```
1  import re
2  # ============ Characters ===================
3  # . - Matches any character except new line
4  # \. - Mathes a dot.
5  # \\ - Matches backslash
6  # \* - Matches astrick
7  # ============ Character set ===================
8  # [abcd] - any character which matches either 'a' or 'b' or 'c' or 'd'
9  # [^abcd] - any character but not 'a' or 'b' or 'c' or 'd'
10 # [a-z] - any character between 'a' through 'z'
11 # ============ Special Sequences ================
12 # \w - Word character. Same as [a-zA-Z0-9_]. Matches alphanumeric and undersc
13 # \W - Non-Word Character. Same as [^a-zA-Z0-9_]. Matches anything but word c
14 # \d - Matches a digit. Same as [0-9]
15 # \D - Matches a Non-Digit. Same as [^0-9]
16 # \s - Matches only whitespace.
17 # \S - Matches only Non-Whitespace.
18 # ========== Anchors =====================
19 # ^ - Start of String
20 # $ - End of String
21 # \b - Word boudary [a-zA-Z0-9_]
22 # \B - Not a word Boundry
23 # [ ] - Matches characters in square brackets
24 # [^ ] - Matches characters Not in square brackets
25
26 # Meta Characters that needs to be Escaped
27 # . ^ $ * + ? { } [ ] \ | ( )
```

```
28
29  # Quantifiers
30  """
31  1. The standard quantifiers (?, +, *, and {min,max}) are greedy.
32  2. When one of these governs a subexpression, such as a?, (expr )+, [0-9]+,
33     there is a minimum number of matches that are required before it can be co
34     attempt it will ever attempt to match,
35  3. They always attempt to match as many times as they can, up to that maximum
36  4. The only time they settle for anything less than their maximum allowed is
37     ends up causing some later part of the regex to fail.
38  5. plus, question mark and star are called quantifiers, because they influenc
39  """
40  # * - Match expression 0 or more times
41  # + - Match expression 1 or more times
42  # ? - Match expression 0 or 1 times
43  # {min,max} - Matches expression exactly 3 times
44
45  # ========== Grouping ====================
46  # ("A"| "B" | "C") - Either "A" or "B" or "C"
47
48  # re.findall()   # returns a list of all the matches
49  # re.sub()  # replaces one pattern with other
50  # re.finditer()  # returns an iterator object
51  # re.search()  # stops at the first match
52
53  # Word Boundary (\b)
54  # "start of word" boudary is simply the position where a sequence of alphanum
55  # "end of word" is the position where a sequence of alphanumeric characters e
56  # ------------------------------------------------------------------------
57  # Rule: The Match That Begins Earliest (from left to right) Wins
58  # ------------------------------------------------------------------------
59  re.findall(r"the", "the theory of relativity")
60
61  re.findall(r"cat", "The dragging belly indicates your cat is too fat")
62
63  re.findall(r'python', 'python and java are object oriented')
64
65  re.findall(r'aeiou', 'hello how are you doing anna')
66
67  re.findall(r'aeiou', 'hello how are you doing anna, aeiou')
68  # ------------------------------------------------------------------------
69  # Character class or set
70  # ------------------------------------------------------------------------
71  # Matches with both "Smith" and "smith"
72  re.findall(r'[sS]mith', 'smith')
```

```
73  re.findall(r'[sS]mith', 'Smith')
74
75  # Matches separate or saperate
76  re.findall(r's[ea]p[ae]rate', 'seperate')
77  re.findall(r's[ea]p[ae]rate', 'saparate')
78
79  # Match any one character in the character set (either a, e, i, o, u)
80  re.findall(r'[aeiou]', 'hello how are you doing anna')
81
82  # Match either a, b, c, d
83  re.findall(r'[abcd]', 'hello world')
84  re.findall(r'[abcd]', 'abcdefghijk')
85
86  # Matching any number between 0-9
87  re.findall(r'[0123456789]', 'The cost of the book is Rs.100')
88
89  # Matching HTML headers
90  re.findall(r'<h[123456]>', "<h1>")
91  re.findall(r'<h[123456]>', "<h2>")
92  re.findall(r'<h[123456]>', "<h3>")
93  re.findall(r'<h[123456]>', "<h4>")
94  re.findall(r'<h[123456]>', "<h5>")
95  re.findall(r'<h[123456]>', "<h6>")
96  # ------------------------------------------------------------------------
97  # Range "-"
98  # ------------------------------------------------------------------------
99  # Matches any number between 0-9
100 re.findall(r'[0-9]', 'The cost of the book is Rs.100')
101
102 # Matches only lower case letters
103 re.findall(r"[a-z]", 'hello HOW ARE YOU')
104
105 # Matches only upper case letters
106 re.findall(r"[A-Z]", 'hello HOW ARE YOU')
107
108 # Matches all upper case and lower case characters
109 re.findall(r"[a-zA-Z]", 'hello HOW ARE YOU')
110
111 # Matches any number between 1-6
112 re.findall(r"<h[1-6]>", "<h1>")
113 re.findall(r"<h[1-6]>", "<h2>")
114 re.findall(r"<h[1-6]>", "<h3>")
115 re.findall(r"<h[1-6]>", "<h4>")
116 re.findall(r"<h[1-6]>", "<h5>")
117 re.findall(r"<h[1-6]>", "<h6>")
```

```
118
119 # Count total number of Upper case and Lower case letters
120 sentence = "Hello World Welcome To Python"
121 upper_case = re.findall(r'[A-Z]', sentence)
122 lower_case = re.findall(r'[a-z]', sentence)
123
124 print(f'Total No of upper case letters {len(upper_case)}')
125 print(f'Total No of lower case letters {len(lower_case)}')
126
127 # Write a program to count the number of white spaces in a given string
128 sentence = "Hello world welcome to Python Hi  How are you. Hi how are you"
129 spaces = re.findall(r' ', sentence)
130
131 # Write a program to count the number of occurrences of each lower case and u
132 sentence = 'hello@world! welcome!!! Python$ hi how are you & where are you?'
133 chrs = re.findall(r'[a-zA-Z]', sentence)
134 d = {chr: chrs.count(chr) for chr in chrs}
135
136 # ------------------------------------------------------------------------
137 # Meta Character "+" (matches 1 or more occurances of previous expression)
138 # ------------------------------------------------------------------------
139 re.findall(r'[0-9]+', 'The cost of the book is Rs.100')
140
141 re.findall(r'[abcd]+', 'abcdefg hijkab')
142
143 re.findall(r'an+a', 'annnnnnnnnna')
144
145 # Matches each word in the string
146 re.findall(r"[a-zA-Z]+", "Hello World Welcome To Python Programming Pyt123on"
147
148 # Count the characters in each word. Please ignore special characters if ther
149 sentence = "Hi there! How are you:) How are you doing today!"
150 words = re.findall(r'[a-zA-Z]+', sentence)
151 word_len = { word: len(word) for word in words}
152
153 # Sum all the numbers in the below string.
154 word = "Pytho12nReg567exp2" # 1 + 2 + 5 + 6 + 7 + 2
155 total = 0
156 numbers = re.findall(r'[0-9]', word)
157 for number in numbers:
158     total += int(number)
159
160 # Adding 12 + 567 + 2
161 word = "Pytho12nReg567exp2"
162 total = 0
```

```
numbers = re.findall(r'[0-9]+', word)
for number in numbers:
    total += int(number)

# Match file names and extensions
message = "Downloading file archive.zip to downloads folder..."
# image.jpeg
# index.xhtml
# python.py
re.findall(r'[a-z]+\.[a-z]+', message)
# ---------------------------------------------------------------------------
# Meta Character "?" (matches 0 or 1 occurance of previous expression)
# ---------------------------------------------------------------------------
re.findall(r'colou?r', "what color do you like")

re.findall(r'https?://', 'https://www.google.com')

re.findall(r'https?://', 'http://www.google.com')

re.findall(r'July?', "Jul the 26th day")

re.findall(r'an?a', "ana")

re.findall(r'an?a', "anna")
# ---------------------------------------------------------------------------
# Meta Character "*" (matches 0 or more occurances of previous expression)
# ---------------------------------------------------------------------------
re.findall(r"an*a", "hello ana")

re.findall(r"an*a", "hello aa")

re.findall(r"an*a", "hello annna")

# Regular Expression for Matching Inbox, Inbox(1), .... Inbox(N)
re.findall(r"Inbox\(?\d*\)?", "Inbox(10)")
re.findall(r"Inbox\(?\d*\)?", "Inbox")
# ---------------------------------------------------------------------------
# Negation "^"
# ---------------------------------------------------------------------------
# Matches everything apart from numbers between 0-9
re.findall(r'[^0-9]', 'The cost of the book is Rs.100')

# Matches everything apart from alphabets 'a', 'b', 'c' and 'd'
re.findall(r'[^abcd]', 'abcdefg hijkab')

```

```
# Matches everything apart from numbers between 0-9
re.findall(r'[^0-9]+', 'The cost of the book is Rs.100')

re.findall(r'[^abcd]+', 'abcdefg hijkab')

# Match only those characters excepts digits
word = '@hello12world34welcome!123'
re.findall(r'[^0-9]', word)

# Count the number of special characters in the below string
sentence = 'hello@world! welcome!!! Python$ hi26 how are you & where are you?
re.findall(r"[^a-zA-Z0-9_\s]", sentence)
# ------------------------------------------------------------------------
# Starts with "^" and ends with "$"
# ------------------------------------------------------------------------
re.findall(r"^hello", "hello world")

re.findall(r"^hello", "world hello")

re.findall(r"hello$", "world hello")

re.findall(r"hello$", "hello world")

re.findall(r'hello$', 'hello world welcome to python')

# Matching the only those lines which ends with "UDP"
with open("./data_files/sample.log") as f:
    for line in f:
        match = re.findall(r".*UDP$", line)
        if match:
            print("".join(match))

# string starts with "hello" and ends with "hello" (meaning exactly one word
re.findall(r"^hello$", "hello")

# Phone Number pattern (4DIGITS-3DIGITS-3DIGITS)
re.findall(r'\d{3}-\d{3}-\d{4}', '456-9832-098')

# matching only 800 and 900 numbers
re.findall(r"^[89]00-\d{3}-\d{4}", '800-123-123')
# ------------------------------------------------------------------------
# Word Boundary (\b) The expression should be a word boundry
# (Transition between non-word character to word character or word character
# ------------------------------------------------------------------------
# starts with word boundry
```

```
253 re.findall(r"\bday", "what a beautiful day today is")
254
255 # ends with word boundry
256 re.findall(r"day\b", "what a beautiful day today is")
257
258 # starts and ends with word boundry
259 re.findall(r"\bday\b", "what a beautiful day today is")
260
261 re.findall(r"\b[0-9]{6}\b", 'Pincode of Bangalore is 560001 and the number is
262
263 # Regular expression which matches words that starts with "h"
264 re.findall(r"\bh[a-zA-Z0-9_]+", 'hello world hi hello universe how are you ha
265
266 # Regular expression which matches words that starts with "P or J"
267 re.findall(r"\b[PJ][a-zA-Z0-9_]+", 'Python is a programming language. Python
268
269 # Regular expression which matches words that ends with "y"
270 re.findall(r"[a-zA-Z0-9_]+y\b", 'hello world hi hello universe how are you ha
271
272 # print only those words starting with vowel character
273 sentence = "hello hi american engieers and indian writers officers united sta
274 words = re.findall(r"\b[aeiou][a-zA-Z0-9_]+", sentence)
275
276 # Matches only Capital Letter words
277 re.findall(r"\b[A-Z]+\b", "This is PYTHON programming LANGUAGE and REGEX")
278
279 # Matches only lower case words
280 re.findall(r"\b[a-z]+\b", "This is PYTHON programming LANGUAGE and REGEX")
281
282 # Matching only pdf files
283 re.findall(r"[a-zA-Z0-9]+\.pdf\b", "downloading apple.pdf to downloads folder
284
285 # Regular expression for matching only 3 letter words in the given string
286 sentence = "hello hi how are you what is your name he is older than me how ol
287 re.findall(r'\b[a-zA-Z0-9_]{3}\b', sentence)
288 # o/p ['how', 'are', 'you', 'how', 'old', 'are', 'you']
289
290 # Extract only 4 digit numbers from the string
291 re.findall(r"\b\d{4}\b", "Copyright 1998. All rights reserved")
292
293 # Regular expression for matching the words which starts with "he"
294 sentence = "he helps the community and he is the hero of the day"
295 re.findall(r"\bhe[a-zA-Z0-9_]*", sentence)
296
297 # Regular expression for matching the words which either starts with "he" or
```

```
298 sentence = "he helps the community and he is the hero of the day she sells se
299 re.findall(r"\b(?:he|se)[a-zA-Z0-9_]*", sentence)
300
301 # Regular Expression - PAN Number
302 sentence = "my pan number is ABCDE1234X and the other number is XYZTR3104J id
303 re.findall(r'\b[A-Z]{5}[0-9]{4}[A-Z]\b', sentence)
304
305 # Different Combintations
306 line = "03/22 08:51:06 WARNING :.....mailslot_create: setsockopt(MCAST_ADD) f
307 re.findall(r"[A-Z]+", line)
308 re.findall(r"\b[A-Z]+", line)
309 re.findall(r"[A-Z]+\b", line)
310 re.findall(r"\b[A-Z]+\b", line)
311
312 # Matches all digits
313 re.findall(r"\d", "654 this string is starting with 12 and ending with number
314
315 # Matches whole numbers
316 re.findall(r"\d+", "654 this string is starting with 12 and ending with numbe
317
318 # Matches only 3 Digit numbers
319 re.findall(r"\d{3}", "654 this string is starting with 12 and ending with num
320
321 # Matches all digit numbers
322 re.findall(r"\b\d{3}\b", "654 this string is starting with 12 and ending with
323
324 # Matches the string that ends with 3 digit number
325 re.findall(r"\b\d{3}\b$", "4632746327 this string is ending with 235")
326
327 # Matches the string that starts with 3 digit number
328 re.findall(r"^\b\d{3}\b", "654 this string is starting with and ending with n
329 # ------------------------------------------------------------------------
330 # Groups ( )
331 # ------------------------------------------------------------------------
332 # Matches either "python" or "java"
333 re.findall('(python|java)', 'python and java are object oriented')
334
335 # Matches 09:00 am/pm or 9:00 am/pm
336 sentence = 'The meeting is between 9:00 am and 12:30 pm'
337 re.findall(r'[0-9]?[0-9]:[0-9][0-9]\s(?:am|pm)', sentence)
338
339 # Regular Expression - YYYY-MM-DD date format
340 _dates = ['2019-01-02', '2019-13-02', '2019-12-26', '26-08-2019', '20-19-20',
341 re.findall(r'\d{4}-(?:0[1-9]|[12][0-9]|3[01])-(?:0[1-9]|[12][0-9]|3[01])', '2
342
```

```
343 # Regular Expression – 24hr time format
344 _formats = ['00:00:00', '23:59:59', '24:00:00', '1:59:20', '12:9:10', '10:20:
345 re.findall(r"(?:[01]\d|2[0-3]):[0-5]\d:[0-5]\d",'23:59:59')
346 # ------------------------------------------------------------------------
347 # dot "." matches with everything
348 # ------------------------------------------------------------------------
349 re.findall(r'.', "hello world")
350
351 re.findall(r'h.', "hello")
352
353 re.findall(r'h.', "hello world hi how how are you")
354
355 re.findall(r'a.b', "acb")
356
357 re.findall(r'a.b', "a b")
358
359 re.findall(r'a.b', "ab")
360
361 re.findall(r'a.b', "a*b a?b")
362
363 re.findall(r'a.b', "abcde")
364
365 re.findall(r'a.a', "ana")
366
367 re.findall(r'a..a', "anna")
368
369 re.findall(r'a.*a', "annnnnna")
370
371 re.findall(r'a.*a', 'aa')
372
373 re.findall(r"^a.*a$", "anna")
374
375 re.findall(r"^a.*a$", "hello anna")
376
377 re.findall(r'a.*a', 'abcad')
378
379 re.findall(r'a.*a$', 'abcad')
380
381 re.findall(r'a.*a$', 'abcada')
382
383 re.findall(r'a.+a', 'ana')
384
385 re.findall(r'a.+a', 'aa')
386 # ------------------------------------------------------------------------
387 # ---------------------------------- Back-referencing ----------------------
```

```
"""
1. Back-referencing is a regular-expression feature which allows you to match
matched earlier in the expression without specifically knowing the text when
2. Back-references provide a convenient way to identify a repeated character
For example, if the input string contains multiple occurrences of an arbitrar
first occurrence with a capturing group, and then use a backreference to matc
3. \1 will try to match what ever is matched in the first bracket
"""
# Repeated word sequences
m = re.findall(r"(world)\1", "thethe python hello worldworld the")

# Repeated words
m = re.findall(r"([a-z]+\s)\1", "the the python hello world world the")

# Repeated words for 2 consequtive times
m = re.finditer(r"([a-z]+\s)\1{2}", "the the python hello world world the the

# Repeated characters
m = re.findall(r"([a-z])\1", "hello hurry programming")

# Repeated numbers
n = re.findall(r"([0-9])\1", "hello 123345, welcome to 001 98799")

# Repeated numbers pattern
n = re.findall(r"([0-9]+\s)\1", "hello 12345 12345 , welcome to 001 98799")

n = re.findall(r"([0-9])+", "hello 1234512345 , welcome to 00198799")

# finding the words that are repeated at the beginning and at the end of the
sentence = "hello world welcome to regex hello"
re.findall(r"^([a-z]+).*\1$", sentence)
# ------------------------------------------------------------------------
# Replacing patterns
# ------------------------------------------------------------------------
# Replace whitespaces with newline character in the below string
sentence = "Hello world welcome to python"
words = re.sub(r'\s', '\n', sentence)
print(words)

# Replace all vowels with "*"
sentence = "hello world welcome to python"
words = re.sub(r'[aeiou]', '*', sentence)
print(words)

# Replace all occurances of digits with "*"
```

```
433 sentence = 'hello123world welcome456to python012'
434 words = re.sub(r'\d', '*', sentence)
435
436 # Replace all occurances of special characters with "*"
437 sentence = 'hello#$%world welcome@!#$%to python*&^%'
438 words = re.sub(r'[^a-zA-Z\s]', "*", sentence)
439
440 # Replace "And" with "&"
441 sentence = "Java and Python are programming languages"
442 _sentence = re.sub(r"\sAnd\s", " & ", sentence)
443
444 # Replace all occurances of "Java" with "Python" in a file
445 with open('java.txt', 'r') as jf:
446     with open('python.txt', 'a') as pf:
447         for line in jf:
448             new_line = re.sub('Java', 'Python', line)
449             pf.write(new_line)
450 # ------------------------------------------------------------------------
451 # re.finditer()
452 matches = re.finditer(r"hello", 'hello world welcome to python hello world')
453 for match in matches:
454     print(match.group())
455
456 # Write a program to find the index of nth occurrence of a sub-string in a st
457 sentence = "hello world welcome to python hello hi how are you hello there"
458 matches = re.finditer(r'hello', sentence)
459 positions = [ (match.start(), match.end()) for match in matches]
460
461 # re.search()
462 match = re.search(r"hello", 'hello world hello world')
463 print(match.group())
464 # ------------------------------------------------------------------------
465 # Regular Expression - IP Addresses
466 # ------------------------------------------------------------------------
467 id_address_format = '\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}'
468 ips = ['10.1.2.3', '127.0.0.0', '199.99.9.9', '199.9.9999.9', '127-0-0-0']
469 # ------------------------------------------------------------------------
470 # Regular Expression - Email format
471 # ------------------------------------------------------------------------
472 email_pattern = r'[\w-]+\.?[\w-]@[\w]+\.(com|edu|in|gov)'
473 emails = ['test.user@company.com',
474           'test.user2@company.com',
475           'test_user@company.com',
476           'testing@company.com',
477           'test-T.user@company.com',
```

```
478           'testing@company',
479           'testingcompany.com'
480           ]
481 # ------------------------------------------------------------------------
482 # Regular Expression - URL Pattern
483 url_pattern = r'https?://[\w.]+'
484 urls = ['http://www.youtube.com',
485         'https://www.google.com',
486         'http://www.amazon.in',
487         'https://www.mail.yahoo.com',
488         'ftp://test.com'
489         'https://www.facebook.com/'
490         ]
491 # ------------------------------------------------------------------------
492 # Count the number of white spaces in the file
493 def count_spaces():
494     with open('./data_files/sample.log') as f:
495         white_spaces = 0
496         for line in f:
497             count = len(re.findall(r"\s", line))
498             white_spaces += count
499     return white_spaces
500 # ------------------------------------------------------------------------
501 # Count the number of Capital Letter words in the file
502 def count_caps():
503     with open('./data_files/sample.log') as f:
504         capital_words = 0
505         for line in f:
506             count = len(re.findall(r"\b[A-Z]+\b", line))
507             capital_words += count
508     return capital_words
509 # ------------------------------------------------------------------------
510 # Count the number of Capital Letters in the file
511 def count_cap_letters():
512     with open('./data_files/sample.log') as f:
513         capital_letters = 0
514         for line in f:
515             count = len(re.findall(r"[A-Z]", line))
516             capital_letters += count
517     return capital_letters
518 # ------------------------------------------------------------------------
519 # Count the number of INFO, TRACE, WARNING, EVENT messages in the file
520 def count_messages(message_name):
521     with open('./data_files/sample.log') as f:
522         message_count = 0
```

```
523         for line in f:
524             count = len(re.findall(message_name, line))
525             message_count += count
526     return message_count
527 # ------------------------------------------------------------------------
528 # Extract all the ip addresses in the sample log file
529 def get_all_ip():
530     ips = [ ]
531     with open('./data_files/sample.log') as f:
532         for line in f:
533             ip = re.findall(r'\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}', line)
534             if ip:
535                 for item in ip:
536                     ips.append(item)
537     return ips
538 # ------------------------------------------------------------------------
539 # Count the number of words in the file
540 def count_words():
541     word_count = 0
542     with open('./data_files/sample.log') as f:
543         for line in f:
544             words = re.findall(r"\b[A-Za-z]+\b", line)
545             word_count += len(words)
546     return word_count
547 # ------------------------------------------------------------------------
548 # Count total number of characters (a-zA-Z) in the file. Ignore digits, white
549 def count_total_letters():
550     with open('./data_files/sample.log') as f:
551         total_letters = 0
552         for line in f:
553             count = len(re.findall(r"[a-zA-Z]", line))
554             total_letters += count
555     return total_letters
556 # ------------------------------------------------------------------------
557 # Stripping leading and trailing whitespaces
558 def regex_strip(line):
559     stripped_line = re.sub(r"^\s*|\s*$", "", line)
560     return stripped_line
561 # ------------------------------------------------------------------------
562 # Flitering only floating point values from file
563 with open("./data_files/points.txt") as f:
564     floats = [ ]
565     for line in f:
566         match = re.findall(r"-?[0-9]+\.[0-9]+", line)
567         # match = re.findall(r"-?[0-9]+\.[0-9]{3}", line) # matches 3 digits
```

```
568             for item in match:
569                 floats.append(item)
570     # ------------------------------------------------------------------------
571     # ---------------------------------- LookAhead and LookBehind Anchors -------
572             # Under development ...
573     # ------------------------------------------------------------------------
574     # Misc
575     # ------------------------------------------------------------------------
576     # Finding the all the links in a html document using regex
577     from requests import request
578     response = request("GET", "http://demowebshop.tricentis.com/")
579     html_code = response.text
580     links = re.findall(r"\s*<a\s.+", html_code)
581     links = [ link.strip() for link in links ]
582     # ------------------------------------------------------------------------
```