import re # a="Digicomm Semiconductor_123 pvt.lmtd"
p=re.findall("[A-Za-z]",a) p=re.findall("[D-Z]",a) print(b) print(c)
'D', 'i', 'g', 'i', 'c', 'o', 'm', 'm', 'S', 'e', 'm', 'i', 'c', 'o', 'n', 'd', 'u', 'c', 't', 'o', 'r', 'p', 'v', 't', 'l', 'm', 't', 'd'] import re txt = "The rain in Spain"
<pre>txt = "The Fain in Spain" x = re.search("^The.*Spain\$", txt) print(x) if x: print("YES! We have a match!") else:</pre>
print("No match") re.Match object; span=(0, 17), match='The rain in Spain'> ES! We have a match!
<pre>import re txt = "The rain in Spain" x = re.search("r.*Spain\$", txt) print(x)</pre>
re.Match object; span=(4, 17), match='rain in Spain'> txt = "The rain in Spain" x = re.findall("ai", txt)
print(x) ai', 'ai'] import re
txt = "The rain in Spain" x = re.findall("Portugal", txt) print(x)
<pre>txt = "The rain in Spain." x = re.search("\s", txt) y=re.findall("\s", txt)</pre>
orint(x) orint(y) orint("The first white-space character is located in position:", x.start()) re.Match object; span=(3, 4), match=' '>
','',''] ne first white-space character is located in position: 3 import re
txt = "The rain in Spain" x = re.search("Portugal", txt) print(x) one
<pre>import re txt = "The rain in Spain" x = re.split("\s", txt)</pre>
respire("NS", txt) print(x) The', 'rain', 'in', 'Spain'] import re
txt = "The rain in Spain" x = re.split("\s", txt, 1) #1 is the max split print(x)
The', 'rain in Spain'] import re txt = "The rain in Spain"
x = re.split("\s", txt, 2) #2 is the max split print(x) The', 'rain', 'in Spain']
<pre>import re txt = "The rain in Spain" x = re.sub("\s", "9", txt) print(x)</pre>
ne9rain9in9Spain import re
txt = "The rain in Spain" x = re.sub("\s", "9", txt, 2) print(x) ne9rain9in Spain
#Print the position (start- and end-position) of the first match occurrence. #The regular expression looks for any words that starts with an upper case "S": import re
#txt = "The rain in Spain" txt = "The rain in Spou" x = re.search(r"\bS\w+", txt) #S1 a A _ not S. # + is one or more print(x)
x = re.search(r"\bS\w*", txt) #S1 a A _ not S. # * is zero or more print(x) x = re.search(r"\bS\w?", txt) #S1 a A _ not S. # ? is zero or one print(x) print(x) #print(x.span())
re.Match object; span=(12, 16), match='Spou'> re.Match object; span=(12, 16), match='Spou'> re.Match object; span=(12, 14), match='Spou'> re.Match object; span=(12, 14), match='Sp'>
<pre># import re txt = "The rain in Spain" x = re.search(r"\bS\w+", txt) print(x.string)</pre>
orint(x.string) ne rain in Spain $x = re.search(r"\bS\w+", txt) #S1 a A _ not S.$
orint(x) ohn Doe, jane.doe@example.com, 555-1234 ohn ohn
ane.doe@example.com 55-1234 str1 = "Hello Sir, how do you do and age is 25 and how are you doing?" print(search("[a-m]", str1))
re.Match object; span=(1, 2), match='e'> print(search("[A-M]", str1))
re.Match object; span=(0, 1), match='H'> #print(search("[m-a]", str1)) #bad character range m-a at position 1 print(search("AH", str1))
re.Match object; span=(0, 1), match='H'> crint(search("^h", str1)) one
orint(search("^Hello", str1)) re.Match object; span=(0, 5), match='Hello'>
orint(search("doing?\$", str1)) # Will take '?' as a special character one orint(search("doing\?\$", str1)) #Using '\' will remove the functionality
re.Match object; span=(55, 61), match='doing?'> print(search("do*", str1)) # Will only search for "do" re.Match object; span=(15, 17), match='do'>
orint(search("do.*", str1)) re.Match object; span=(15, 61), match='do you do and age is 25 and how are you doing?'>
orint(search("do.*\?", str1)) # WIll search for word starting with "do" and have "?" afterwards re.Match object; span=(15, 61), match='do you do and age is 25 and how are you doing?'> orint(search("do+", str1))
re.Match object; span=(15, 17), match='do'> from re import search
str1 = "Hello Sir, how do you do and age is 25 and how are you doing?" orint(search("do.+", str1)) re.Match object; span=(15, 61), match='do you do and age is 25 and how are you doing?'> orint(search("\?.*", str1)) # It also consider 0 element so gives output as "?"
print(search("\?.*", str1)) # It also consider 0 element so gives output as "?" re.Match object; span=(60, 61), match='?'> print(search("\?.+", str1)) # Returns NULL print(search("\?.+", str1)) # Returns NULL
orint(search("He?", str1)) # Just gives 'He' re.Match object; span=(0, 2), match='He'>
orint(search("He.?", str1)) # Give "Hel" re.Match object; span=(0, 3), match='Hel'> str1 = "Hello Sir, how do you do and age is 25 and how are you doing?"
orint(search("He.?1", str1)) # Gives "Hell" re.Match object; span=(0, 4), match='Hell'> orint(search("He.?0", str1)) # Will return NULL as there are more than 1 character for ?
one orint(search("He.{2}o", str1)) re.Match object; span=(0, 5), match='Hello'>
orint(search("He.{3}o", str1)) # Will return NULL one
orint(search("do doing", str1)) # Will return the first match re.Match object; span=(15, 17), match='do'> orint(search("done doing", str1)) # If first one not present then check for next
re.Match object; span=(55, 60), match='doing'> crint(search("done does doing", str1)) # Also support more than 2 terms re.Match object; span=(55, 60), match='doing'>
str1 = "Hello Sir, how do you do and age is 25 and how are you doing ?" print(search("\AHello", str1)) re.Match object; span=(0, 5), match='Hello'>
orint(search("\Ahello", str1)) one
print(search("\A", str1)) # Also takes blank space re.Match object; span=(0, 0), match=''> print(search(r"doing?\b", str1))
re.Match object; span=(55, 60), match='doing'> print(search(r"Hell\b", str1)) one
orint(search(r"do\b", str1)) re.Match object; span=(15, 17), match='do'> orint(search(r"and\b", str1))
re.Match object; span=(25, 28), match='and'> str1 = "Hello Sir, how do you do and age is 25 and how are you doing ?" orint(search(r"\bare", str1))
re.Match object; span=(47, 50), match='are'> print(search("Hell\B", str1)) print(search("do\B", str1))
orint(search("and\B", str1)) orint(search("\Bare", str1)) re.Match object; span=(0, 4), match='Hell'> re.Match object; span=(55, 57), match='do'>
one one orint(search("do\d", str1)) orint(search("\d", str1))
orint(search("\D", str1)) orint(search("do\D", str1)) one re.Match object; span=(36, 37), match='2'>
re.Match object; span=(0, 1), match='H'> re.Match object; span=(15, 18), match='do '> str1 = "Hello Sir, how do you do and age is 25 and how are you doing ?" orint(search("\s", str1))
orint(search("Hell\s", str1)) orint(search("Hello\s", str1)) re.Match object; span=(5, 6), match=' '> one re.Match object; span=(0, 6), match=' Hello '>
re.Match object; span=(0, 6), match='Hello '> print(search("Hell\S", str1)) print(search("Hello\S", str1)) re.Match object; span=(0, 5), match='Hello'>
orint(search("Hell\w", str1)) orint(search("Hello\w", str1))
orint(search("age is \w", str1)) re.Match object; span=(0, 5), match='Hello'> one re.Match object; span=(29, 37), match='age is 2'>
orint(search("Hello\Z", str1)) orint(search("doing\Z", str1)) orint(search("doing\?\Z", str1)) orint(search("doing\?\Z", str1))
one re.Match object; span=(55, 61), match='doing?'> #Sets str1 = "Hello Sir, how do you do and age is 25 and how are you doing ?"
orint(search("[abcd]", str1)) orint(search("[^abcd]", str1)) orint(search("[1-3][1-5]", str1)) re.Match object; span=(15, 16), match='d'>
re.Match object; span=(0, 1), match='H'> re.Match object; span=(36, 38), match='25'> import re from re import *
check_str = "\nHello this is a demo string\nTHAT we Will string Be usIng for PracTicE purposE\n and THAT is very good\n# This is comment number 1\n# This is comment number 2\n This is a NORmal StateMeNt\n#This is the last comment. orint(findall("\bis", check_str, M)) #Make \(^\) and \$ match the start and end of each line. orint(findall("\bis", check_str, M)) orint(findall("\bis", check_str, M)) orint(findall("string\$", check_str, M))
orint(findall("string\$", check_str, M)) orint(findall(".THAT", check_str, S)) #Make . match any character, including newlines. orint(sub("THAT", "that", check_str)) #substitution orint(sub(check_str, check_str.upper(), check_str)) orint(sub(check_str, check_str.lower(), check_str)) orint(sub(check_str, check_str.str)) orint(sub(check_str, check_str.swapcase(), check_str))
is', 'is', 'is', 'is', 'is'] THAT'] Hello'] string']
'\nTHAT', ' THAT'] ello this is a demo string nat we Will string Be usIng for PracTicE purposE
and that is very good This is comment number 1 This is comment number 2 This is a NORmal StateMeNt This is the last comment.
ELLO THIS IS A DEMO STRING HAT WE WILL STRING BE USING FOR PRACTICE PURPOSE AND THAT IS VERY GOOD THIS IS COMMENT NUMBER 1
THIS IS COMMENT NUMBER 2 THIS IS A NORMAL STATEMENT THIS IS THE LAST COMMENT. Ello this is a demo string
ello this is a demo string nat we will string be using for practice purpose and that is very good this is comment number 1 this is comment number 2 this is a normal statement
chis is the last comment. ELLO THIS IS A DEMO STRING nat WE WILL STRING bE USING FOR pRACtice PURPOSe
AND that IS VERY GOOD this IS COMMENT NUMBER 1 this IS COMMENT NUMBER 2 this IS A norMAL STATEMENT this IS THE LAST COMMENT.
check_str = "\nHello this is a demo string\nTHAT we Will string Be usIng for PracTicE purposE\n and THAT is very good\n# This is comment number 1\n# This is comment number 2\n This is a NORmal StateMeNt\n#This is the last comment. This is comment number 1\n', '# This is comment number 2\n']
myfile = open("verilog.sv", 'r') local_data = myfile.read() #print(local_data) orint(findall(r"\bmodule", local_data, I))
<pre>"module', 'module'] # matches.group is a method used to retrieve the captured groups from a match object. date_str = "mamathagollavilli97@gmail.com" temp_var = re.match('(\w+)@(\w+)\.(\w+)', date_str)</pre>
orint(temp_var) orint(temp_var.group(0)) orint(temp_var.group(1))
orint(temp_var.group(2)) orint(temp_var.group(3)) #print(temp_var.group(4)) #no such group re.Match object; span=(0, 29), match='mamathagollavilli97@gmail.com'>
mathagollavilli97@gmail.com
amathagollavilli97@gmail.com amathagollavilli97 nail om
mathagollavilli97 nail pm import re #\1: Matches the same text as the first captured group, which is "repeated". # Example string text = "This is is a test test string with repeated repeated words."
amathagollavilli97 mail om import re #\1: Matches the same text as the first captured group, which is "repeated". # Example string
<pre>import re will Matches the same text as the first captured group, which is "repeated". # Example string text = "This is is a test test string with repeated repeated words." # Pattern to find repeated words pattern = r"\b(\w+\)\s+\1\b" # Find all matches matches = re.findal(pattern, text) print(matches) # Output: ['is', 'test', 'repeated']</pre>
import re **Nil Matches the same text as the first captured group, which is "repeated". **Example string **Pattern to find repeated words sattern = "This is is a test test string with repeated words." **Pattern to find repeated words sattern = refindall matches matches = re.findall(pattern, text) **prind all matches matches = re.findall(pattern, text) **prind matches **Output: ['is', 'test', 'repeated'] **prind matches **Output: ['is', 'test', 'repeated'] **prind matches **Prind mat
mmontre with factors the same text as the first captured group, which is "repeated". **Example string text = "Min is us a test test string with repeated words." **Pattern to find repeated words authors = "Min(w+)\s\1\b\1\b\1\b\1\b\1\c\1\b\1\c\1\b\1\b\1\b
mmethagellavilian mingor re min instences the same sext as the first captured group, which is "repeated". # Example stitum text = "This is is a test test string with repeated words." # Eattern to find repeated words attern = "This (we)sated words
mathemorphism is a second of the first captured group, which is "repeated". **Example string **Example stri
ment inpolity in its a test test structured group, which is "repeated". **Parties to its assess test structured group, which is "repeated". **Parties to its a test test structured group, which is "repeated". **Parties to its a repeated wors. **Parties to its a test test structured group, which is "repeated words." **Parties of its directors are all in a repeated words. **Parties of it
manufacture in a same feat as the first ceptured group, which is "repested". ***example string ***part in a first cest as the first ceptured group, which is "repested". ***example string ***part in a first cest string with repeated words." ***statemark for the repeated words." ***statemark for the repeated words." ***statemark for the repeated words." ***part all nearbos* ***national string ("repeated") ***part in a first prepated") ***part in a first prepated") ***part in a first prepated") ***part in a first prepated ("repeated") **part in a first prepated ("repeated ("repeated") **part in a first prepated ("repeated") **part in a first prepated ("repeated ("repeat
intercaptive littles Will address the same cert as the first captured group, which is "repeated". ***Example refine the same cert as the first captured group, which is "repeated". ***Example refine it is a cest test straing with repeated repeated words.' ***Example refine it is a cest test straing with repeated words.' ***Example refine it is a cest test straing with repeated words.' ***Example refine it is a cest test straing with repeated words.' ***Example refine it is a cest test straing with repeated words.' ***Example refine it is a cest test straing with repeated words.' ***Example refine it is a cest test straing with repeated words.' ***Example refine words.' ***Example refine words.' ***Example refine words.' ***Example with refine w
intercaptivalities with a control of the control o
intercaptivalities with a control of the case cert as the first captured group, which is "repeated". ***Example refine the same cert as the first captured group, which is "repeated". ***Example refine the same cert as the first string with repeated reseated words." ***Example refine the same cert as the first string with repeated words." ***Example refine the same cert is string with repeated words." ***Example refine the same cert is string with repeated words." ***Example refine the same cert is string with repeated words." ***Example refine words. ***Example refine ***Example
intercaptivalities with a control of the case cert as the first captured group, which is "repeated". ***Example refine the same cert as the first captured group, which is "repeated". ***Example refine the same cert as the first string with repeated reseated words." ***Example refine the same cert as the first string with repeated words." ***Example refine the same cert is string with repeated words." ***Example refine the same cert is string with repeated words." ***Example refine the same cert is string with repeated words." ***Example refine words. ***Example refine ***Example
TABLE OF THE STATE
restrict of the control of the contr
######################################
TEMBERS AND
######################################

temp_var = r"\b(\w+)\s+\1+\b"
nrint(sub(temp_var_r"\1" date_str))

print(sub(temp_var,r"\1", date_str))
Hello this is a sample string which will be will be used for testing