name is'
```

51. Write a Python program to insert spaces between words starting with capital letters and convert output into sentence case.

In [7]:
```
def insert_space_custom(test_string):
    pattern = re.compile(r"(\w)([A-Z])")
    return pattern.sub(r"\1 \2", test_string).capitalize()
```

In [8]:
```
s1 = "MyNameIsSmarnika"
insert_space_custom(s1)
```

Out[8]:
```
'My name is smarnika'
```